Введение в статистическое обучение с примерами на языке R

Джеймс Г.

Уиттон Д.

Хасти Т.

Тибширани Р.

An Introduction to Statistical Learning with Applications in R

Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani

Введение
в статистическое обучение
с примерами на языке R

Джеймс Г.
Уиттон Д.
Хасти Т.
Тибширани Р.

ДМК Пресс
издательство
Москва, 2017

УДК 519.25/.6:004.434R
ББК 22.17с5
Д40

Джеймс Г., Уиттон Д., Хасти Т., Тибширани Р.

Д40 Введение в статистическое обучение с примерами на языке R. Пер. с англ. С. Э. Мастицкого – М.: ДМК Пресс, 2017. – 456 с.: ил.

ISBN 978-5-97060-495-3

Книга представляет собой доступно изложенное введение в статистическое обучение – незаменимый набор инструментов, позволяющих извлечь полезную информацию из больших и сложных наборов данных, которые начали возникать в последние 20 лет в таких областях, как биология, экономика, маркетинг, физика и др. В этой книге описаны одни из наиболее важных методов моделирования и прогнозирования, а также примеры их практического применения. Рассмотренные темы включают линейную регрессию, классификацию, создание повторных выборок, регуляризацию, деревья решений, машины опорных векторов, кластеризацию и др. Описание этих методов сопровождается многочисленными иллюстрациями и практическими примерами. Поскольку цель этого учебника заключается в продвижении методов статистического обучения среди практикующих академических исследователей и промышленных аналитиков, каждая глава включает примеры практической реализации соответствующих методов с помощью R – чрезвычайно популярной среды статистических вычислений с открытым кодом.

Издание рассчитано на неспециалистов, которые хотели бы применять современные методы статистического обучения для анализа своих данных. Предполагается, что читатели ранее прослушали лишь курс по линейной регрессии и не обладают знаниями матричной алгебры.

УДК 519.25/.6:004.434R
ББК 22.17с5

Translation from the English language edition:
An Introduction to Statistical Learning
by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani

ISBN 978-1-4614-7137-0 (англ.)

ISBN 978-5-97060-495-3 (рус.)

Оглавление

От переводчика

В последние несколько лет наблюдается небывалый рост объема, скорости получения и сложности данных в самых разных областях жизнедеятельности человека. Неудивительно, что и спрос на специалистов, способных извлечь полезную информацию из этих потоков данных, сегодня высок, как никогда раньше. Важную роль в подготовке таких специалистов играет учебная литература по современным методам статистического анализа. Написать хороший учебник — это титанический труд, однако авторы книги, которую Вы сейчас держите в руках, справились с этой задачей блестяще. Простота изложения материала, многочисленные практические примеры и хорошо продуманные лабораторные работы и упражнения сделали книгу «An Introduction to Statistical Learning with Applications in R» чрезвычайно популярной в академических кругах и среди аналитиков коммерческих организаций во всем мире. Для меня было честью выполнить перевод этой работы, и я рад, что теперь она стала доступной и для русскоязычных читателей.

К сожалению, в первом издании этой книги на русском языке, которое вышло в апреле 2016 г., был найден целый ряд опечаток и ошибок, возникших в ходе верстки[1]. Все обнаруженные с тех пор ошибки были учтены и исправлены в настоящем издании, за что я безмерно благодарен помогавшим с этой работой читателям. В случае обнаружения новых недостатков, сообщайте, пожалуйста, по адресу rtutorialsbook@gmail.com.

Я благодарен Дмитрию Мовчану и всей команде «ДМК Пресс» за помощь с подготовкой и изданием этой книги, а также Артему Груздеву, Дмитрию Дерябину и Александру Вишератину за оказанные ими консультации и советы по улучшению первых вариантов рукописи. Наконец, я хотел бы поблагодарить свою жену Светлану за ее поддержку во всех моих начинаниях, одним из которых стала работа над этим переводом.

Сергей Мастицкий *Лондон, декабрь 2016 г.*

[1] Полный список этих опечаток и ошибок можно найти на GitHub-странице книги: `https://github.com/ranalytics/islr-ru`.

Предисловие

К статистическому обучению относят набор инструментов, предназначенных для моделирования и понимания сложно организованных данных. Это недавно разработанная область статистики, которая развилась параллельно с достижениями в компьютерных науках и особенно машинном обучении. Данная область охватывает многие методы, включая лассо и разреженную регрессию, классификационные и регрессионные деревья, бустинг и метод опорных векторов.

Одновременно со взрывообразным ростом круга задач, связанных с «большими данными», статистическое обучение стало очень популярным во многих научных областях, а также в маркетинге, финансах и других бизнес–дисциплинах. Люди с навыками статистического обучения очень востребованны.

Одна из первых книг в этой области — «*Основы статистического обучения*» (ОСО)[2] — была опубликована в 2001 г., а в 2009 г. вышло ее второе издание. ОСО стала очень популярной книгой среди не только статистиков, но и специалистов из смежных областей. Одна из причин такой популярности заключается в относительно легкодоступном стиле изложения. Однако ОСО предназначена для людей с основательной математической подготовкой. Новая книга «*Введение в статистическое обучение*» возникла в связи с ощутимой необходимостью в более широком и не таком техническом изложении материала. В этой новой книге мы освещаем многие из тех же тем, которые присутствуют в ОСО, но уделяем основное внимание практическому применению соответствующих методов, а не их математическим деталям. Мы разработали лабораторные работы, иллюстрирующие реализацию каждого метода с использованием статистического пакета R. Эти лабораторные работы позволяют читателю получить ценный практический опыт.

Эта книга подойдет для студентов и магистрантов, углубленно изучающих статистику или родственные дисциплины, а также для представителей других наук, которые желают применять инструменты статистического обучения для анализа своих данных. Ее можно использовать в качестве учебника для курса, длящегося один или два семестра.

Мы благодарим за ценные комментарии следующих читателей черновых вариантов этой книги: Паллави Басу, Александру Чулдечову, Патрика Данахера, Уилла Фитиана, Луэллу Фу, Сэма Гросса, Макса Гразьера Г'Селла, Кортни Паулсон, Ксингао Кьяо, Элизу Шенг, Ноа Симон, Кена Минга Тана и Ксина Лу Тана.

[2] Hastie T., Tibshirani R., Friedman J. (2001) The Elements of Statistical Learning. Springer, 745 p.

«Делать предсказания трудно, особенно в отношении будущего».

Йоги Берра

Джеймс Гарет (Лос–Анджелес, США)

Даниела Уиттен (Сиэттл, США)

Тревор Хасти (Пало Альто, США)

Роберт Тибширани (Пало Альто, США)

Глава 1

Введение

Обзор задач статистического обучения

Под *статистическим обучением* понимают огромный набор инструментов, предназначенных для *понимания данных*. Эти инструменты можно разделить на две группы: *обучение с учителем* и *обучение без учителя*. В общих чертах статистическое обучение подразумевает построение статистической модели для предсказания, или оценивания, некоторой *выходной переменной* на основе одной или нескольких *входных переменных*. Подобные проблемы встречаются в настолько разнящихся областях, как бизнес, медицина, астрофизика и государственное управление. При обучении без учителя имеются входные переменные, но нет предсказываемой переменной; тем не менее мы можем выявить закономерности и структуру в таких данных. В качестве иллюстрации некоторых практических приложений статистического обучения ниже мы кратко обсудим три реальных набора данных, рассматриваемых в этой книге.

Данные по заработной плате

В этом примере мы исследуем связь нескольких факторов с уровнем заработной платы у группы мужчин из центрально–атлантического региона США (в этой книге мы будем ссылаться на соответствующие данные как «набор данных `Wage`»). В частности, мы хотим выяснить зависимость между заработной платой работника (переменная `wage`) и его возрастом (`age`), уровнем образования (`education`), а также календарным годом (`year`). Посмотрите, например, на график, представленный слева на рис. 1.1, где показана связь между заработной платой и возрастом работников из этого набора данных. Имеется свидетельство в пользу того, что `wage` увеличивается по мере возрастания `age`, а затем снова снижается примерно после 60 лет. Синяя линия, которая соответствует оценке среднего уровня `wage` для заданного значения `age`, позволяет увидеть этот тренд более четко.

Зная возраст работника, мы можем *предсказать* его заработную плату по этой кривой. Однако на рис. 1.1 также хорошо виден значительный разброс относительно этого среднего значения, из чего следует, что сама по себе переменная `age` вряд ли позволит с большой точностью предсказать `wage` для конкретного человека.

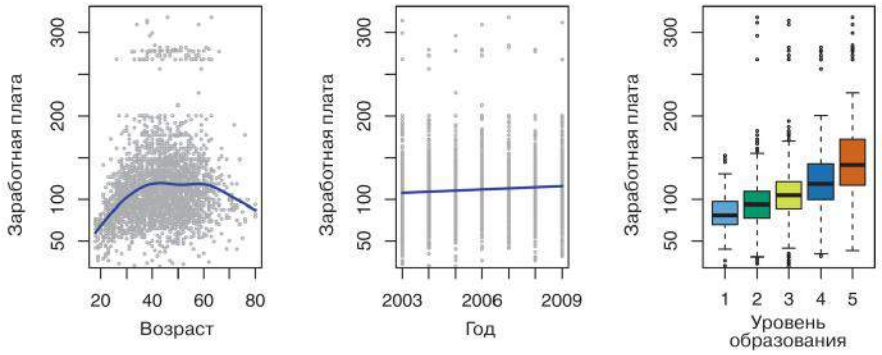

РИСУНОК 1.1. *Таблица* `Wage` *с данными по заработной плате мужчин из центрально–атлантического региона США.* Слева: `wage` *как функция от* `age`. *В среднем* `wage` *увеличивается одновременно с* `age` *до возраста около 60 лет, после чего начинает снижаться.* В центре: `wage` *как функция от* `year`. *В период с 2003 по 2009 г. имеет место медленный, но устойчивый рост* `wage` *в среднем на 10 000$ в год.* Справа: *диаграмма размахов* `wage` *как функции от* `education`, *где 1 соответствует самому низкому уровню образования (неоконченная средняя школа), а 5 – самому высокому уровню (ученая степень). В среднем* `wage` *возрастает с уровнем образования*

У нас имеется также информация по уровню образования каждого работника и его заработной плате `wage` за каждый год `year`. Графики, представленные в центре и справа на рис. 1.1, показывают `wage` в зависимости от `year` и `education` и свидетельствуют о том, что каждый из этих факторов связан с `wage`. С 2003 по 2009 г. значения зарплаты с каждым годом линейно возрастают примерно на 10 000$, хотя этот рост очень слабый, по сравнению с разбросом в данных. Зарплаты также выше у людей с более высоким уровнем образования: работники с наименьшим уровнем образования (1) в целом зарабатывают гораздо меньше, чем работники с самым высоким уровнем (5). Очевидно, что наиболее точное предсказание `wage` для конкретного человека будет получено при объединении информации по его возрасту `age`, уровню образования `education` и году `year`. В главе 3 мы обсудим линейную регрессию, которую можно применить для предсказания `wage` по этим данным. В идеале мы должны предсказывать `wage` с учетом нелинейного характера связи этой переменной с `age`. В главе 7 мы рассмотрим класс методов, предназначенных для решения данной проблемы.

Данные по рынку акций

В случае с набором данных `Wage` предсказывается *непрерывное*, или *количественное*, выходное значение. Часто такую ситуацию называют проблемой *восстановления регрессии*. Однако в некоторых случаях мы можем столкнуться с необходимостью предсказать нечисловое значение, т. е. *категориальную*, или *качественную*, выходную переменную. Так, в главе 4 мы рассмотрим набор данных по рынку акций, который описывает днев-

ные изменения индекса Standard & Poor's 500 (S&P) в течение 5–летнего периода (с 2001 по 2005 г.). Мы будем ссылаться на него как на «набор данных `Smarket`». Задача заключается в предсказании *возрастания* или *снижения* индекса на основе его удельного изменения за последние 5 дней. Здесь проблема статистического обучения не подразумевает предсказания числового значения. Вместо этого предсказывается рост (`Up`) или снижение (`Down`) рынка акций для того или иного дня. Это известно как проблема *классификации*. Модель, способная с высокой точностью предсказывать направление движения рынка, была бы очень полезной!

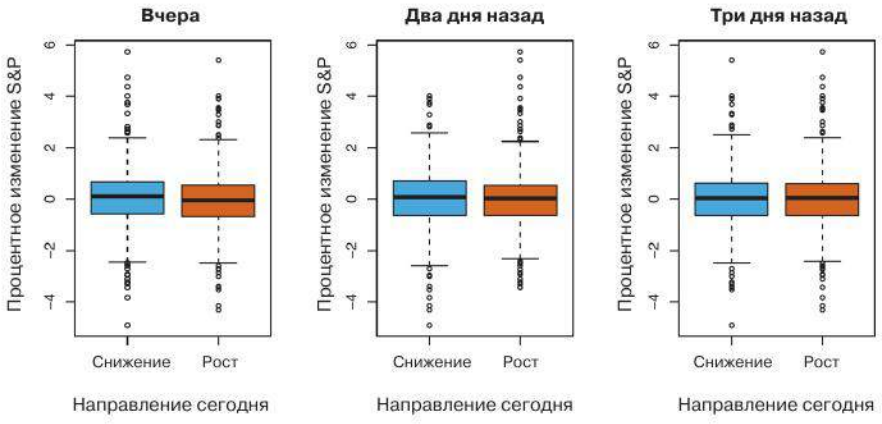

РИСУНОК 1.2. Слева: *диаграмма размахов, отражающая процентное изменение индекса S&P по сравнению со вчерашним значением для дней, когда происходили рост или снижение рынка (по данным* `Smarket`). В центре и справа: *то же, но показаны процентные изменения по сравнению с двумя и тремя предыдущими днями соответственно*

На рис. 1.2 слева представлена диаграмма размахов, отражающая процентные изменения индекса акций по сравнению с предыдущим днем: для 648 дней, когда в следующие за ними дни рынок вырос, и для 602 дней, когда рынок ушел вниз. Эти две диаграммы почти идентичны, что указывает на невозможность простой стратегии по использованию вчерашнего состояния индекса S&P для предсказания его сегодняшнего положения. Остальные графики, на которых приведены диаграммы размахов для процентных изменений в сравнении с двумя и тремя предыдущими днями, также указывают на отсутствие выраженной связи между прошлым и текущим состояниями индекса. Безусловно, отсутствие связи здесь ожидаемо, иначе при наличии тесных корреляций между следующими друг за другом днями мы могли бы использовать простую торговую стратегию для получения прибыли. Тем не менее в главе 4 мы подробно исследуем эти данные при помощи нескольких методов статистического обучения. Интересно, что есть некоторые указания на наличие слабых закономерностей в этих данных, предполагающие возможность правильного предсказания направления движения рынка примерно в 60% случаев (по крайней мере, для этого 5–летнего периода; рис. 1.3).

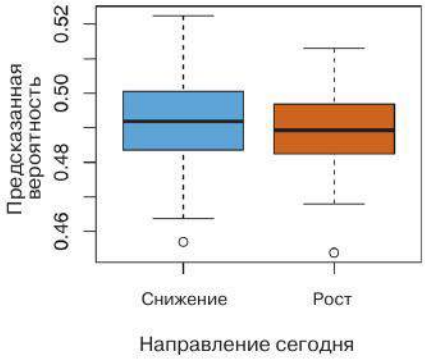

РИСУНОК 1.3. *Мы подогнали квадратичную дискриминантную модель для части данных* `Smarket`*, соответствующей периоду с 2001 по 2004 г., и предсказали вероятность снижения рынка акций для данных за 2005 г. В среднем предсказанная вероятность снижения рынка выше для дней, когда снижение в действительности имело место. На основе этих результатов мы можем правильно предсказать направление движения рынка в 60% случаев*

Данные по экспрессии генов

Два предыдущих примера иллюстрируют данные, в которых есть как входные, так и выходные переменные. Однако еще один важный класс проблем охватывает ситуации, в которых мы наблюдаем только входные переменные, без соответствующей зависимой переменной. Например, выполняя маркетинговые исследования, мы могли бы располагать демографической информацией для ряда уже имеющихся или потенциальных клиентов. У нас может возникнуть желание понять, какие клиенты похожи друг на друга, для чего мы объединили бы отдельных людей в группы в соответствии с их наблюдаемыми характеристиками. Такая ситуация известна как проблема *кластеризации*. В отличие от предыдущих примеров, здесь мы не пытаемся предсказать какую–либо выходную переменную.

Мы посвящаем главу 10 обсуждению методов статистического обучения, предназначенных для решения проблем, в которых нет естественной выходной переменной. Мы рассматриваем набор данных `NCI60`, состоящий из 6830 значений уровня экспрессии генов в 64 линиях раковых клеток. Вместо предсказания какой–то конкретной выходной переменной нам интересно выяснить наличие групп, или кластеров, среди этих клеточных линий на основе измерений генной экспрессии. Решить этот вопрос нелегко, отчасти из–за наличия тысяч значений уровня экспрессии для каждой линии, которое затрудняет визуализацию данных.

График, показанный на рис. 1.4 слева, решает эту проблему путем представления каждой из 64 клеточных линий при помощи всего лишь двух чисел — Z_1 и Z_2. Это первые две *главные компоненты* данных, которые сводят 6830 значений уровня экспрессии по каждой линии до двух чисел, или *измерений*. Несмотря на вероятность потери некоторой части информации в результате такого снижения размерности, теперь появля-

ется возможность визуально исследовать данные на наличие кластеров. Выбор числа кластеров часто бывает трудной проблемой. Однако график, приведенный на рис. 1.4 слева, указывает на наличие не менее четырех групп клеточных линий, которые мы пометили разными цветами. Теперь мы можем подробнее изучить клеточные линии из каждого кластера на предмет их сходства по типу рака и тем самым лучше понять взаимосвязь между уровнями генной экспрессии и раком.

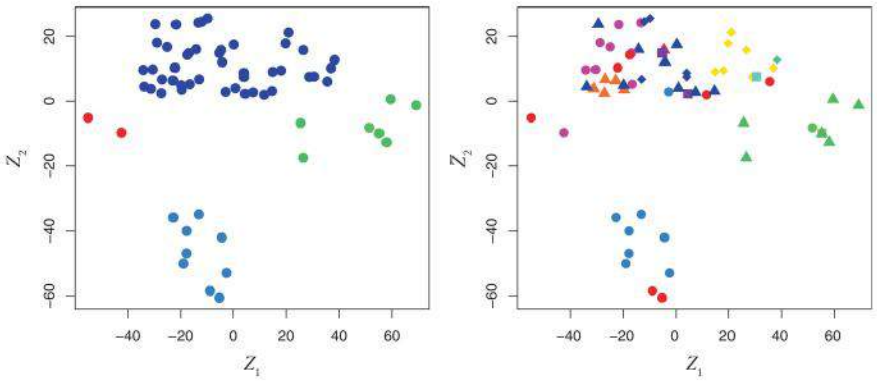

РИСУНОК 1.4. Слева: *представление данных по уровню экспрессии генов* NCI60 *в двумерном пространстве, образованном переменными* Z_1 *и* Z_2. *Каждая точка соответствует одной из 64 клеточных линий. Клеточные линии образуют примерно четыре группы, которые мы представили разными цветами.* Справа: *то же, что и слева, за тем исключением, что мы выделили каждый из 14 типов рака при помощи символов разной формы и цвета. Клеточные линии, соответствующие одному типу рака, стремятся располагаться в этом двумерном пространстве рядом*

Оказывается, что в случае с этим конкретным набором данных клеточные линии соответствуют 14 различным типам рака. (Эта информация, однако, не была использована при создании диаграммы на рис. 1.4 слева.) Справа на рис. 1.4 показаны те же данные, однако 14 типов рака отмечены символами разной формы и цвета. Хорошо видно, что клеточные линии с одинаковым типом рака стремятся располагаться близко друг к другу в этом двумерном представлении. Кроме того, несмотря на то что информация по раку не была использована для создания первого графика, полученная кластеризация в значительной мере соответствует действительным типам рака, наблюдаемым на втором графике. Это в определенной мере является независимым подтверждением верности нашего кластерного анализа.

Краткая история развития статистического обучения

Несмотря на то что термин «*статистическое обучение*» достаточно новый, многие из основополагающих концепций этой дисциплины были раз-

работаны очень давно. В начале XIX века Лежандр и Гаусс опубликовали статьи по *методу наименьших квадратов*. Этот подход был впервые успешно применен для решения проблем астрономии. Линейная регрессия используется для предсказания значений количественных переменных, таких, например, как заработная плата. Для предсказания значений качественных переменных (например, выживет пациент или нет, пойдет рынок акций вверх или вниз) Фишер в 1936 г. предложил *линейный дискриминантный анализ*. В 1940–х г. разные авторы предложили альтернативный подход — *логистическую регрессию*. В начале 1970–х г. Нельдер и Веддербурн ввели термин «*обобщенные линейные модели*» для целого класса методов статистического обучения, которые включают как линейную, так и логистическую регрессию в качестве частных случаев.

К концу 1970–х г. стали доступными многие другие методы обучения на основе данных. Однако почти всегда это были линейные методы, поскольку с вычислительной точки зрения подгонка *нелинейных* зависимостей в то время была неосуществимой. К началу 1980–х гг. вычислительные технологии, наконец, были усовершенствованы до уровня, который больше не ограничивал работу с нелинейными методами. В середине 1980–х г. Брейман, Фридман, Ольшен и Стоун ввели *деревья регрессии и классификации* и стали одними из первых, кто детально продемонстрировал большой потенциал для практической реализации этого метода, включая перекрестную проверку для выбора модели. В 1986 г. Хасти и Тибширани ввели термин «*обобщенные аддитивные модели*» для класса нелинейных дополнений обобщенных линейных моделей, а также разработали соответствующее программное обеспечение.

С тех пор благодаря появлению *машинного обучения* и других дисциплин, статистическое обучение развилось в новую ветвь статистики, уделяющую основное внимание обучению с учителем и без учителя, а также прогнозированию. В последние годы прогресс в статистическом обучении был связан с ростом доступности мощного и относительно удобного программного обеспечения, каковым является популярная и бесплатная система `R`. Потенциально это может привести к дальнейшей трансформации дисциплины из набора методов, используемых и разрабатываемых статистиками и специалистами в области компьютерных наук, в неотъемлемый набор инструментов для гораздо более широкого сообщества.

Об этой книге

Книга «Основы статистического обучения» (ОСО), написанная Хасти, Тибширани и Фридманом, была впервые опубликована в 2001 г. С тех пор она превратилась в важную справочную работу по фундаментальным основам статистического обучения. Ее успех обусловлен широким и детальным рассмотрением многих тем статистического обучения, а также тем фактом, что (в сравнении со многими специализированными учебниками по статистике) она доступна для широкой аудитории. Однако больше всего успех ОСО связан с тематикой этой книги. На момент публикации интерес к области статистического обучения начинал свой взрывообразный рост. ОСО стала одной из первых доступных и всеобъемлющих вводных работ по этой теме.

С момента публикации ОСО статистическое обучение продолжило свой расцвет. Развитие этой дисциплины приняло две формы. Наиболее заметный рост был связан с разработкой новых и усовершенствованных подходов статистического обучения, предназначенных для получения ответов на широкий круг вопросов в ряде научных областей. Однако статистическое обучение расширило также и свою аудиторию. В 1990–х г. рост доступности вычислительных ресурсов вызвал волну интереса к этой области со стороны неспециалистов по статистике, которым не терпелось начать использовать современные статистические инструменты для анализа своих данных. К сожалению, высокотехничная природа этих методов означала, что сообщество их пользователей оставалось ограниченным преимущественно экспертами по статистике, компьютерным и смежным областям, имеющими необходимую подготовку (и время) для освоения и реализации соответствующих методов.

В последние годы новое и усовершенствованное программное обеспечение значительно облегчило практическое применение многих методов статистического обучения. В то же время во многих областях, таких как бизнес, здравоохранение, генетика, социальные науки и т. д., произошло осознание того, что статистическое обучение является мощным инструментом для решения важных практических задач. Как следствие оно перестало быть чем–то, что представляет преимущественно академический интерес, и превратилось в популярную дисциплину с огромной потенциальной аудиторией. Несомненно, этот тренд продолжится по мере роста доступности огромных объемов данных и программного обеспечения, предназначенного для их анализа.

Цель книги «*Введение в статистическое обучение*» (ВСО) состоит в том, чтобы содействовать превращению статистического обучения из академической в популярную практическую дисциплину. ВСО не предназначена для замены ОСО, которая является гораздо более обстоятельной работой как по числу рассматриваемых в ней методов, так и по глубине их описания. Мы рассматриваем ОСО в качестве справочника для профессионалов (имеющих ученые степени по статистике, машинному обучению или сходным направлениям), которым необходимо понимать технические детали, лежащие в основе подходов статистического обучения. Однако сообщество пользователей методов машинного обучения расширилось и включает людей с более широким кругом интересов и с разным образованием. Поэтому мы убеждены, что сейчас появилось место для менее технической и более доступной версии ОСО.

В ходе преподавания этих тем на протяжении многих лет мы обнаружили, что они представляют интерес для магистрантов и аспирантов из настолько далеких друг от друга дисциплин, как бизнес–администрирование, биология и компьютерные науки, а также для ориентированных на количественные дисциплины студентов старших курсов. Для этой разнородной аудитории важно иметь возможность понимать модели, их предпосылки, а также сильные и слабые стороны различных методов. Однако многие технические детали методов статистического обучения, такие как алгоритмы оптимизации и теоретические свойства методов, для этой аудитории не представляют большого интереса. Мы убеждены, что таким студентам не нужно иметь глубокого понимания этих аспектов, для того чтобы начать осознанно применять различные методы и сделать

вклад в соответствующие научные дисциплины с помощью инструментария статистического обучения.

Книга ВСО основана на следующих четырех предпосылках.

1. *Многие методы статистического обучения применимы и полезны для широкого круга академических и практических дисциплин, выходящих далеко за рамки статистической науки.* Мы убеждены, что многие современные процедуры статистического обучения должны стать (и станут) настолько же широко доступными и используемыми, как классические методы наподобие линейной регрессии. В связи с этим вместо попытки охватить все возможные подходы (а это невыполнимая задача) мы сконцентрировались на представлении методов, которые считаем наиболее широко применимыми.

2. *Статистическое обучение не следует рассматривать как набор «черных ящиков».* Не существует метода, который одинаково хорошо сработает во всех возможных ситуациях. Без понимания всех «винтиков» внутри «ящика»» и без взаимодействия с этими «винтиками» невозможно выбрать наилучший «ящик». Поэтому мы предприняли попытку тщательно описать модель, идею, допущения и компромиссы, лежащие в основе каждого рассматриваемого нами метода.

3. *Несмотря на важность понимания функции, выполняемой каждым «винтиком», нет необходимости уметь конструировать саму машину, находящуюся внутри «ящика».* Поэтому мы минимизировали обсуждение технических деталей, имеющих отношение к процедурам подгонки моделей и теоретическим свойствам методов. Мы предполагаем, что читатель чувствует себя комфортно с простейшими математическими концепциями, но мы не ожидаем от него ученой степени в области математических наук. Например, мы почти полностью исключили использование матричной алгебры, и всю книгу можно понять без знания матриц и векторов.

4. *Мы предполагаем, что читатель интересуется применением методов статистического обучения для решения практических проблем.* Чтобы удовлетворить этот интерес и мотивировать к применению обсуждаемых методов, после каждой главы мы приводим раздел с лабораторными работами. В каждой лабораторной работе мы знакомим читателя с реалистичным практическим приложением методов, рассмотренных в соответствующей главе. Когда мы преподавали этот материал в наших курсах, мы отводили на лабораторные работы примерно треть всего времени и нашли их чрезвычайно полезными. Многие студенты, которые поначалу испытывали затруднения при работе с командным интерфейсом R, усвоили необходимые навыки в течение семестра. Мы использовали R потому, что эта система является бесплатной и достаточно мощной для реализации всех рассмотренных в книге методов. Кроме того, она имеет расширения, которые можно загрузить для реализации буквально тысяч дополнительных методов. Но важнее всего то, что R предпочитают академические статистики, и новые методы часто становятся доступными в R на несколько лет раньше того, как они появляются в платных

программах. Тем не менее лабораторные работы в ВСО автономны, и их можно пропускать, если читатель желает использовать другое программное обеспечение или не намерен применять обсуждаемые методы к реальным проблемам.

Кому следует прочесть эту книгу?

Эта книга предназначена для всех, кто интересуется применением современных статистических методов для моделирования и прогнозирования на основе данных. Эта группа читателей включает ученых, инженеров, финансовых аналитиков, а также людей с меньшей технической и математической подготовкой, которые имеют образование в таких областях, как социальные науки или бизнес. Мы ожидаем, что читатель прослушал как минимум один вводный курс по статистике. Знание линейной регрессии также полезно, но не обязательно, поскольку в главе 3 мы даем обзор ключевых концепций, лежащих в основе этого метода. Уровень математики в этой книге умеренный, и детальное знание матричных операций не требуется. Книга содержит введение в язык статистического программирования `R`. Предыдущий опыт программирования на другом языке, вроде `MATLAB` или `Python`, полезен, но не обязателен.

Мы успешно преподавали материал на этом уровне магистрантам и аспирантам, изучающим бизнес, компьютерные науки, биологию, науки о Земле, психологию и многие другие направления естественных и гуманитарных наук. Эта книга также могла бы оказаться подходящей для студентов последних курсов, которые уже прослушали курс по линейной регрессии. В контексте математически более строгого курса, где основным учебником является ОСО, ВСО можно было бы использовать в качестве дополнительного источника для преподавания вычислительных аспектов различных методов.

Обозначения и простая матричная алгебра

Выбор системы обозначений для учебника — это всегда сложная задача. В большинстве случаев мы применяем те же условные обозначения, что и в ОСО.

Мы будем использовать n для обозначения числа отдельных значений, или наблюдений, в нашей выборке. При помощи p мы будем обозначать число имеющихся переменных, на основе которых можно делать предсказания. Например, набор данных `Wage` состоит из 12 переменных для 3000 людей, так что у нас есть $n = 3000$ наблюдений и $p = 12$ переменных, таких как `year`, `age`, `wage` и др. Заметьте, что на протяжении всей этой книги для обозначения имен переменных мы используем цветной шрифт: `Имя Переменной`.

В некоторых примерах p может быть довольно большим, порядка нескольких тысяч или даже миллионов; подобная ситуация достаточно часто возникает, например, при анализе современных биологических данных или данных по интернет–рекламе.

Обычно при помощи x_{ij} мы будем обозначать i–е значение j–й переменной, где $i = 1, 2, \ldots, n$, а $j = 1, 2, \ldots, p$. На протяжении этой книги

i будет использоваться для индексирования выборок или отдельных наблюдений (от 1 до n), а j — для индексирования переменных (от 1 до p). С помощью $\mathbf{X}$ мы обозначаем матрицу размером $n \times p$, чей (i, j)–й элемент — это x_{ij}. Другими словами,

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}.$$

Читателям, не знакомым с матрицами, полезно будет мысленно представлять $\mathbf{X}$ в виде таблицы чисел с n строками и p столбцами.

В ряде случаев нам будут интересны строки матрицы $\mathbf{X}$, которые мы записываем как $x_1, x_2, \ldots, x_n$. Здесь x_i представляет собой вектор длиной p, содержащий значения p переменных для i-го наблюдения. Другими словами,

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}. \tag{1.1}$$

(По определению, векторы представлены в виде столбцов.) Например, для набора данных `Wage` x_i — это вектор длиной 12, состоящий из значений `year`, `age`, `wage` и других переменных для i–го человека. В других случаях вместо строк нам будут интересны столбцы матрицы $\mathbf{X}$, которые мы записываем как $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$. Каждый из этих столбцов является вектором длиной n, т. е.

$$\mathbf{x}_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix}.$$

Например, в случае с данными `Wage` $\mathbf{x}_1$ содержит $n = 3000$ значений `year`.

Используя эту нотацию, матрицу $\mathbf{X}$ можно записать как

$$\mathbf{X} = (\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_p),$$

или

$$\mathbf{X} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix}.$$

Символ ^{T} обозначает *транспозицию* матрицы или вектора. Так, например,

$$\mathbf{X}^T = \begin{pmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{np} \end{pmatrix},$$

тогда как

$$x_i^T = (x_{i1} \quad x_{i2} \quad \cdots \quad x_{ip}).$$

Мы используем y_i для обозначения i–го наблюдения переменной, которую мы хотим предсказать (например, `wage`). Следовательно, в векторной форме мы записываем набор всех n наблюдений как

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Тогда наши наблюдаемые данные состоят из пар $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, где каждый элемент x_i — это вектор длиной p. Если $p = 1$, то x_i является просто скаляром.

В этой книге вектор длиной n всегда будет обозначаться при помощи *прописной буквы, выделенной жирным шрифтом*, т. е.

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}.$$

Однако векторы, чья длина отличается от n (например, векторы признаков длиной p, как в (1.1)), будут обозначаться при помощи прописных букв, выполненных обычным шрифтом (например, a). Скаляры также будут обозначаться при помощи таких *прописных букв*, т. е. a. В редких случаях, когда использование прописных букв с обычным шрифтом может привести к двусмысленности, мы будем пояснять, что имеется в виду. Матрицы будут обозначаться с использованием *заглавных букв, выполненных жирным шрифтом* (например, $\mathbf{A}$). Случайные переменные будут обозначаться *заглавными буквами, выполненными обычным шрифтом* (например, A), вне зависимости от их размерности.

Иногда у нас будет возникать необходимость указать размерность конкретного объекта. Чтобы показать, что объект является вектором, мы будем использовать нотацию $a \in \mathbb{R}$. Чтобы показать, что это вектор длиной k, мы будем использовать обозначение $a \in \mathbb{R}^k$ (или $a \in \mathbb{R}^n$, если он имеет длину n). Объекты, которые являются матрицами размером $r \times s$, мы будем обозначать как $\mathbf{A} \in \mathbb{R}^{r \times s}$.

Мы избегали использования матричной алгебры везде, где это было возможно. Однако в нескольких случаях полностью избегать ее становилось слишком обременительно. В этих редких случаях важно понимать концепцию умножения двух матриц. Предположим, что $\mathbf{A} \in \mathbb{R}^{r \times d}$,

а $\mathbf{B} \in \mathbb{R}^{d \times s}$. Результат умножения $\mathbf{A}$ на $\mathbf{B}$ обозначается как $\mathbf{AB}$. Элемент (i, j) матрицы $\mathbf{AB}$ вычисляется путем умножения каждой i–й строки $\mathbf{A}$ на соответствующий элемент j–го столбца $\mathbf{B}$. Иначе говоря, $(\mathbf{AB})_{ij} = \sum_{k=1}^{d} a_{ik} b_{kj}$. В качестве примера представьте, что

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad \text{и} \quad \mathbf{B} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}.$$

Тогда

$$\mathbf{AB} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 1 \times 5 + 2 \times 7 & 1 \times 6 + 2 \times 8 \\ 3 \times 5 + 4 \times 7 & 3 \times 6 + 4 \times 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}.$$

Заметьте, что результатом этой операции является матрица размером $r \times s$. $\mathbf{AB}$ можно вычислить только, если число столбцов в $\mathbf{A}$ равно числу строк в $\mathbf{B}$.

Структура этой книги

Глава 2 вводит основные термины и концепции, лежащие в основе статистического обучения. Эта глава описывает также классификатор на основе *K ближайших соседей* — очень простой метод, который удивительно хорошо работает для многих проблем. Главы 3 и 4 охватывают классические линейные методы регрессии и классификации. В частности, глава 3 дает обзор *линейной регрессии* — фундаментальной отправной точки всех регрессионных методов. В главе 4 мы обсуждаем два наиболее важных классических метода классификации — *логистическую регрессию* и *линейный дискриминантный анализ*.

Центральной проблемой всех случаев использования статистического обучения является выбор наилучшего метода для решения конкретной задачи. Поэтому в главе 5 мы знакомим читателя с *перекрестной проверкой* и *бутстрепом*, которые можно использовать для оценивания точности нескольких методов и выбора самого подходящего из них.

Многие недавние исследования в области статистического обучения были посвящены нелинейным методам. Однако линейные методы часто имеют преимущества по сравнению со своими нелинейными конкурентами в смысле интерпретируемости, а иногда и точности предсказаний. Поэтому в главе 6 мы рассматриваем целый ряд линейных методов, как классических, так и более современных, которые предлагают потенциальные улучшения по сравнению со стандартной линейной регрессией. Речь идет о *пошаговом отборе*, *гребневой регрессии*, *регрессии на главные компоненты*, методе *частных наименьших квадратов* и *лассо–регрессии*.

Остальные главы посвящены миру нелинейного статистического обучения. Сначала в главе 7 мы вводим несколько нелинейных методов, которые хорошо работают для проблем с одной входной переменной. Затем мы показываем, как эти методы можно использовать для построения нелинейных *аддитивных* моделей с несколькими входными переменными. В главе 8 мы исследуем методы, основанные на *решающих деревьях*, включая *бэггинг*, *бустинг* и *случайные леса*. *Метод опорных векторов*

— совокупность подходов для решения задач как линейной, так и нелинейной классификации — обсуждается в главе 9. Наконец, в главе 10 мы рассматриваем ситуацию, когда у нас имеются входные переменные, но нет выходной переменной. В частности, мы описываем *анализ главных компонент*, кластеризацию с помощью *метода K средних*, а также *иерархическую кластеризацию*.

В конце каждой главы мы приводим одну или несколько лабораторных работ с использованием R, в которых подробно разбираем примеры практического применения рассмотренных в этой главе методов. Эти лабораторные работы демонстрируют сильные и слабые стороны разных подходов, а также предоставляют полезный справочный материал по синтаксису, необходимому для реализации разных методов. Возможно, читатель предпочтет работать над этими лабораторными в своем собственном темпе, но это могут быть также и групповые сессии, являющиеся частью классной работы. В каждой лабораторной работе мы представляем результаты, которые получили на момент написания книги. Однако постоянно выходят новые версии R, и со временем используемые в лабораторных работах пакеты будут обновлены. Поэтому возможно, что в будущем представленные в соответствующих разделах результаты больше не будут в точности соответствовать результатам, полученным читателем. По мере необходимости мы будем обновлять лабораторные работы на сайте книги.

Мы используем значок  для обозначения разделов и упражнений повышенной сложности. Эти разделы могут быть легко пропущены читателями, которые не желают так глубоко погружаться в соответствующий материал или не имеют необходимой математической подготовки.

Данные, использованные в лабораторных работах и упражнениях

В этом учебнике мы иллюстрируем методы статистического обучения на практических примерах из маркетинга, финансов, биологии и других областей. Пакет ISLR, доступный на сайте книги, содержит несколько наборов данных, которые необходимы для выполнения соответствующих лабораторных работ и упражнений. Один из таких наборов данных входит в состав библиотеки MASS, а еще один является частью базового дистрибутива R. Таблица 1.1 содержит сводную информацию по данным, необходимым для выполнения лабораторных работ и упражнений. Несколько таких наборов данных, используемых в главе 2, доступно также в виде текстовых файлов на сайте книги.

Веб–сайт книги

Веб–сайт этой книги находится по адресу www.StatLearning.com. Он содержит целый ряд ресурсов, включая связанный с книгой R–пакет и некоторые дополнительные наборы данных.

ТАБЛИЦА 1.1. *Список наборов данных, необходимых для выполнения лабораторных работ и упражнений в этом учебнике. Все данные доступны в пакете* `ISLR`*, за исключением* `Boston` *и* `USArrests` *(они входят в состав базового дистрибутива* `R`*)*

Название	Описание
`Auto`	Расход топлива, мощность двигателя и другая информация по автомобилям
`Boston`	Стоимость жилья и другая информация по пригородам Бостона
`Caravan`	Информация по людям, которым была предложена страховка для их жилых прицепов
`Carseats`	Информация по продажам автомобильных сидений в 400 магазинах
`College`	Демографические характеристики, стоимость обучения и другие данные по колледжам США
`Default`	Данные по должникам компании, выпускающей кредитные карты
`Hitters`	Данные по заработной плате бейсбольных игроков
`Khan`	Измерения генной экспрессии для четырех типов рака
`NCI60`	Измерения генной экспрессии для 64 раковых клеточных линий
`OJ`	Информация по продажам апельсинового сока марок Citrus Hill и Minute Made
`Portfolio`	Исторические значения финансовых активов, используемые для распределения инвестиционных средств
`Smarket`	Дневные удельные изменения доходности индекса S&P за 5–летний период
`USArrests`	Криминальная статистика в расчете на 100 000 жителей в 50 штатах США
`Wage`	Данные исследования доходов мужчин в центрально–атлантическом регионе США
`Weekly`	1098 значений недельной доходности рынка акций за 21 год

Благодарности

Несколько графиков в этой книге было заимствовано из OCO: рис. 6.7, 8.3 и 10.12. Все остальные графики в книге новые.

Глава 2

Статистическое обучение

2.1 Что такое статистическое обучение?

Рассмотрим простой пример, который поможет нам приступить к изучению статистического обучения. Представьте себе, что мы являемся консультантами–статистиками, нанятыми некоторым клиентом для разработки рекомендаций по повышению продаж определенного продукта. Набор данных `Advertising` включает сведения по продажам (`sales`) этого продукта в 200 различных регионах, а также по величине регионального бюджета на рекламу продукта в средствах массовой информации (СМИ) трех видов: телевидение (`TV`), радио (`radio`) и газеты (`newspaper`). Эти данные представлены на рис. 2.1. Наш клиент не имеет прямой возможности увеличить продажи. С другой стороны, он может контролировать затраты на рекламу в каждом из трех типов СМИ. Следовательно, если мы обнаружим связь между затратами на рекламу и продажами, то сможем дать рекомендации нашему клиенту по корректированию бюджета на рекламу, что опосредованно приведет к увеличению продаж. Другими словами, наша цель состоит в разработке модели, которую можно будет использовать для верных предсказаний продаж на основе данных по бюджету для трех типов СМИ.

входная и выходная переменные

При таком сценарии уровни бюджета на рекламу в разных СМИ являются *входными переменными*, тогда как `sales` — *выходной переменной*. Входные переменные обычно обозначаются при помощи символа X с нижним индексом, позволяющим различать отдельные переменные. Так, X_1 может обозначать бюджет `TV`, X_2 — бюджет `radio`, а X_3 — бюджет `newspaper`. Входные переменные известны под разными названиями — *предикторы*, *независимые переменные*, *признаки*, или иногда просто *переменные*. Выходную переменную — в данном случае `sales` — часто называют *откликом*, или *зависимой переменной*, и обычно обозначают при помощи символа Y. В этой книге мы будем использовать указанные термины попеременно.

независимая и зависимая переменные

предиктор

признак

отклик

В качестве более общего случая представьте, что мы наблюдаем некоторый количественный отклик Y и p отдельных предикторов $X_1, X_2, \ldots, X_p$. Мы делаем предположение о том, что существует определенная связь между Y и $X = (X_1, X_2, \ldots, X_p)$, которую в очень общей форме можно записать как

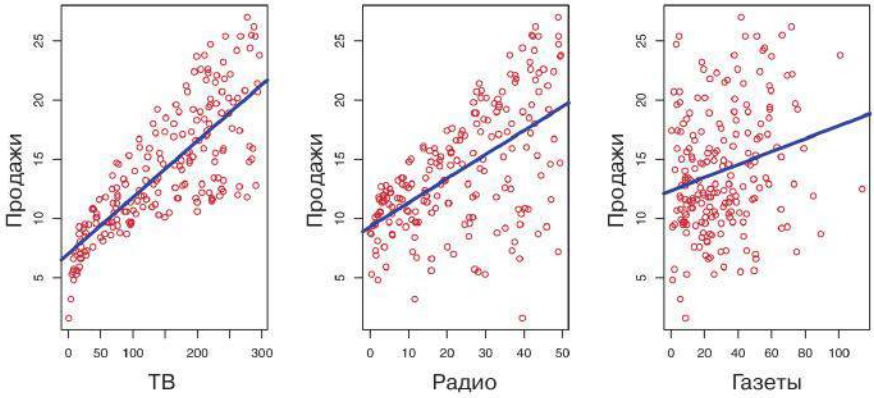

РИСУНОК 2.1. *Набор данных* `Advertising`. *На рисунке показан объем продаж* (`sales`, *тыс. единиц) в зависимости от величины бюджета на рекламу (тыс. долларов) на телевидении* (`TV`), *радио* (`radio`) *и в газетах* (`newspaper`) *в 200 регионах*

$$Y = f(X) + \epsilon. \tag{2.1}$$

ошибка

Здесь f — это некоторая фиксированная, но не известная функция от $X_1, \ldots, X_p$, а ϵ — *ошибка*, которая не зависит от X и имеет нулевое среднее значение. В таком представлении f выражает *систематическую* информацию о Y, содержащуюся в X.

В качестве другого примера рассмотрим данные для 30 человек из таблицы `Income`, отражающие зависимость дохода (`income`) от количества лет, затраченных этими людьми на образование (`years`; см. рис. 2.2 слева). Этот график предполагает, что мы могли бы предсказать доход на основе длительности обучения. Однако функция, связывающая входную переменную с выходной переменной, в общем случае неизвестна. В такой ситуации мы должны оценить f на основе имеющихся наблюдений. Поскольку таблица `Income` содержит имитированные данные, то f известна и показана на рис. 2.2 справа в виде кривой голубого цвета. Вертикальные отрезки соответствуют ошибкам ϵ. Заметьте, что некоторые из 30 наблюдений лежат выше голубой линии, а некоторые — ниже, но в целом среднее значение ошибок примерно равно нулю.

В общем случае функция f может включать более одной входной переменной. На рис. 2.3 мы изображаем `income` как функцию от длительности обучения и стажа (`seniority`). Здесь f представляет собой двумерную плоскость, которую мы должны оценить на основе имеющихся данных.

В сущности, под статистическим обучением понимают совокупность методов для оценивания f. В этой главе мы рассматриваем некоторые из ключевых теоретических концепций, применяемых при нахождении f, а также инструменты для определения качества полученных оценок.

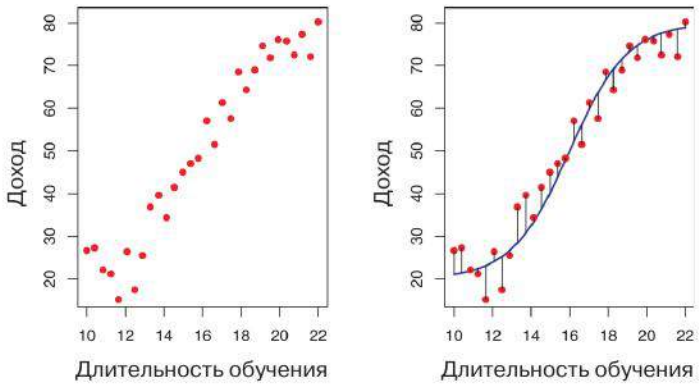

РИСУНОК 2.2. *Набор данных* `Income`. Слева: *красные точки соответствуют значениям переменных* `income` *(доход, десятки тыс. долларов) и* `years` *(длительность обучения, лет) у 30 человек.* Справа: *голубая кривая соответствует истинной функции связи между* `income` *и* `years`*, которая обычно неизвестна (в этом случае она известна, поскольку данные были имитированы). Черные вертикальные линии показывают ошибки соответствующих наблюдений. Заметьте, что некоторые ошибки положительны (когда наблюдение лежит выше голубой кривой), а некоторые — отрицательны (когда наблюдение находится ниже кривой). В целом среднее значение ошибок примерно равно нулю*

2.1.1 Зачем оценивать f?

Существуют две основные причины, по которым мы хотели бы оценить f: *предсказание* и *статистический вывод*. Обсудим каждую из этих причин по порядку.

Предсказание

Во многих ситуациях набор входных переменных X легко доступен, однако получить выходную переменную Y не так просто. Благодаря тому, что ошибки имеют нулевое среднее значение, при таком сценарии мы можем предсказать Y с помощью

$$\hat{Y} = \hat{f}(X), \tag{2.2}$$

где $\hat{f}$ представляет собой нашу оценку f, а $\hat{Y}$ — предсказанное значение Y. В подобной ситуации $\hat{f}$ часто рассматривают как *черный ящик*, в том смысле, что если $\hat{f}$ обеспечивает верные предсказания Y, то точная форма этой функции для нас не важна.

В качестве примера предположим, что $X_1, \ldots, X_p$ являются характеристиками образца крови пациента, которые легко измерить в лаборатории, а Y — это переменная, отражающая риск того, что пациент проявит резко негативную реакцию на определенный лекарственный препарат. Естественным будет желание предсказать Y по X, поскольку таким образом

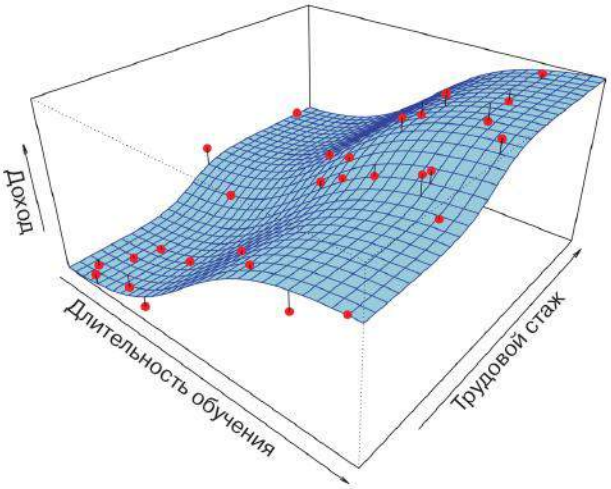

РИСУНОК 2.3. *График показывает* `income` *как функцию от* `years` *и* `seniority` *(переменные из набора данных* `Income`*). Голубая плоскость отражает истинную зависимость* `income` *от* `years` *и* `seniority`*, которая известна, поскольку эти данные были имитированы. Красные точки показывают наблюденные значения для 30 человек*

мы сможем избежать назначения данного препарата пациентам с высоким риском негативной реакции, т. е. пациентам с высокими значениями оценок Y.

устранимая и неустранимая ошибки

Точность $\hat{Y}$ в качестве предсказанного значения Y зависит от двух величин, которые мы будем называть *устранимой ошибкой* и *неустранимой ошибкой*. Обычно $\hat{f}$ не будет идеальной оценкой f, и эта неточность приведет к возникновению некоторой ошибки. Такая ошибка является *устранимой*, поскольку потенциально мы можем улучшить точность $\hat{f}$, используя более подходящий статистический метод для оценивания f. Но даже если бы имелась возможность достичь настолько идеальной оценки f, что $\hat{Y} = f(X)$, наше предсказанное значение все равно содержало бы в себе некоторую ошибку! Подобная ошибка известна как *неустранимая*, поскольку как бы хорошо мы ни оценили f, мы не сможем снизить ошибку, привнесенную за счет ϵ.

Почему же неустранимая ошибка превышает нулевое значение? Величина ϵ может включать неучтенные переменные, которые полезны для предсказания Y, но поскольку мы их не измеряем, то f не может использовать их для предсказания. Например, риск негативной реакции может варьировать у того или иного пациента в определенный день в зависимости от условий производства самого лекарственного препарата или от общего состояния здоровья пациента в тот день.

Представьте себе некоторую конкретную оценку $\hat{f}$ и набор предикторов X, которые дают предсказание $\hat{Y} = \hat{f}(X)$. Допустите на мгновение, что и $\hat{f}$, и X являются фиксированными величинами. Тогда можно легко показать, что

$$E(Y-\hat{Y})^2 = E[f(X)+\epsilon-\hat{f}(X)]^2 = \underbrace{[f(X)-\hat{f}(X)]^2}_{\text{устранимая}} + \underbrace{\text{Var}(\epsilon)}_{\text{неустранимая}}, \quad (2.3)$$

где $E(Y-\hat{Y})^2$ представляет собой среднее, или *ожидаемое*, значение квадрата разности между предсказанным и истинным значением Y, а $\text{Var}(\epsilon)$ — *дисперсию*, связанную с ошибкой ϵ.

ожидаемое значение

дисперсия

В этой книге сделан упор на методы, предназначенные для оценивания f с целью минимизации устранимой ошибки. Важно помнить, что неустранимая ошибка всегда будет обеспечивать верхнюю границу точности нашего предсказания Y. На практике эта граница почти всегда неизвестна.

Статистический вывод

Часто мы заинтересованы в понимании того, как изменение $X_1, \ldots, X_p$ влияет на Y. В такой ситуации мы хотим оценить f, но наша цель не обязательно заключается в получении предсказаний для Y. Вместо этого мы хотим понять взаимоотношение между X и Y или, более конкретно, понять функциональную связь между Y и $X_1, \ldots, X_p$. В этом случае $\hat{f}$ нельзя рассматривать в качестве черного ящика, поскольку нам нужно знать ее точную форму. При таком сценарии мы можем быть заинтересованы в ответе на следующие вопросы:

- *Какие предикторы связаны с откликом?* Часто только небольшая часть имеющихся в распоряжении предикторов тесно связана с Y. В зависимости от стоящей задачи нахождение ограниченного числа *важных* предикторов в большом наборе возможных переменных может оказаться чрезвычайно полезным.

- *Какова связь между откликом и каждым предиктором?* Некоторые предикторы могут иметь положительную связь с Y в том смысле, что увеличение предиктора вызывает возрастание значений Y. Другие предикторы могут оказывать противоположный эффект. В зависимости от сложности f связь между откликом и некоторым предиктором может зависеть также от значений других предикторов.

- *Можно ли связь между Y и каждым предиктором адекватно обобщить в виде линейного уравнения, или эта связь является более сложной?* Исторически сложилось так, что большинство методов для оценивания f подразумевали линейную форму. В некоторых ситуациях такое допущение является обоснованным или даже желательным. Однако часто истинная связь является более сложной и линейная модель не способна обеспечить адекватное представление зависимости между входными и выходными переменными.

В этой книге мы увидим несколько примеров, относящихся к ситуациям, когда требуется выполнение предсказаний, получение статистических выводов или комбинация обеих этих задач.

Например, представьте себе фирму, которая заинтересована в проведении персонализированной маркетинговой кампании. Цель заключается в использовании имеющихся в распоряжении фирмы демографических данных для нахождения людей, которые положительно ответят на высланное по почте предложение. В этом случае демографические переменные служат в качестве предикторов, а отклик на маркетинговую кампанию (положительный или отрицательный) является выходной переменной. Фирма не ставит задачей сформировать глубокое понимание взаимоотношений между каждым отдельным предиктором и откликом — вместо этого ей просто нужна точная модель для предсказания отклика на основе предикторов. Это пример построения модели для получения предсказаний.

С другой стороны, рассмотрим данные из таблицы `Advertising`, изображенные на рис. 2.1. Интерес могут представлять ответы на следующие вопросы:

- *Какие СМИ способствуют продажам?*
- *Какие СМИ вызывают наибольший всплеск продаж?*
- *Насколько тесно рост продаж связан с тем или иным увеличением затрат на телерекламу?*

Такая ситуация подпадает под парадигму статистических выводов. Другой пример включает моделирование бренда продукта, который клиент мог бы купить, исходя из таких переменных, как цена, местоположение магазина, уровни скидок, цена у конкурентов и т. д. В этой ситуации нас больше всего могло бы интересовать то, как каждая отдельная переменная влияет на вероятность покупки. Например, *какое влияние на продажи оказывает изменение цены?* Это пример моделирования с целью сделать статистический вывод.

Наконец, иногда моделирование выполняют для получения как предсказаний, так и статистических выводов. Например, в случае с недвижимым имуществом мы могли бы попытаться увязать значения стоимости домов с такими входными переменными, как уровень преступности, район, расстояние от реки, качество воздуха, доход соседей, размер домов и т. д. В этом случае нам может быть интересно то, как отдельные входные переменные влияют на цены: например, *насколько дороже будет дом с видом на реку?* Это проблема по получению статистического вывода. С другой стороны, нас могло бы интересовать просто предсказание стоимости дома на основе его характеристик: *недо– или переоценен этот дом?* Это проблема предсказания.

В зависимости от того, какова наша главная цель — предсказание, статистический вывод, или комбинация этих двух проблем, — для оценивания f подходящими могут оказаться разные методы. Например, *линейные модели* позволяют делать относительно простые и интерпретируемые статистические выводы, но в сравнении с другими подходами они могут давать менее точные предсказания. С другой стороны, некоторые из высоконелинейных подходов, обсуждаемых нами в последних главах этой книги, потенциально могут обеспечить очень точные предсказания для Y, но за счет менее интерпретируемой модели, по которой статистические выводы сделать сложнее.

линейные модели

2.1.2 Как мы оцениваем f?

В этой книге мы рассматриваем многие линейные и нелинейные подходы для оценивания f. Тем не менее часто эти методы имеют определенные общие характеристики. Мы приводим обзор этих общих черт в данном разделе. Во всех случаях предполагается, что у нас имеется набор из n отдельных наблюдений. Например, на рис. 2.2 у нас есть $n = 30$ наблюдений. Эти наблюдения называются *обучающими данными*[1], поскольку мы используем их для тренировки, или обучения, нашего метода тому, как оценить f. Пусть x_{ij} символизирует значение j–го предиктора, или входной переменной, у наблюдения i, где $i = 1, 2, \ldots, n$, а $j = 1, 2, \ldots, p$. Аналогичным образом пусть y_i обозначает отклик у i–го наблюдения. Обучающие данные тогда состоят из пар $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, где $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$.

обучающие данные

Наша цель заключается в применении некоторого метода статистического обучения к входным данным для нахождения неизвестной функции f. Другими словами, мы хотим найти такую функцию $\hat{f}$, что $Y \approx \hat{f}(X)$ для любого наблюдения (X, Y). В общих чертах большинство методов статистического обучения для решения этой задачи можно разделить на *параметрические* и *непараметрические*.

параметрические и непараметрические методы

Параметрические методы

Параметрические методы подразумевают основанную на модели процедуру из двух шагов.

1. Во–первых, мы делаем некоторое предположение о функциональной форме f. Например, одно из простых предположений заключается в том, что f является линейной функцией от X:

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p. \tag{2.4}$$

   Это *линейная модель*, которая будет детально обсуждаться в главе 3. Как только мы допустили, что f имеет линейную форму, проблема оценивания f значительно упрощается. Вместо нахождения совершенно произвольной p–мерной функции $f(X)$ нам нужно будет оценить лишь $p + 1$ коэффициентов $\beta_0, \beta_1, \ldots, \beta_p$.

2. После выбора модели нам потребуется процедура, которая использует обучающие данные для *подгонки*, или *обучения*, модели. В случае линейной модели (2.4) нам необходимо оценить параметры $\beta_0, \beta_1, \ldots, \beta_p$. Другими словами, мы хотим найти такие значения этих параметров, при которых

   подгонка
   обучение

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

   Наиболее распространенный метод для подгонки модели (2.4) известен как *метод наименьших квадратов*, и мы обсудим его в главе 3. Однако метод наименьших квадратов является лишь одним из многих возможных способов. В главе 6 мы рассмотрим другие подходы для нахождения параметров уравнения (2.4).

   метод наименьших квадратов

[1] Синонимами этого термина являются «обучающая выборка» и «обучающее множество», которые также будут использоваться в дальнейшем. — *Прим. пер.*

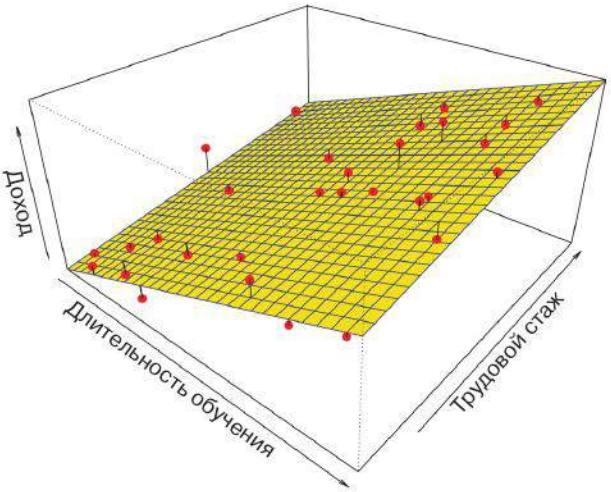

РИСУНОК 2.4. *Линейная модель, подогнанная по методу наименьших квадратов к данным* `Income` *(см. рис. 2.3). Выборочные значения представлены точками красного цвета, а желтая плоскость показывает подогнанную к данным модель*

Описанный выше подход, основанный на модели, называется *параметрическим*; он сводит проблему оценивания f к проблеме оценивания некоторого набора параметров. Допущение о параметрической форме f упрощает проблему нахождения этой функции, поскольку, как правило, оценить набор параметров $\beta_0, \beta_1, \ldots, \beta_p$ линейной модели (2.4) гораздо проще, чем подогнать совершенно произвольную функцию f. Потенциальный недостаток параметрического подхода заключается в том, что выбираемая нами модель обычно не будет совпадать с истинной неизвестной формой f. Если выбранная модель слишком отличается от истинной, то наша оценка f будет плохой. Мы можем попытаться решить эту проблему, выбрав *гибкие* модели, которые способны описать многие из возможных функциональных форм f. В целом, однако, подгонка более гибкой модели требует оценивания большего числа параметров. Такие более сложные модели могут приводить к явлению, известному как *переобучение*, которое, в сущности, означает то, что эти модели начинают слишком близко аппроксимировать ошибками, или *шум*, в данных. Эти проблемы обсуждаются на протяжении всей книги.

гибкие модели

переобучение

шум

Рисунок 2.4 демонстрирует пример применения параметрического метода к данным `Income` (см. рис. 2.3). Мы подогнали линейную модель вида

$$\texttt{income} \approx \beta_0 + \beta_1 \times \texttt{education} + \beta_2 \times \texttt{seniority}.$$

Поскольку мы допустили наличие линейной связи между откликом и этими двумя предикторами, то вся проблема по подгонке модели сводится к оцениванию β_0, β_1 и β_2, которое мы выполняем с использованием линейной регрессии по методу наименьших квадратов. Сравнивая рис. 2.3 и 2.4, мы можем увидеть, что представленная на рис. 2.4 линейная модель не вполне верна: истинная функция f имеет некоторый изгиб, который

не отражен в этой линейной модели. Тем не менее линейная модель, похоже, хорошо справляется с задачей по описанию положительной зависимости между education и income, а также несколько менее выраженной положительной зависимости между seniority и income. Возможно, что при таком небольшом числе наблюдений это лучшее, что мы можем сделать.

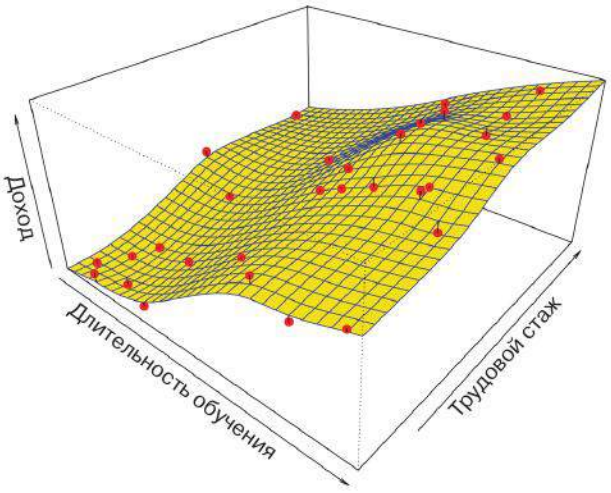

РИСУНОК 2.5. *Гладкий сплайн типа «тонкая пластина», подогнанный к данным* Income *(см. рис. 2.3), показан желтым цветом; выборочные наблюдения представлены точками красного цвета. Сплайны обсуждаются в главе 7*

Непараметрические методы

Непараметрические методы не делают явных предположений в отношении функциональной формы f. Вместо этого они выполняют поиск такой оценки f, которая приближается к данным максимально близко, не будучи при этом слишком грубой или слишком извилистой. Такие подходы могут иметь существенное преимущество по сравнению с параметрическим методами: избегая предположения о конкретной функциональной форме f, они способны точно описать более широкий ряд возможных форм f. Любой параметрический подход несет с собой возможность того, что используемая для оценивания f функциональная форма значительно отличается от истинной функции, вследствие чего итоговая модель будет плохо описывать данные. В то же время непараметрические методы полностью уходят от этой опасности, поскольку, по сути, не делается никакого предположения о форме f. Однако непараметрические методы страдают существенным недостатком: поскольку они не сводят проблему оценивания f к ограниченному набору параметров, то для получения точной оценки f требуется очень большое число наблюдений (намного больше, чем обычно необходимо для какого–либо параметрического метода).

Пример применения непараметрического метода для описания данных Income показан на рис. 2.5. Для оценивания f использован *сплайн типа*

сплайн типа «тонкая пластина»

«тонкая пластина». Этот метод не предписывает f какую–либо предварительно заданную модель. Вместо этого он пытается получить такую оценку f, которая максимально близко находилась бы к наблюдаемым данным, при условии что эта аппроксимация является *гладкой*. В рассматриваемом случае непараметрический метод дал удивительно точную оценку истинной функции f, показанной на рис. 2.3. Для подгонки сплайна типа «тонкая пластина» аналитик должен выбрать степень сглаживания. Рисунок 2.6 изображает результат подгонки такого же сплайна с меньшим уровнем сглаживания, который позволяет получить более неровную плоскость. Полученная функция в точности соответствует данным! Однако сплайн, показанный на рис. 2.6, намного более изменчив в сравнении с истинной функцией f из рис. 2.3. Это — пример переобучения, о котором мы говорили ранее. Такая ситуация нежелательна, поскольку полученная модель не будет давать точных предсказаний по новым наблюдениям, не входившим в состав исходного набора данных. Мы обсуждаем методы подбора *корректной* степени сглаживания в главе 5. Сплайны обсуждаются в главе 7.

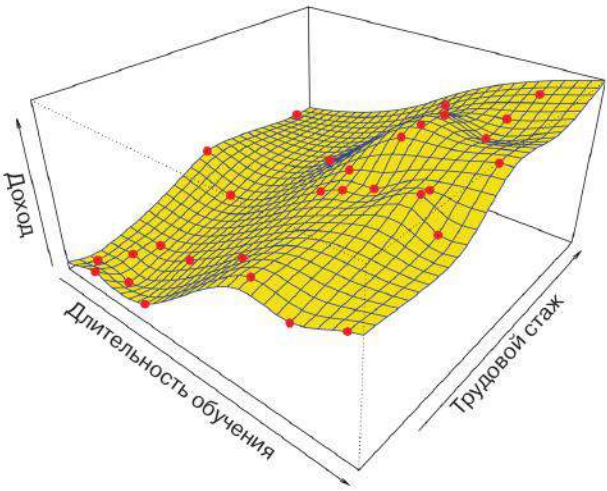

РИСУНОК 2.6. *Грубый сплайн типа «тонкая пластина», подогнанный к данным* `Income` *(рис. 2.3). Эта модель не делает ошибок на обучающих данных*

Как мы выяснили, параметрические и непараметрические методы имеют свои преимущества и недостатки. На протяжении этой книги мы рассматриваем оба подхода.

2.1.3 Компромисс между точностью предсказаний и интерпретируемостью модели

Некоторые из методов, рассматриваемых нами в этой книге, являются менее гибкими, или более ограниченными, в том смысле, что они способны породить лишь относительно небольшой набор функциональных форм

для оценивания f. Например, линейная регрессия является относительно негибким подходом, поскольку она может порождать только линейные функции вроде линий на рис. 2.1 или плоскости на рис. 2.4. Другие методы, такие как сплайны типа «тонкая пластина», приведенные на рис. 2.5 и 2.6, являются намного более гибкими, так как они могут порождать гораздо более широкий спектр возможных форм для оценивания f.

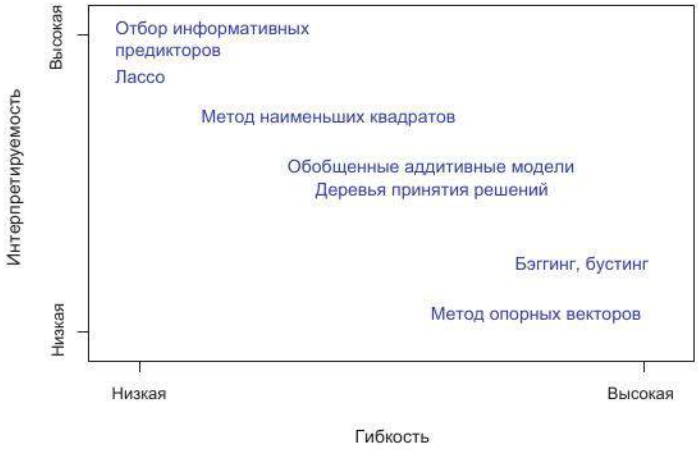

РИСУНОК 2.7. *Один из способов представить компромисс между гибкостью и интерпретируемостью разных методов статистического обучения. В целом при возрастании гибкости метода его интерпретируемость снижается*

Резонно мог бы возникнуть следующий вопрос: *зачем нам вообще выбирать более ограниченный метод вместо очень гибкого метода*? Имеется несколько причин предпочесть более ограниченную модель. Если мы преимущественно заинтересованы в статистических выводах, то ограниченные модели являются гораздо более интерпретируемыми. Например, когда целью является статистический вывод, линейная модель может быть хорошим выбором, поскольку довольно легко будет понять взаимоотношения между Y и $X_1, X_2, \ldots, X_p$. В то же время очень гибкие методы, такие как сплайны, обсуждаемые в главе 7 и показанные на рис. 2.5 и 2.6, а также методы бустинга, обсуждаемые в главе 8, могут приводить к настолько сложным оценкам f, что будет трудно понять, как любой отдельно взятый предиктор связан с откликом.

Рисунок 2.7 иллюстрирует компромисс между гибкостью и интерпретируемостью некоторых методов, рассматриваемых нами в этой книге. Обсуждаемая в главе 3 линейная регрессия по методу наименьших квадратов является относительно негибкой, но при этом вполне интерпретируемой. Метод *лассо*, обсуждаемый в главе 6, основан на линейной модели (2.4), но использует альтернативную процедуру для оценивания коэффициентов $\beta_1, \beta_2, \ldots, \beta_p$. Эта новая процедура накладывает более жесткие ограничения при нахождении оценок коэффициентов и приравнивает некоторые из них в точности к нулю. Следовательно, в этом смысле лассо является менее гибким подходом, чем линейная регрессия. Кроме того, этот ме-

лассо

тод легче поддается интерпретации по сравнению с линейной регрессией, поскольку в конечной модели переменная–отклик будет зависеть только от небольшого подмножества предикторов, а именно тех, чьи оценки коэффициентов оказались отличными от нуля. В то же время *обобщенные аддитивные модели* (GAM)[2], обсуждаемые в главе 7, расширяют (2.4), позволяя моделировать определенные нелинейные зависимости. Как следствие GAM являются более гибкими, чем линейная регрессия. Они также несколько менее интерпретируемы, чем линейная регрессия, поскольку связь между каждым предиктором и откликом в них моделируется с использованием кривой. Наконец, полностью нелинейные методы, такие как *бэггинг*, *бустинг* и *метод опорных векторов* с нелинейными ядрами, обсуждаемые в главах 8 и 9, являются чрезвычайно гибкими подходами, которые трудно интерпретировать.

обобщенные аддитивные модели

бэггинг бустинг метод опорных векторов

Мы выяснили, что в случаях, когда целью анализа является статистический вывод, есть явные преимущества в использовании простых и относительно негибких методов статистического обучения. Однако в некоторых ситуациях мы заинтересованы только в предсказании, а интерпретируемость предсказательной модели нам просто неинтересна. Например, если мы пытаемся разработать алгоритм для предсказания цены акций, то единственным нашим требованием к алгоритму будет точность прогноза — интерпретируемость значения не имеет. В такой ситуации мы могли бы ожидать, что лучше всего будет использовать наиболее гибкую из имеющихся моделей. Удивительно, но это не всегда так! Часто мы будем получать более точные предсказания с помощью менее гибкого метода. Это явление, которое на первый взгляд может показаться нелогичным, объясняется склонностью более гибких методов к переобучению. Мы видели пример переобучения на рис. 2.6. Подробнее мы будем обсуждать эту важную концепцию в разделе 2.2 и в других частях книги.

2.1.4 Обучение с учителем и без учителя

с учителем и без учителя

Большинство проблем статистического обучения попадает в одну из двух категорий: *обучение с учителем* и *без учителя*. Все рассмотренные до сих пор примеры из этой главы относятся к категории обучения с учителем. Для каждого измеренного значения предиктора x_i, где $i = 1, \ldots, n$, имеется соответствующее значение отклика y_i. Мы желаем построить модель, которая описывает связь между откликом и предикторами с целью точного предсказания отклика для будущих наблюдений (прогнозирование) или для лучшего понимания взаимоотношений между откликом и предикторами (статистические выводы). Многие классические методы статистического обучения, такие как линейная регрессия и *логистическая регрессия* (глава 4), а также более современные подходы, вроде GAM, бустинга и метода опорных векторов, относятся к категории обучения с учителем. Таким проблемам посвящена большая часть этой книги.

логистическая регрессия

Обучение без учителя, в свою очередь, описывает несколько более сложную ситуацию, в которой для каждого наблюдения $i = 1, \ldots, n$ мы имеем вектор измерений x_i, но без соответствующего отклика y_i. Подогнать линейную регрессионную модель невозможно, поскольку подлежащая предсказанию зависимая переменная отсутствует. При таком сцена-

[2] Аббревиатура от «generalized additive models». — *Прим. пер.*

рии мы в некотором смысле работаем вслепую: подобную ситуацию называют обучением *без учителя*, поскольку у нас нет зависимой переменной, которая «руководила» бы нашим анализом. Какой статистический анализ все же возможен? Мы можем попытаться понять взаимоотношения между переменными или между наблюдениями. Один из статистических инструментов, который мы можем использовать в такой ситуации, — это *кластерный анализ*, или кластеризация. Цель кластерного анализа заключается в установлении того, разделяются ли наблюдения $x_1, \ldots, x_n$ на относительно четко выраженные группы. Например, в исследовании по сегментированию рынка мы могли бы учесть различные характеристики (переменные) для потенциальных клиентов, такие как почтовый индекс, семейный доход и покупательские привычки. Мы могли бы ожидать, что клиенты образуют разные группы, такие как «тратящие много» и «тратящие мало». Если бы информация по стоимости покупок была доступна для каждого клиента, тогда было бы возможным обучение с учителем. Однако такая информация отсутствует, т. е. мы не знаем покупательную способность потенциальных клиентов. При таком сценарии мы можем попытаться выполнить кластерный анализ на основе измеренных переменных с целью определить обособленные группы клиентов. Обнаружение подобных групп может представлять интерес в связи с тем, что они могут различаться в отношении некоторой интересной характеристики, такой как типичные траты на покупки.

кластерный анализ

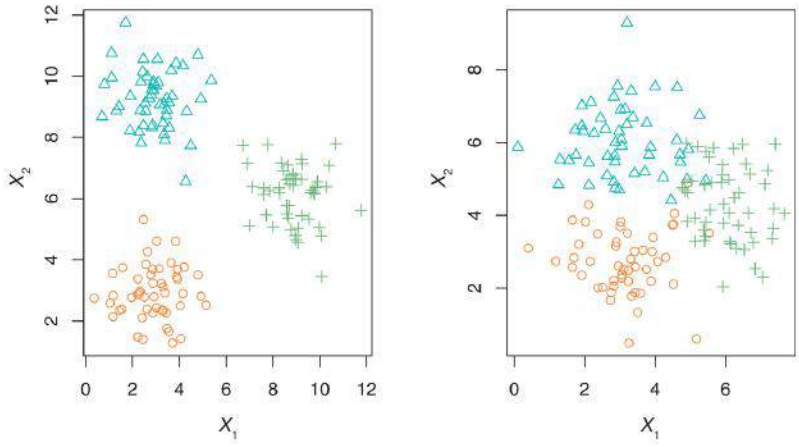

РИСУНОК 2.8. *Кластеризация данных, содержащих три группы. Каждая группа показана с использованием разных цветных символов.* Слева: *три группы хорошо разделены. В такой ситуации кластерный анализ успешно их обнаружит.* Справа: *группы в определенной степени перекрываются. Здесь задача кластеризации является более сложной*

На рис. 2.8 представлен простой пример проблемы кластеризации. Мы изобразили 150 наблюдений со значениями двух переменных — X_1 и X_2. Каждое наблюдение соответствует одной из трех отдельных групп. В целях демонстрации члены каждой группы обозначены нами в виде разноцветных символов. Однако на практике групповые принадлежности неиз-

вестны, и задача заключается именно в определении группы, к которой относится каждое наблюдение. На рис. 2.8 слева эта задача довольно проста, поскольку группы хорошо разделены. В то же время рисунок справа иллюстрирует более трудную задачу, в которой имеется некоторое перекрытие между группами. Нельзя ожидать, что в этом случае метод кластеризации правильно отнесет все перекрывающиеся точки к их соответствующим группам (голубой, зеленой или оранжевой).

В примерах, показанных на рис. 2.8, есть только две переменные, и поэтому для обнаружения кластеров можно просто визуально изучить диаграммы рассеяния. Однако на практике мы часто сталкиваемся с данными, которые содержат намного больше, чем две переменные. В таких случаях представить наблюдения на графике непросто. Например, если в нашем наборе данных есть p переменных, то можно будет построить $p(p-1)/2$ отдельных диаграмм рассеяния, и их визуальное инспектирование для обнаружения кластеров будет просто невыполнимо. По этой причине важную роль играют методы автоматической кластеризации. Мы обсуждаем кластеризацию и другие методы обучения без учителя в главе 10.

Многие проблемы естественным образом распадаются на категории обучения с учителем и без учителя. Однако иногда вопрос о том, к какой из этих двух категорий следует отнести конкретный анализ, менее ясен. Предположим, например, что у нас есть набор из n наблюдений. Для m наблюдений, где $m < n$, у нас имеются измерения как для предикторов, так и для отклика. Для остальных $n-m$ наблюдений у нас есть измерения предикторов, но нет отклика. Подобный сценарий может возникать, когда измерение предикторов является относительно дешевым, но сбор информации по соответствующим откликам сопряжен с гораздо большими затратами. Мы называем такую ситуацию проблемой *обучения смешанного типа*. В этом случае желательно применить такой метод статистического обучения, который включает как те m наблюдений, для которых измерения отклика доступны, так и те $m-n$ наблюдений, для которых такие измерения отсутствуют. Хотя это очень интересная тема, она лежит за рамками данной книги.

обучение смешанного типа

2.1.5 Различия между проблемами регрессии и классификации

типы переменных

Переменные можно охарактеризовать либо как *количественные*, либо как *качественные* (последние известны также как «*категориальные*»). Количественные переменные принимают числовые значения. В качестве примеров можно привести возраст, рост и доход человека, стоимость дома и цену акций. В свою очередь, качественные переменные принимают значения, соответствующие одному из K различных *классов*, или категорий. Примеры качественных переменных: пол человека (мужской или женский), бренд купленного продукта (A, B или C), переход в категорию неплательщиков по долгам (да или нет), а также диагноз ракового заболевания (острая миелоидная лейкемия, острая лимфобластическая лейкемия или отсутствие лейкемии). Обычно проблемы, связанные с количественным откликом, называют проблемами *регрессии*, тогда как проблемы, связанные с качественным откликом, часто называют проблемами *классифика-*

класс

регрессия и классификация

ции. Однако это различие не всегда бывает таким четким. Регрессия по методу наименьших квадратов (глава 3) используется для количественного отклика, тогда как логистическая регрессия обычно применяется для качественного *бинарного* отклика (т. е. переменной с двумя классами). Поэтому логистическая регрессия часто используется как метод классификации. Но поскольку этот метод оценивает вероятности классов, о нем можно думать также и как о методе регрессии. Некоторые статистические методы, такие как метод K ближайших соседей (главы 2 и 4) и бустинг (глава 8), могут применяться как для количественных, так и для качественных откликов.

Обычно мы выбираем метод на основе того, является ли отклик количественным или качественным; например, мы могли бы использовать линейную регрессию для количественного отклика и логистическую регрессию — для качественного. В то же время тип предикторов (т. е. являются ли они количественными или качественными) обычно считается менее важным. Большинство обсуждаемых в этой книге методов статистического обучения может применяться вне зависимости от типа предикторов, при условии что перед выполнением анализа все качественные предикторы должным образом *закодированы*.

2.2 Описание точности модели

Одна из ключевых целей этой книги заключается в том, чтобы представить читателю широкий круг методов статистического обучения, выходящих далеко за рамки стандартного подхода линейной регрессии. Зачем описывать столько разных методов статистического обучения вместо одного, *самого лучшего* метода? *В статистике бесплатный сыр бывает только в мышеловке*: ни один из методов не доминирует над другими для всех возможных наборов данных. На некотором конкретном наборе данных тот или иной метод может оказаться оптимальным, тогда как другие методы могут сработать лучше на похожих или отличающихся данных. Поэтому выбор метода, который дает наилучшие результаты для имеющегося набора данных, представляет собой важную задачу. Выбор такого метода может оказаться одним из самых трудных аспектов применения статистического обучения на практике.

В этом разделе мы обсуждаем некоторые из наиболее важных концепций, возникающих при выборе процедуры статистического обучения для того или иного набора данных. По мере дальнейшего изложения мы будем объяснять, как представленные здесь концепции можно применять на практике.

2.2.1 Измерение качества модели

Чтобы оценить эффективность некоторого метода статистического обучения в отношении имеющегося набора данных, нам необходим способ для измерения того, насколько хорошо полученные при помощи этого метода предсказания совпадают с наблюдаемыми данными. Другими словами, нам нужно количественно выразить степень того, насколько предсказанное значение отклика близко к истинному значению отклика у соответ-

ствующего наблюдения. В регрессионном анализе наиболее часто используемой для этого мерой является *среднеквадратичная ошибка* (MSE)[3], вычисляемая как

среднеквадратичная ошибка

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2, \tag{2.5}$$

где $\hat{f}(x_i)$ — это предсказанное значение, которое $\hat{f}$ дает для i–го наблюдения. MSE будет низкой, если предсказанные значения отклика очень близки к истинным значениям, и высокой, если для некоторых наблюдений предсказанные и истинные значения существенно разнятся.

MSE в (2.5) вычисляется на основе обучающих данных, которые были использованы для подгонки модели, и поэтому точнее будет называть ее *ошибкой на обучающей выборке*[4]. Однако обычно нас не очень заботит то, как хорошо метод работает на обучающих данных. Вместо этого *мы заинтересованы в точности предсказаний, которые мы получаем, когда применяем наш метод к не использованным ранее контрольным данным*. Почему это нас заботит? Представьте, что мы заинтересованы в разработке алгоритма предсказания цены акций на основе их доходности в прошлом. Мы можем обучить некоторый метод, используя значения доходности акций за последние 6 месяцев. Однако нам не очень интересно, насколько хорошо наш метод предсказывает значения цены акций для прошлой недели. Вместо этого нас интересует то, как хорошо он будет предсказывать цену на завтра или для следующего месяца. В том же ключе представьте, что у нас есть измерения клинических показателей (например, вес, давление крови, рост, возраст, семейная история болезни) для нескольких пациентов, а также информация о том, болен ли каждый пациент диабетом. Мы можем использовать этих пациентов, чтобы обучить статистический метод для предсказания риска диабета на основе клинических измерений. В действительности мы хотим, чтобы этот метод верно предсказывал риск диабета для *будущих пациентов* на основе их клинических измерений. Нам не очень интересно, насколько точно метод предсказывает риск диабета для пациентов, использованных для обучения модели, поскольку мы уже знаем, кто из этих пациентов болен.

ошибка на обучающей выборке

контрольная выборка

Чтобы выразить это математически, представьте, что мы подгоняем нашу статистическую модель к обучающим наблюдениям $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ и получаем оценку $\hat{f}$. Далее мы можем вычислить $\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n)$. Если полученные значения примерно равны $y_1, y_2, \ldots, y_n$, то MSE на обучающей выборке, рассчитанная по (2.5), окажется низкой. Однако нам совсем нет дела то того, что $\hat{f}(x_i) \approx y_i$; вместо этого мы хотим знать, является ли величина $\hat{f}(x_0)$ примерно равной y_0, где (x_0, y_0) — *неизвестное ранее проверочное наблюдение, которое не использовалось при обучении статистической модели*. Мы хотим выбрать такой метод, который дает минимальную *ошибку на контрольной выборке*[5], а не минимальную ошибку на обучающей выборке. Другими словами,

ошибка на контрольной выборке

[3] Аббревиатура от «mean squared error». — *Прим. пер.*

[4] Синонимами являются также термины «ошибка на обучающих данных», «ошибка на обучающем множестве» и «ошибка обучения». — *Прим. пер.*

[5] Синонимами являются также термины «ошибка на проверочной выборке», «ошибка на тестовой выборке» и «ошибка на проверочном множестве». — *Прим. пер.*

располагая большим количеством наблюдений, мы могли бы вычислить

$$\text{Ave}(y_0 - \hat{f}(x_0))^2, \tag{2.6}$$

то есть среднеквадратичную ошибку предсказаний для этих проверочных наблюдений (x_0, y_0). Мы хотели бы выбрать такую модель, для которой это среднее значение — среднеквадратичная ошибка на контрольной выборке — является как можно меньшим.

Каким же образом нам выбрать метод, минимизирующий MSE на контрольной выборке? В некоторых случаях у нас в распоряжении может оказаться набор контрольных данных, т. е. мы можем иметь доступ к набору наблюдений, которые не были использованы для обучения статистической модели. Тогда мы можем просто вычислить величину из (2.6) по этим проверочным наблюдениям и выбрать метод статистического обучения с наименьшей контрольной MSE. Но как быть, если проверочные наблюдения недоступны? В этом случае теоретически можно было бы выбрать метод статистического обучения, который минимизирует MSE на обучающей выборке. Это могло бы показаться разумным подходом, поскольку MSE на обучающей и контрольной выборках выглядят тесно связанными между собой. К сожалению, у этой стратегии есть фундаментальная проблема: нет гарантии, что метод с минимальной MSE на обучающих данных также будет иметь минимальную MSE на контрольных данных. Грубо говоря, проблема заключается в том, что многие статистические методы специально оценивают коэффициенты с целью минимизации MSE на обучающей выборке. Ошибка обучения у этих методов может быть довольно низкой, но MSE на контрольных данных часто гораздо выше.

Рисунок 2.9 иллюстрирует этот феномен на простом примере. Слева на этом рисунке приведены наблюдения, которые были сгенерированы нами на основе (2.1), а истинная функция f показана кривой черного цвета. Оранжевая, голубая и зеленая кривые иллюстрируют три возможные оценки f, полученные при помощи методов с возрастающим уровнем гибкости. Оранжевая линия представляет собой линейную регрессионную модель, которая относительно негибка. Голубая и зеленая кривые были получены при помощи обсуждаемых в главе 7 *сглаживающих сплайнов* с разными уровнями гладкости. Хорошо видно, что по мере увеличения гибкости кривые все ближе подступают к наблюдениям. Зеленая кривая является наиболее гибкой и описывает данные очень хорошо; однако мы видим, что она слабо напоминает истинную функцию f (показана черным) в силу свой извилистости. Изменяя уровень гибкости сглаживающего сплайна, мы можем подогнать к этим данным много разных моделей.

сглаживающий сплайн

Перейдем теперь к графику, приведенному на рис. 2.9 справа. Серая линия показывает среднее значение MSE на обучающей выборке в зависимости от уровня гибкости, или, более формально, от *числа степеней свободы*, для нескольких сглаживающих сплайнов. Число степеней свободы представляет собой величину, которая обобщает гибкость кривой; более подробно это понятие обсуждается в главе 7. Оранжевые, голубые и зеленые квадраты показывают ошибки обучения, связанные с соответствующими кривыми на рисунке слева. Более ограниченная и, следовательно, более гладкая кривая имеет меньше степеней свободы, чем извилистая кривая (заметьте, что на рис. 2.9 линейная регрессия имеет две степени свободы и является наиболее ограниченной). MSE на обучающей выборке

число степеней свободы

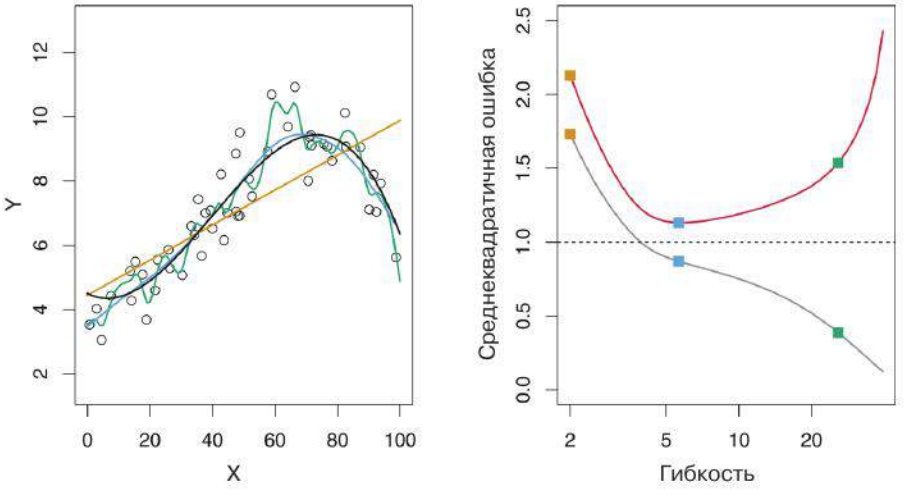

РИСУНОК 2.9. Слева: *данные, имитированные на основе f, показаны полыми точками черного цвета. Представлены три способа подгонки f: линейная регрессия (оранжевая линия) и две модели гладких сплайнов (голубая и зеленая линии).* Справа: *среднеквадратичная ошибка на обучающих (серая линия) и контрольных данных (красная линия), а также минимально возможное значение ошибки для контрольных данных (пунктирная линия). Квадратные символы соответствуют ошибкам на обучающих и контрольных выборках, которые были получены для трех моделей, изображенных на графике слева*

монотонно снижается по мере возрастания гибкости. В этом примере истинная функция f нелинейна, в связи с чем линейная модель (оранжевый цвет) недостаточно гибка для получения удовлетворительной оценки f. Из всех трех методов зеленая кривая имеет наименьшую MSE на обучающих данных, поскольку она соответствует наиболее гибкой из трех моделей, представленных на графике слева.

В этом примере истинная функция f нам известна, и поэтому мы также можем вычислить MSE для очень большой контрольной совокупности в зависимости от гибкости модели. (Конечно, как правило, f неизвестна, и такие вычисления будут невозможны.) MSE на контрольной выборке показана на рис. 2.9 справа в виде красной кривой. Как и в случае с MSE на обучающих данных, MSE на контрольной выборке сначала снижается по мере увеличения гибкости. Однако в какой–то момент MSE на контрольной выборке снова начинает возрастать. Как следствие оранжевая и зеленая кривые имеют самые высокие ошибки на контрольных данных. Голубая кривая имеет минимальную MSE на контрольной выборке, что не должно удивлять, поскольку визуально эта модель выглядит как наилучшая оценка f (см. рис. 2.9 слева). Горизонтальная прерывистая линия показывает $Var(\epsilon)$ — неустранимую ошибку из (2.3), которая соответствует наименьшей достижимой MSE на контрольных данных для всех возможных методов. Следовательно, сглаживающий сплайн, показанный в виде голубой линии, близок к оптимуму.

Как видно на рис. 2.9 (справа), по мере возрастания гибкости метода статистического обучения происходит монотонное снижение MSE на обучающих данных и *U–образное* изменение MSE на контрольных данных. Это является фундаментальным свойством статистического обучения, которое остается справедливым для любого имеющегося набора данных и для любого применяемого статистического метода. По мере увеличения гибкости модели MSE на обучающей выборке будет снижаться, но MSE на контрольной выборке необязательно будет вести себя тем же образом. Ситуацию, когда некоторый метод обеспечивает небольшую MSE на обучающих данных и высокую MSE на проверочных данных, называют *переобучением* модели. Это происходит потому, что наша процедура статистического обучения слишком усердно пытается найти закономерности в обучающих данных и в результате может обнаружить некоторые закономерности, которые просто случайны и никак не связаны с истинными свойствами неизвестной функции f. При переобучении модели MSE на контрольной выборке будет большой потому, что предполагаемые закономерности, найденные нашим методом в обучающих данных, в проверочных данных просто не существуют. Заметьте, что вне зависимости от того, случилось ли переобучение, мы почти всегда ожидаем, что MSE на обучающих данных будет ниже, чем MSE на контрольных данных, поскольку большинство методов статистического обучения так или иначе пытаются минимизировать ошибку обучения. К переобучению относят также частный случай, когда менее гибкая модель обеспечивает меньшую MSE на контрольных данных.

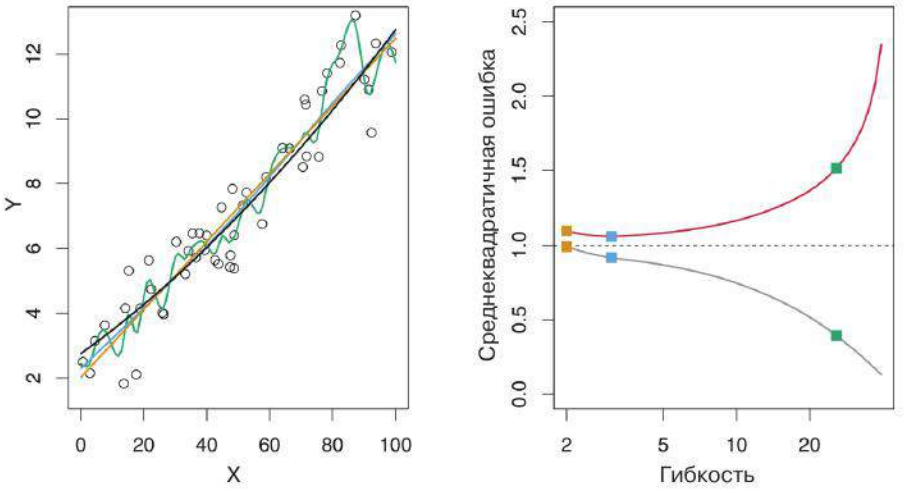

РИСУНОК 2.10. *То же, что на рис. 2.9, но с истинной функцией f, которая намного ближе к линейной. В этой ситуации линейная регрессия очень хорошо описывает данные*

На рис. 2.10 приведен пример, в котором истинная функция f приблизительно линейна. Как и раньше, мы наблюдаем монотонное снижение MSE на обучающих данных по мере возрастания гибкости модели, а кривая MSE на контрольных данных имеет U–образную форму. Одна-

ко в связи с тем, что истинная зависимость близка к линейной, MSE на контрольных данных лишь незначительно снижается перед последующим возрастанием, и поэтому оранжевая линия, подогнанная по методу наименьших квадратов, подходит для описания данных гораздо лучше, чем очень гибкая зеленая кривая. Наконец, рис. 2.11 иллюстрирует пример, в котором функция f в значительной мере нелинейна. Кривые MSE для обучающих и контрольных выборок по–прежнему демонстрируют те же общие закономерности, но теперь имеет место быстрое снижение обеих этих кривых, перед тем как MSE на контрольных данных начинает медленно возрастать.

На практике вычислить MSE по обучающим данным относительно легко, однако оценить MSE на контрольных данных гораздо труднее, поскольку они обычно отсутствуют. Как показывают предыдущие три примера, уровень гибкости, соответствующий модели с наименьшей контрольной MSE, может значительно варьировать в зависимости от свойств данных. В этой книге мы обсуждаем целый ряд подходов, которые можно использовать на практике получения оценки этого минимального значения. Одним из важных методов, предназначенных для оценивания MSE на контрольных данных, является *перекрестная проверка*[6] (глава 5).

перекрестная проверка

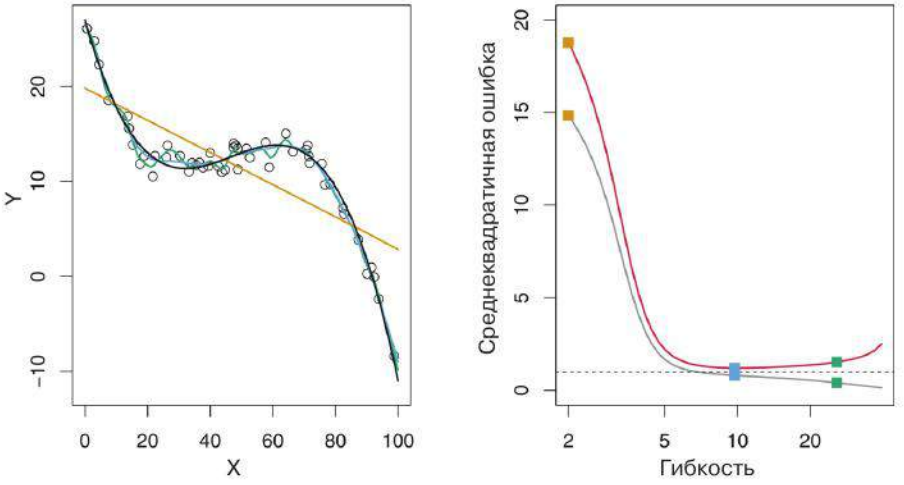

РИСУНОК 2.11. *То же, что на рис. 2.9, но с истинной функцией f, существенно отличной от линейной. В этой ситуации линейная регрессия описывает данные очень плохо*

2.2.2 Компромисс между смещением и дисперсией

Оказывается, что U–образная форма кривых, описывающих MSE на контрольных данных (рис. 2.9–2.11), является результатом двух конкурирующих свойств методов статистического обучения. Хотя математическое доказательство выходит за рамки этой книги, можно показать, что для

[6] Используются также термины «кросс–проверка», «кросс–валидация» и «скользящий контроль». — *Прим. пер.*

некоторого контрольного значения x_0 ожидаемую MSE всегда можно разложить на сумму трех фундаментальных величин: *дисперсии* $\hat{f}(x_0)$, квадрата *смещения* $\hat{f}(x_0)$ и дисперсии остатков ϵ. Другими словами[7]:

дисперсия
смещение

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + Var(\epsilon). \qquad (2.7)$$

Здесь $E\left(y_0 - \hat{f}(x_0)\right)^2$ обозначает *математическое ожидание* ошибки на контрольной выборке и представляет собой среднее значение MSE, которое мы получили бы при многократном повторном оценивании f на основе большого числа обучающих выборок и вычислении ошибки для каждого контрольного значения x_0. Общую ожидаемую MSE на контрольной выборке можно вычислить путем усреднения $E\left(y_0 - \hat{f}(x_0)\right)^2$ для всех возможных проверочных значений x_0.

математическое ожидание ошибки

Уравнение (2.7) говорит нам о том, что для минимизации ожидаемой ошибки на проверочных данных мы должны выбрать такой метод статистического обучения, который одновременно обеспечивает *низкую дисперсию* и *низкое смещение*. Заметьте, что дисперсия по определению является положительной величиной, равно как и квадрат смещения. В итоге мы видим, что математическое ожидание MSE на контрольной выборке никогда не может быть ниже неустранимой ошибки $\mathrm{Var}(\epsilon)$ из уравнения (2.3).

Что мы понимаем под *дисперсией* и *смещением* метода статистического обучения? *Дисперсия* означает величину, на которую $\hat{f}$ изменилась бы при оценивании этой функции с использованием другой обучающей выборки. Поскольку для подгонки модели используются обучающие данные, то разные обучающие выборки будут приводить к разным $\hat{f}$. В идеале оценки f, полученные на разных выборках, должны варьировать незначительно. Однако если некоторый метод обладает высокой дисперсией, то небольшие изменения в обучающих данных могут привести к большим изменениям в $\hat{f}$. В целом более гибкие статистические методы имеют более высокую дисперсию. Рассмотрим зеленую и оранжевую кривые на рис. 2.9. Гибкая зеленая кривая очень близко следует за отдельными наблюдениями. Она обладает высокой дисперсией, поскольку изменение любого наблюдения может вызвать значительное изменение оценки $\hat{f}$. В то же время подогнанная по методу наименьших квадратов оранжевая кривая является относительно негибкой и имеет низкую дисперсию, поскольку сдвиг любого наблюдения приведет, скорее всего, лишь к небольшому сдвигу положения этой кривой.

В свою очередь, *смещение* означает ошибку, вводимую за счет аппроксимирования проблемы из реального мира, которая может оказаться чрезвычайно сложной, при помощи гораздо более простого метода. Например, линейная регрессия предполагает, что между Y и $X_1, X_2, \ldots, X_p$ имеется линейная зависимость. Маловероятно, что какая–либо проблема из реального мира действительно описывается такой простой зависимостью, в связи с чем применение линейной регрессии несомненно приведет к определенному смещению оценки f. На рис. 2.11 истинная функция f явно нелинейна, и поэтому не важно, как много наблюдений у нас есть

[7] В соответствии с англоязычными терминами Var и Bias в приведенной формуле обозначают дисперсию и смещение соответственно. — *Прим. пер.*

в распоряжении — верную оценку при помощи линейной регрессии получить здесь будет невозможно. Другими словами, линейная регрессия в этом примере вызывает большое смещение. Однако на рис. 2.10 истинная функция f очень близка к линейной, и поэтому при наличии достаточного объема данных линейная регрессия сможет дать верную оценку. В целом более гибкие методы вызывают меньшее смещение.

Как правило, при использовании более гибких методов дисперсия будет возрастать, а смещение — снижаться. Относительная скорость изменения этих двух величин определяет направление изменения MSE на контрольной выборке. По мере увеличения гибкости некоторого класса методов снижение смещения вначале обычно происходит быстрее, чем рост дисперсии. В результате этого математическое ожидание MSE на контрольной выборке снижается. Однако в определенный момент возрастающая гибкость перестает оказывать влияние на смещение, но при этом вызывает рост дисперсии. Когда это происходит, MSE на контрольной выборке возрастает. Заметьте, что мы уже наблюдали такое начальное снижение ошибки на контрольных данных и последующее ее возрастание (см. графики, представленные справа на рис. 2.9–2.11).

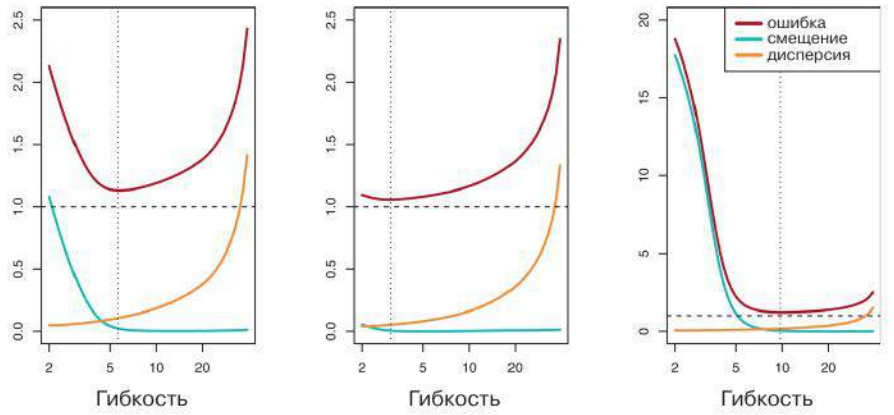

РИСУНОК 2.12. *Квадрат смещения (голубая кривая), дисперсия (оранжевая кривая), Var(ϵ) (пунктирная линия) и MSE на контрольной выборке (красная кривая) для трех наборов данных, показанных на рис. 2.9–2.11. Вертикальная пунктирная линия показывает уровень гибкости, соответствующий наименьшей MSE*

Три графика на рис. 2.12 иллюстрируют свойства уравнения (2.7) на примере данных, показанных на рис. 2.9–2.11. В каждом случае сплошная синяя линия показывает квадрат смещения для разных уровней гибкости, а оранжевая кривая соответствует дисперсии. Горизонтальная прерывистая линия отражает $\mathrm{Var}(\epsilon)$ — неустранимую ошибку. Наконец, красная кривая, соответствующая MSE на контрольной выборке, является суммой этих трех величин. Во всех трех случаях по мере увеличения гибкости метода дисперсия возрастает, а смещение снижается. Однако уровень гибкости, соответствующий оптимальной MSE на контрольной выборке, значительно различается между этими тремя наборами данных, поскольку

квадрат смещения и дисперсия в каждом случае изменяются с разными скоростями. На рис. 2.12 слева уровень смещения поначалу быстро падает, вызывая резкое начальное снижение математического ожидания MSE на контрольной выборке. С другой стороны, на центральном графике рис. 2.12 истинная функция f близка к линейной, в связи с чем имеет место лишь небольшое снижение уровня смещения по мере увеличения гибкости, а MSE на контрольной выборке незначительно снижается перед началом быстрого роста по мере увеличения дисперсии. Наконец, на рисунке 2.12 справа с увеличением гибкости наблюдается значительное падение уровня смещения, поскольку истинная функция f существенно нелинейна. Имеется также очень небольшое возрастание дисперсии по мере роста гибкости. В результате MSE на контрольной выборке значительно падает, перед тем, как снова начать медленно возрастать по мере увеличения гибкости.

Связь между смещением, дисперсией и MSE на контрольной выборке, описанная в уравнении 2.7 и показанная на рис. 2.12, известна как *компромисс между смещением и дисперсией*[8]. Хороший результат, достигаемый методом статистического обучения на контрольных данных, требует как низкой дисперсии, так и низкого квадрата смещения. Это называют компромиссом, поскольку можно легко получить метод с чрезвычайно низким смещением, но высокой дисперсией (например, нарисовав кривую, которая проходит через каждую точку обучающей выборки), или метод с очень низкой дисперсией, но высоким смещением (путем подгонки горизонтальной линии к данным). Трудность заключается в нахождении метода, у которого малы как дисперсия, так и квадрат смещения. Данный компромисс является одной из наиболее важных тем этой книги.

компромисс

В практической ситуации, когда функция f неизвестна, выполнить непосредственный расчет MSE на контрольных данных, уровня смещения и дисперсии для некоторого метода статистического обучения обычно невозможно. Тем не менее следует всегда помнить о компромиссе между смещением и дисперсией. В этой книге мы рассматриваем методы, которые являются чрезвычайно гибкими и поэтому могут фактически полностью устранить смещение. Однако это не гарантирует того, что они будут работать лучше более простого метода, такого как линейная регрессия. В качестве экстремального примера предположим, что истинная функция f является линейной. В этой ситуации линейная регрессия будет обладать нулевым смещением, в связи с чем более сложному методу будет сложно конкурировать. Однако если истинная функция f в значительной мере нелинейна и у нас есть много обучающих наблюдений, тогда, возможно, нам стоит воспользоваться более гибким методом, как показано на рис. 2.11. В главе 5 мы обсуждаем перекрестную проверку, которая представляет собой способ оценки контрольной MSE с использованием обучающих данных.

2.2.3 Задачи классификации

До сих пор при обсуждении качества моделей мы фокусировались на задачах регрессии. Однако многие из встретившихся нам концепций, таких как компромисс между смещением и дисперсией, переносятся на задачи классификации лишь с небольшими модификациями, обусловленны-

[8] В оригинале используется термин «bias–variance trade–off». — *Прим. пер.*

ми тем, что y_i больше не являются количественными значениями. Предположим, что мы пытаемся оценить f на основе обучающих наблюдений $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, где $y_1, \ldots, y_n$ теперь представляют собой значения качественной переменной. Наиболее распространенный способ количественного описания точности нашей оценки $\hat{f}$ заключается в расчете *частоты ошибок* на обучающей выборке, т. е. доли ошибок, допущенных при применении нашей оцененной функции $\hat{f}$ к обучающим наблюдениям:

частота ошибок

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i). \tag{2.8}$$

Здесь $\hat{y}_i$ представляет собой метку класса, предсказанную для i–го наблюдения при помощи $\hat{f}$. $I(y_i \neq \hat{y}_i)$ — это *индикаторная переменная*[9], равная 1 при $y_i \neq \hat{y}_i$ и 0 при $y_i = \hat{y}_i$. Если $I(y_i \neq \hat{y}_i) = 0$, то i–е наблюдение было предсказано нашим методом классификации правильно; иначе оно классифицировано неверно. Следовательно, уравнение (2.8) позволяет вычислить долю неправильно классифицированных случаев.

индикаторная переменная

ошибка обучения

Величину из уравнения 2.8 называют *частотой ошибок на обучающей выборке*, поскольку она рассчитывается на основе данных, используемых для обучения нашего классификатора. Как и в задачах регрессии, нам более всего интересны частоты ошибок, получаемые в результате применения нашего классификатора к контрольным наблюдениям, которые не использовались в ходе обучения модели. *Частота ошибок на контрольной выборке*, связанная с набором проверочных наблюдений вида (x_0, y_0), вычисляется как[10]

ошибка на контрольной выборке

$$\text{Ave}(I(y_0 \neq \hat{y}_0)), \tag{2.9}$$

где $\hat{y}_0$ представляет собой метку класса, полученную в результате применения классификатора к проверочному наблюдению с вектором предикторов x_0. *Хорошим* является тот классификатор, у которого ошибка (2.9) минимальна.

Байесовский классификатор

Можно показать (хотя математическое доказательство выходит за рамки данной книги), что средняя частота ошибок на контрольных данных, приведенная в (2.9), минимизируется очень простым классификатором, который *присваивает каждому наблюдению наиболее вероятный класс с учетом соответствующих значений предикторов*. Другими словами, нам следует отнести проверочное наблюдение с вектором предикторов x_0 к такому классу j, для которого вероятность

$$\Pr(Y = j | X = x_0) \tag{2.10}$$

условная вероятность

максимальна. Заметьте, что (2.10) представляет собой *условную вероятность*[11], т. е. вероятность того, что $Y = j$ при наблюдаемом векторе предикторов x_0. Этот очень простой классификатор называют *байесовским*

[9] В оригинале используется термин «indicator variable». В русскоязычных источниках применяется также термин «фиктивная переменная». — *Прим. пер.*

[10] Ave означает усреднение (от «average»). — *Прим. пер.*

[11] В оригинале используется термин «conditional probability». — *Прим. пер.*

классификатором. В проблеме с двумя классами, где имеется только два возможных значения отклика, например *класс 1* или *класс 2*, байесовский классификатор предсказывает класс 1, если $\Pr(Y = 1|X = x_0) > 0.5$, и класс 2 в остальных случаях.

байесовский классификатор

На рис. 2.13 приведен пример с использованием имитированного набора данных в двумерном пространстве, образованном предикторами X_1 и X_2. Оранжевые и синие кружки соответствуют обучающим наблюдениям, которые принадлежат к двум разным классам. Для каждого значения X_1 и X_2 имеется своя вероятность того, что отклик относится к «оранжевому» или «синему классу». Поскольку это имитированные данные, то мы знаем, как они были получены, и мы можем вычислить условные вероятности для каждой пары значений X_1 и X_2. Закрашенная оранжевым цветом область отражает множество точек, для которых $\Pr(Y = \text{оранжевый}|X)$ превышает 50%, тогда как область, закрашенная синим цветом, показывает множество точек, для которых эта вероятность ниже 50%. Фиолетовая пунктирная линия соответствует точкам, для которых вероятность в точности равна 50%. Эта линия называется *байесовской решающей границей*[12]. Предсказание байесовского классификатора определяется байесовской решающей границей: наблюдение, попадающее в оранжевую область, будет отнесено к «оранжевому классу», а наблюдение, попадающее в синюю область, — к «синему классу».

байесовская решающая граница

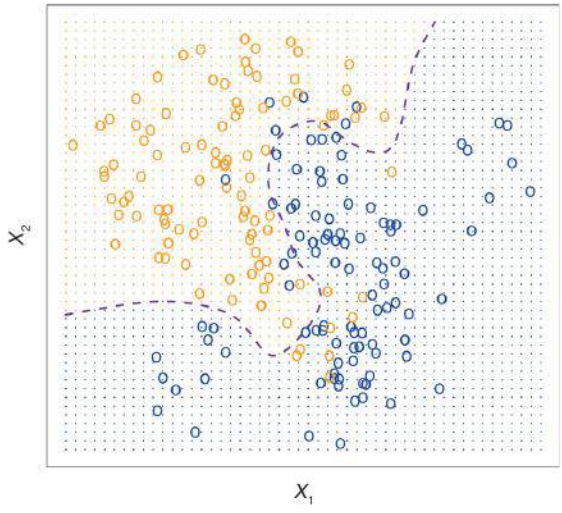

РИСУНОК 2.13. *Набор имитированных данных, состоящий из 100 наблюдений в каждой группе (обозначены синим и оранжевым цветами). Фиолетовая пунктирная линия соответствует байесовской решающей границе. Фоновая сетка оранжевого цвета обозначает область, в которой наблюдение из контрольной выборки будет отнесено к «оранжевому» классу, а сетка синего цвета соответствует области, в которой контрольное наблюдение будет отнесено к «синему» классу*

[12] В оригинале используется термин «Bayes decision boundary». — *Прим. пер.*

байесовская частота ошибок

Байесовский классификатор обеспечивает наименьшую возможную частоту ошибок на контрольной выборке, называемую *байесовской частотой ошибок*. Поскольку байесовский классификатор всегда выберет класс, для которого величина (2.10) максимальна, частота ошибок при $X = x_0$ составит $1 - \max_j \Pr(Y = j|X = x_0)$. В целом общая байесовская частота ошибок вычисляется как

$$1 - E\left(\max_j \Pr(Y = j|X)\right), \tag{2.11}$$

где математическое ожидание есть средняя вероятность, рассчитанная по всем возможным значениям X. Для наших имитированных данных байесовская частота ошибок составляет 0.1304. Это больше 0, поскольку классы в генеральной совокупности перекрываются и $\Pr(Y = j|X = x_0) < 1$ для некоторых значений x_0. Байесовская частота ошибок аналогична обсуждавшейся ранее неустранимой ошибке.

Метод K ближайших соседей

Теоретически для предсказания качественного отклика всегда желательно было бы использовать байесовский классификатор. Однако для реальных данных мы не знаем условного распределения Y при заданном X, и поэтому вычисление байесовского классификатора невозможно. Следовательно, байесовский классификатор служит в качестве недостижимого золотого стандарта, с которым сравниваются другие методы. Многие методы пытаются оценить условное распределение Y при заданном X и затем относят то или иное наблюдение к классу с наибольшей *оцененной* вероятностью. Одним из них является *метод K ближайших соседей* (KNN)[13]. Для некоторого положительного целого числа K и контрольного наблюдения x_0 классификатор KNN сначала определяет K наблюдений из обучающей выборки (обозначаются как $\mathcal{N}_0$), которые находятся максимально близко к x_0. Затем он оценивает условную вероятность для класса j как долю примеров в $\mathcal{N}_0$, у которых значение отклика равно j:

$$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j). \tag{2.12}$$

Наконец, KNN применяет теорему Байеса и относит проверочное наблюдение x_0 к классу с наибольшей вероятностью.

На рис. 2.14 приведен пример, поясняющий метод KNN. На графике слева мы изобразили небольшой обучающий набор данных, состоящий из шести голубых и шести оранжевых точек. Наша цель — сделать предсказание для точки, обозначенной черным крестиком. Предположим, что мы выбрали $K = 3$. Тогда KNN сначала определит три наблюдения, расположенных к крестику ближе всего. Эта близлежащая область обозначена кругом. Она содержит две синие точки и одну оранжевую, что приводит к вероятности 2/3 для «синего класса» и 1/3 для «оранжевого класса». Следовательно, KNN предскажет, что черный крестик принадлежит к «синему классу». На рис. 2.14 справа мы применили метод KNN с $K = 3$

[13] Аббревиатура от «K–nearest neighbors». — *Прим. пер.*

для всех возможных значений X_1 и X_2 и изобразили соответствующую решающую границу.

Несмотря на то что это очень простой подход, KNN часто может приводить к созданию классификаторов, которые удивительно близки к оптимальному байесовскому классификатору. На рис. 2.15 показана решающая граница метода KNN с $K = 10$, примененного к большему по размеру набору имитированных данных из рис. 2.13. Обратите внимание: даже несмотря на то, что истинная решающая граница классификатору KNN неизвестна, его решающая граница очень похожа на границу байесовского классификатора. Частота ошибок на контрольной выборке при использовании KNN составляет 0.1363, что близко к байесовской частоте ошибок — 0.1304.

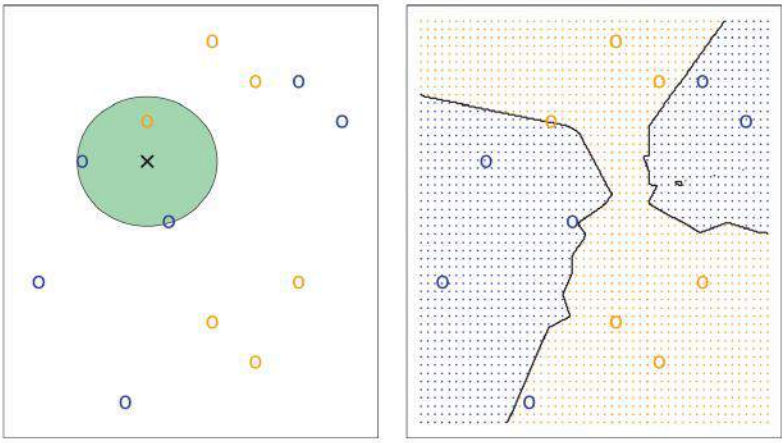

РИСУНОК 2.14. *Метод KNN с $K = 3$ проиллюстрирован на простом примере с шестью «синими» и шестью «оранжевыми» наблюдениями.* Слева: *подлежащее классификации наблюдение из контрольной выборки показано черным крестиком. Определены три ближайшие к этому наблюдению точки, в результате чего оно отнесено к наиболее часто встречающемуся среди соседей классу — в данном случае к «синему классу».* Справа: *решающая граница KNN из этого примера показана линией черного цвета. Синяя сетка соответствует области, в которой наблюдение из контрольной выборки будет отнесено к «синему классу», а сетка оранжевого цвета — области, в которой наблюдение из контрольной выборки будет отнесено к «оранжевому классу»*

Выбор K имеет огромный эффект на итоговый классификатор KNN. На рис. 2.16 приведены две модели KNN, подогнанные к имитированным данным из рис. 2.13 с использованием $K = 1$ и $K = 100$. При $K = 1$ решающая граница чрезмерно гибка и обнаруживает закономерности в данных, которые не согласуются с байесовской решающей границей. Это соответствует классификатору с низким смещением и очень высокой дисперсией. По мере увеличения K метод становится менее гибким и формирует решающую границу, похожую на прямую линию. Это соответствует классификатору с низкой дисперсией и высоким смещением. Для этих

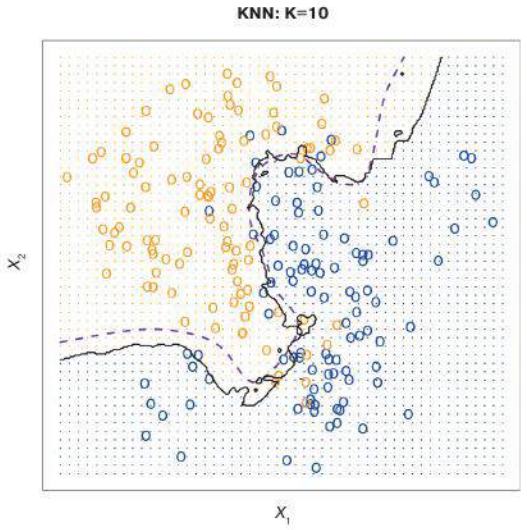

РИСУНОК 2.15. *Черная линия показывает решающую границу KNN для данных из рис. 2.13, найденную с использованием* $K = 10$. *Байесовская решающая граница представлена в виде фиолетовой пунктирной линии. Эти две решающие границы очень похожи*

имитированных данных ни $K = 1$, ни $K = 100$ не дают хороших предсказаний: частоты ошибок на контрольных данных составляют 0.1695 и 0.1925 соответственно.

Как и в случае с регрессией, сильной связи между ошибкой на обучающих данных и ошибкой на контрольных данных нет. При $K = 1$ ошибка обучения KNN равна 0, однако ошибка на контрольных данных может быть довольно высокой. Как правило, при использовании более гибких методов частота ошибки на обучающей выборке будет снижаться, но частота ошибок на контрольной выборке не обязательно будет вести себя тем же образом. На рис. 2.17 мы изобразили ошибки KNN на обучающих и проверочных данных в зависимости от $1/K$. При увеличении $1/K$ этот метод становится более гибким. Подобно регрессионным моделям, при увеличении гибкости частота ошибок на обучающих данных равномерно снижается. Однако ошибка на контрольных данных проявляет характерную U–образную форму, поначалу снижаясь (с минимумом примерно на отметке $K = 10$), а затем снова возрастая, когда метод становится чрезмерно гибким и приводит к переобучению.

При решении задач как регрессии, так и классификации выбор правильного уровня гибкости является критическим для успеха любого метода статистического обучения. Компромисс между смещением и дисперсией и вытекающая из него U–образная форма кривой ошибки на контрольных данных могут сделать этот выбор тяжелой задачей. В главе 5 мы вернемся к данной теме и обсудим различные подходы для оценивания частот ошибок на контрольных выборках, а тем самым и нахождения оптимального уровня гибкости для того или иного метода статистического обучения.

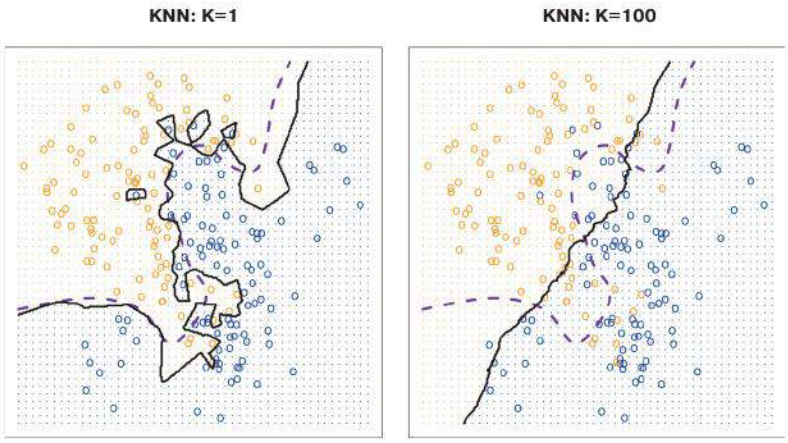

РИСУНОК 2.16. *Сравнение решающих границ KNN (черные сплошные линии), полученных по данным из рис. 2.13 с использованием* $K = 1$ *и* $K = 100$. *При* $K = 1$ *решающая граница получается чрезмерно гибкой, тогда как при* $K = 100$ *она недостаточно гибка. Байесовская решающая граница показана в виде фиолетовой пунктирной линии*

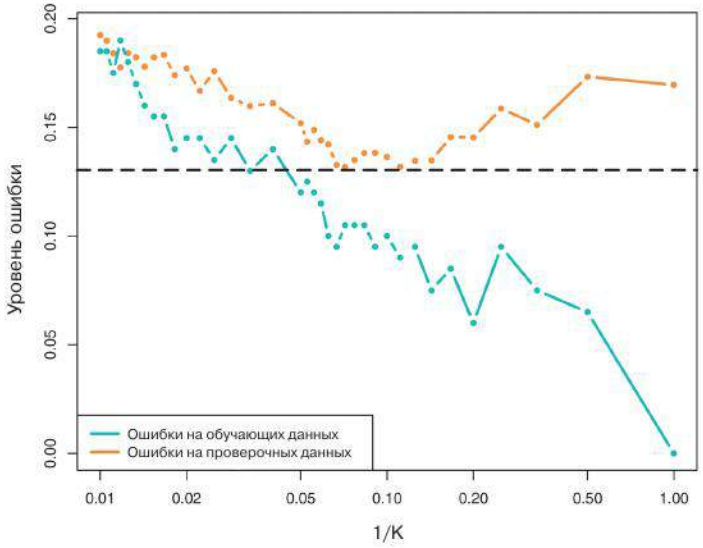

РИСУНОК 2.17. *Ошибки KNN–классификатора на обучающей (синяя кривая, 200 наблюдений) и контрольной выборках (оранжевая линия, 5000 наблюдений) для данных из рис. 2.13, показанные в зависимости от возрастающей гибкости* $(1/K)$ *или, что эквивалентно, в зависимости от снижающегося числа соседей* K. *Черная пунктирная линия показывает байесовскую частоту ошибок. Извилистость кривых обусловлена малым размером обучающей выборки*

2.3 Лабораторная работа: введение в R

В этой лабораторной работе мы представим некоторые простые команды R. Лучший способ изучения нового языка заключается в экспериментировании с его командами. R можно загрузить с сайта

http://cran.r-project.org/

2.3.1 Основные команды

функция

Для выполнения тех или иных операций R использует *функции*. Для запуска функции с именем funcname мы набираем funcname(input1, input2), где входные параметры, или *аргументы*, input1 и input2 сообщают R, как именно следует исполнить эту функцию. Например, для создания вектора с несколькими числами мы используем функцию *конкатенации* c()[14]. Следующая команда говорит R объединить числа 1, 3, 2 и 5 и сохранить их в виде вектора с именем x. Когда мы наберем x, то в ответ получим этот вектор.

аргумент

c()

вектор

```
> x <- c(1, 3, 2, 5)
> x
[1] 1 3 2 5
```

Обратите внимание на то, что > не является частью команды — R выводит этот знак просто, чтобы показать свою готовность к выполнению следующей команды. Мы можем сохранять объекты не только с помощью <-, но также и =:

```
> x = c(1, 6, 2)
> x
[1] 1 6 2
> y = c(1, 4, 3)
```

Многократное нажатие клавиши со стрелкой *вверх* приведет к показу предыдущих команд, которые можно отредактировать. Это полезно, поскольку необходимость повторения похожих команд возникает часто. Кроме того, ввод команды ?funcname всегда откроет новое окно со справочным файлом, содержащим дополнительную информацию по функции funcname.

Мы можем попросить R выполнить сложение двух чисел. В этом случае программа сначала добавит первое число из x к первому числу из y и т. д. Однако x и y должны быть одинаковой длины. Мы можем проверить их длину при помощи функции length().

length()

```
> length(x)
[1] 3
> length(y)
[1] 3
> x + y
[1] 2 10 5
```

[14] «Concatenate» значит «соединять», «объединять». — *Прим. пер.*

Функция `ls()` позволяет просмотреть список всех объектов, таких как данные и функции, которые мы сохранили до этого момента. Функцию `rm()` можно использовать для удаления любого нежелательного объекта. `ls()` `rm()`

```
> ls()
[1] "x" "y"
> rm(x, y)
> ls()
character (0)
```

Имеется также возможность удалить все объекты за один раз:

```
> rm(list = ls())
```

Функцию `matrix()` можно использовать для создания матрицы с числами. Перед применением функции `matrix()` мы можем узнать о ней больше: `matrix()`

```
> ?matrix
```

Справочный файл сообщает, что функция `matrix()` принимает несколько входных параметров, но пока мы сосредоточимся на первых трех: данные (элементы матрицы), количество строк и количество столбцов. Сначала создадим простую матрицу.

```
> x = matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Обратите внимание, что мы могли бы с тем же успехом не набирать `data =`, `nrow =` и `ncol =` в приведенной выше команде `matrix()`, а просто ввести

```
x = matrix(c(1, 2, 3, 4), 2, 2)
```

и это имело бы тот же эффект. Однако иногда бывает полезно указать имена аргументов, т. к. иначе `R` будет предполагать, что аргументы функции подаются на нее в том же порядке, который приведен в справочном файле этой функции. Как показано в данном примере, по умолчанию `R` создает матрицы путем последовательного заполнения столбцов. В качестве альтернативы можно использовать опцию `byrow = TRUE` для заполнения матрицы по строкам.

```
> matrix(c(1, 2, 3, 4), 2, 2, byrow = TRUE)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Заметьте, что в приведенной выше команде мы не присвоили матрице никакого имени вроде `x`. В этом случае матрица выводится на экран, но не сохраняется для будущих вычислений. Функция `sqrt()` возвращает квадратный корень каждого элемента вектора или матрицы. Команда `x^2` возводит каждый элемент `x` в квадрат; возможно использование любой степени, включая дроби и отрицательные степени.

```
> sqrt(x)
     [,1] [,2]
[1,] 1.00 1.73
[2,] 1.41 2.00
> x^2
     [,1] [,2]
[1,]    1    9
[2,]    4   16
```

rnorm()

Функция rnorm() генерирует вектор случайных нормально распределенных значений, при этом аргумент n задает размер выборки. Каждый раз при вызове этой функции мы будем получать другой результат. Здесь мы создаем два коррелирующих набора чисел — x и y — и применяем функцию cor() для расчета корреляции между ними.

cor()

```
> x = rnorm(50)
> y = x + rnorm(50, mean = 50, sd = .1)
> cor(x, y)
[1] 0.995
```

По умолчанию rnorm() создает случайные переменные, представляющие стандартное нормальное распределение со средним значением 0 и стандартным отклонением 1. Однако, как показано выше, среднее значение и стандартное отклонение можно изменить при помощи аргументов mean и sd. Иногда мы хотим, чтобы наш код в точности воспроизводил один и тот же набор случайных чисел; для этого мы можем использовать функцию set.seed(). Функция set.seed() принимает в качестве аргумента произвольное целое число.

set.seed()

```
> set.seed(1303)
> rnorm(50)
[1] -1.1440 1.3421 2.1854 0.5364 0.0632 0.5022 -0.0004
. . .
```

Мы используем set.seed() во всех лабораторных работах каждый раз, когда выполняем вычисления, содержащие случайные величины. Как правило, это должно помочь пользователю воспроизвести наши результаты. Однако следует отметить, что с появлением новых версий R могут возникнуть небольшие несоответствия между книгой и тем, что выдает R.

mean()
var()
sd()

Функции mean() и var() можно использовать для вычисления среднего значения и дисперсии некоторого набора чисел. Применение sqrt() к результату работы var() даст стандартное отклонение. Или можно просто применить функцию sd().

```
> set.seed(3)
> y = rnorm(100)
> mean(y)
[1] 0.0110
> var(y)
[1] 0.7329
> sqrt(var(y))
[1] 0.8561
> sd(y)
[1] 0.8561
```

2.3.2 Графики

Использование функции `plot()` является основным способом визуализации данных в R. Например, `plot(x, y)` создаст график зависимости значений `y` от значений `x`. Имеется большое количество дополнительных опций, которые можно подать на функцию `plot()`. Например, использование аргумента `xlab` приведет к появлению заголовка оси X. Для получения дополнительной информации о функции `plot()` введите команду `?plot`.

```
> x = rnorm(100)
> y = rnorm(100)
> plot(x, y)
> plot(x, y, xlab = "this is the x-axis",
       ylab = "this is the y-axis", main = "Plot of X vs Y")
```

Часто у нас будет возникать желание сохранить построенный в R график. Используемая для этого команда будет зависеть от типа файла, который мы хотим создать. Например, для создания PDF–файла мы используем функцию `pdf()`, а для создания JPEG–файла — функцию `jpeg()`.

`pdf()`
`jpeg()`

```
> pdf("Figure.pdf")
> plot(x, y, col = "green")
> dev.off()
null device
1
```

Функция `dev.off()` сообщает R, что мы завершили создание графика. В качестве альтернативы мы можем просто скопировать содержимое графического окна и вставить его в файл нужного типа, вроде документа Word.

Функцию `seq()` можно использовать для создания последовательности чисел. Например, `seq(a, b)` создает вектор из целых чисел от `a` до `b`. Есть много опций: например, `seq(0, 1, length = 10)` создаст последовательность из 10 чисел, которые равномерно размещены на промежутке от `0` до `1`. Команда `3:11` является сокращенным вариантом `seq(3, 11)` для целых чисел.

`seq()`

```
> x = seq(1, 10)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> x = 1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
> x = seq(-pi, pi, length = 50)
```

Теперь мы построим несколько более сложные графики. Функция `contour()` создает *контурную диаграмму* для изображения трехмерных данных, которая напоминает топографическую карту. Эта функция принимает три аргумента:

`contour()`
контурная диаграмма

1. Вектор значений `x` (первое измерение).

2. Вектор значений `y` (второе измерение).

3. Матрица, чьи элементы соответствуют значениям z (третье измерение) для каждой пары координат (x, y).

Как и в случае с функцией plot(), есть много входных параметров, которые можно использовать для тонкой настройки функции contour(). Чтобы узнать о них подробнее, просмотрите справочный файл, набрав ?contour.

```
> y = x
> f = outer(x, y, function(x, y) cos(y)/(1 + x^2))
> contour(x, y, f)
> contour(x, y, f, nlevels = 45, add = T)
> fa = (f - t(f))/2
> contour(x, y, fa, nlevels = 15)
```

image()

Функция image() работает так же, как и contour(), за исключением того, что она создает цветную диаграмму, где цвет зависит от значения z. Такая диаграмма известна как *тепловая карта* и иногда используется для изображения температуры в прогнозах погоды. В качестве альтернативы для создания трехмерных графиков можно использовать persp(). Аргументы theta и phi контролируют углы обзора графика.

тепловая карта

persp()

```
> image(x, y, fa)
> persp(x, y, fa)
> persp(x, y, fa, theta = 30)
> persp(x, y, fa, theta = 30, phi = 20)
> persp(x, y, fa, theta = 30, phi = 70)
> persp(x, y, fa, theta = 30, phi = 40)
```

2.3.3 Индексирование данных

Часто мы хотим изучить только некоторую часть данных. Предположим, что наши данные хранятся в матрице A.

```
> A = matrix(1:16, 4, 4)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Тогда выполнение команды

```
> A[2, 3]
[1] 10
```

приведет к выбору элемента, соответствующего второй строке и третьему столбцу. Первое число после открывающей скобки [всегда соответствует строке, а второе — столбцу. Мы можем также выбрать несколько строк и столбцов одновременно, указав векторы в качестве индексов.

```
> A[c(1, 3), c(2, 4)]
     [,1] [,2]
```

```
[1,]    5   13
[2,]    7   15
> A[1:3, 2:4]
     [,1] [,2] [,3]
[1,]    5    9   13
[2,]    6   10   14
[3,]    7   11   15
> A[1:2,]
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
> A[, 1:2]
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

В последних двух примерах отсутствует индекс либо для столбцов, либо для строк. Это означает, что следует включать все столбцы или все строки соответственно. R рассматривает одну строку или столбец матрицы как вектор.

```
> A[1, ]
[1] 1 5 9 13
```

Использование знака минус - в индексе говорит R, что нужно оставить все строки или столбцы, за исключением указанных в этом индексе.

```
> A[-c(1, 3), ]
     [,1] [,2] [,3] [,4]
[1,]    2    6   10   14
[2,]    4    8   12   16
> A[-c(1, 3), -c(1, 3, 4)]
[1] 6 8
```

Функция dim() выводит размерность матрицы, т. е. число содержащихся в ней строк, за которым следует число столбцов. dim()

```
> dim(A)
[1] 4 4
```

2.3.4 Загрузка данных

Для большинства анализов первым шагом является импортирование набора данных в R. Функция read.table() — один из основных инструментов для этого. Справочный файл содержит детали, касающиеся использования этой функции. Для экспортирования данных мы можем использовать функцию write.table(). read.table() write.table()

Перед попыткой загрузить некоторый набор данных мы должны убедиться в том, что R знает правильную директорию, где нужно его искать. Например, в системе Windows можно было бы выбрать директорию при помощи опции Change dir... из раздела File главного меню. Однако

детали того, как это сделать, зависят от используемой операционной системы (Windows, Mac, Unix), и поэтому дополнительную информацию мы здесь не приводим. Мы начинаем с загрузки набора данных Auto. Эти данные являются частью пакета ISLR (мы обсуждаем пакеты в главе 3), но для демонстрации функции read.table() мы загружаем их из текстового файла. Следующая команда загрузит файл Auto.data в R и сохранит его в виде объекта Auto в формате, который называется «таблица данных»[15]. (Этот текстовый файл можно найти на сайте книги.) После загрузки данных можно применить функцию fix() для просмотра этих данных в окне, напоминающем электронную таблицу. Однако перед вводом любых следующих команд эта таблица должна быть закрыта.

таблица данных

```
> Auto = read.table("Auto.data")
> fix(Auto)
```

Заметьте, что Auto.data — это просто текстовый файл, который вы могли бы открыть также на своем компьютере при помощи стандартного текстового редактора. Часто перед загрузкой того или иного набора данных в R хорошей идеей будет просмотреть его с использованием текстового редактора или других программ, таких как Excel.

Этот конкретный набор данных не был загружен корректно, поскольку система R предположила, что имена переменных являются частью данных и поэтому включила их в первую строку. Эти данные содержат также несколько пропущенных значений, отмеченных вопросительным знаком ?. Пропущенные значения — частое явление в реальных данных. Использование опции header = T (header = TRUE) при вызове функции read.table() сообщает R о том, что первая строка файла содержит имена переменных, а использование опции na.strings говорит R о том, что каждый раз при обнаружении определенного сочетания символов (например, вопросительного знака) это сочетание следует рассматривать в качестве пропущенного элемента матрицы.

```
> Auto = read.table("Auto.data", header = T, na.strings = "?")
> fix(Auto)
```

Excel — это программа для хранения данных в одном из распространенных форматов. Легким способом загрузки таких данных в R является сохранение их в виде CSV–файла (от «comma separated values», т. е. «значения, разделенные запятыми») и использование функции read.csv() для загрузки.

```
> Auto = read.csv("Auto.csv", header = T, na.strings = "?")
> fix(Auto)
> dim(Auto)
[1] 397 9
> Auto [1:4, ]
```

dim() Функция dim() сообщает нам, что данные содержат 397 наблюдений, или строк, и девять переменных, или столбцов. Есть разные способы работы с пропущенными значениями. В данном случае пропущенные значения содержатся только в пяти строках, и поэтому мы используем функцию na.omit() na.omit(), чтобы просто удалить эти строки.

[15] В оригинале используется термин «data frame». — *Прим. пер.*

```
> Auto = na.omit(Auto)
> dim(Auto)
[1] 392 9
```

После корректной загрузки данных мы можем применить функцию `names()` для просмотра имен переменных. names()

```
> names(Auto)
[1] "mpg"          "cylinders"      "displacement"  "horsepower"
[5] "weight"       "acceleration"   "year"          "origin"
[9] "name"
```

2.3.5 Дополнительные графические и количественные сводки

Мы можем воспользоваться функцией `plot()` для создания *диаграмм рассеяния* количественных переменных. Однако простой ввод имен переменных приведет к сообщению об ошибке, поскольку `R` не знает, что эти переменные нужно искать в таблице `Auto`. диаграмма рассеяния

```
> plot(cylinders, mpg)
Error in plot(cylinders, mpg) : object 'cylinders' not found
```

Для обращения к переменной мы должны указать название набора данных и имя этой переменной, объединенные знаком `$`. В качестве альтернативы мы можем использовать функцию `attach()`, чтобы сообщить `R` о том, что переменные в этом наборе данных нужно сделать доступными по их имени.

```
> plot(Auto$cylinders, Auto$mpg)
> attach(Auto)
> plot(cylinders, mpg)
```

Переменная `cylinders`[16] хранится в виде количественного вектора, и поэтому она была распознана `R` как количественная переменная. Но поскольку имеется лишь небольшое число возможных значений `cylinders`, эту переменную, возможно, следует рассматривать как качественную. Функция `as.factor()` конвертирует количественные переменные в качественные. as.factor()

```
> cylinders = as.factor(cylinders)
```

Если отображенная на оси X переменная является качественной, то функция `plot()` автоматически построит *диаграмму размахов*. Как обычно, для настройки графика можно указать целый ряд опций. диаграмма размахов

```
> plot(cylinders, mpg)
> plot(cylinders, mpg, col = "red")
> plot(cylinders, mpg, col = "red", varwidth = T)
> plot(cylinders, mpg, col = "red", varwidth = T, horizontal=T)
> plot(cylinders, mpg, col = "red", varwidth = T,
       xlab = "cylinders", ylab = "MPG")
```

[16] Речь идет о количестве цилиндров в двигателе автомобиля. — *Прим. пер.*

hist()
гистограмма

Для построения *гистограммы* можно использовать функцию `hist()`. Заметьте, что `col = 2` имеет тот же эффект, что и `col = "red"`.

```
> hist(mpg)
> hist(mpg, col = 2)
> hist(mpg, col = 2, breaks = 15)
```

матрица диаграмм рассеяния

Функция `pairs()` создает *матрицу диаграмм рассеяния*, т. е. диаграмму рассеяния для каждой пары переменных из того или иного набора данных. Мы можем также построить диаграммы рассеяния только для определенного подмножества переменных.

```
> pairs(Auto)
> pairs(~ mpg + displacement + horsepower + weight +
       acceleration, Auto)
```

Сочетание функций `plot()` и `identify()` обеспечивает полезный интерактивный метод определения значений конкретной переменной на графике. Мы подаем на `identify()` три аргумента: переменные для отображения на осях X и Y и переменную, чьи значения мы хотели бы указать рядом с каждой точкой. Тогда щелчок мышью по некоторой точке на графике заставит `R` вывести соответствующее значение интересующей нас переменной. Щелчок правой кнопкой мыши по графику остановит работу функции `identify()` (Ctrl + щелчок на компьютерах Mac). Выведенные функцией `identify()` числа соответствуют порядковым номерам строк в таблице данных.

```
> plot(horsepower, mpg)
> identify(horsepower, mpg, name)
```

summary()

Функция `summary()` рассчитывает количественные сводки по каждой переменной в конкретном наборе данных.

```
> summary(Auto)
      mpg          cylinders      displacement
 Min.   : 9.00   Min.   :3.000   Min.   : 68.0
 1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0
 Median :22.75   Median :4.000   Median :151.0
 Mean   :23.45   Mean   :5.472   Mean   :194.4
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8
 Max.   :46.60   Max.   :8.000   Max.   :455.0

    horsepower       weight       acceleration
 Min.   : 46.0   Min.   :1613   Min.   : 8.00
 1st Qu.: 75.0   1st Qu.:2225   1st Qu.:13.78
 Median : 93.5   Median :2804   Median :15.50
 Mean   :104.5   Mean   :2978   Mean   :15.54
 3rd Qu.:126.0   3rd Qu.:3615   3rd Qu.:17.02
 Max.   :230.0   Max.   :5140   Max.   :24.80

       year           origin                      name
 Min.   :70.00   Min.   :1.000   amc matador       : 5
 1st Qu.:73.00   1st Qu.:1.000   ford pinto        : 5
 Median :76.00   Median :1.000   toyota corolla    : 5
```

```
Mean    :75.98   Mean    :1.577   amc gremlin        : 4
3rd Qu. :79.00   3rd Qu. :2.000   amc hornet         : 4
Max.    :82.00   Max.    :3.000   chevrolet chevette : 4
                                  (Other)           :365
```

Для качественных переменных, таких как `name`, `R` перечислит число наблюдений, попадающих в каждую категорию. Мы можем также получить сводку только по одной переменной.

```
> summary(mpg)
   Min.  1st Qu.  Median   Mean 3rd Qu.  Max.
   9.00   17.00   22.75  23.45  29.00  46.60
```

Закончив использование `R`, мы вводим `q()` для завершения работы, или для выхода из программы. При выходе из `R` у нас имеется возможность сохранить текущую *рабочую среду*, чтобы в следующий раз сделать доступными все объекты, которые мы создали во время этой сессии работы с `R`. Перед выходом из `R` мы можем захотеть сохранить историю всех команд, выполненных во время последней сессии; это можно сделать при помощи функции `savehistory()`. В следующий раз при запуске `R` мы можем загрузить эту историю команд при помощи функции `loadhistory()`.

`q()`

рабочая среда

`save history()`

`load history()`

2.4 Упражнения

Теоретические

1. Для каждого из пунктов (a)–(d) укажите, будем ли мы обычно ожидать от гибкого статистического метода лучшего или худшего качества предсказаний, по сравнению с негибким методом. Обоснуйте свой ответ.

   (a) Размер выборки n очень большой, а количество предикторов p невелико.

   (b) Количество предикторов p очень большое, а число наблюдений n невелико.

   (c) Связь между предикторами и откликом в значительной мере нелинейна.

   (d) Дисперсия остатков, т. е. $\sigma^2 = \mathrm{Var}(\epsilon)$, чрезвычайно велика.

2. Объясните, является ли каждый из описанных ниже сценариев задачей на классификацию или регрессию, и укажите, заинтересованы ли мы больше в статистическом выводе или в предсказании. Наконец, приведите n и p.

   (a) Мы собираем данные по 500 наиболее успешным компаниям США. Для каждой компании мы учитываем доход, число работников, отрасль и заработную плату исполнительного директора. Мы заинтересованы в понимании факторов, которые влияют на зарплату исполнительного директора.

(b) Мы собираемся запустить новый продукт и хотим знать, будет он *успешным* или *неуспешным*. Мы собираем данные по 20 похожим продуктам, запущенным ранее. Для каждого продукта мы учитываем его успех или провал, продажную цену, величину маркетингового бюджета, цену у конкурента и десять других переменных.

(c) Мы заинтересованы в предсказании процентного изменения курса доллара США в зависимости от недельных изменений цены акций на мировых рынках. В связи с этим мы собираем недельные данные для всего 2012 года. Для каждой недели мы учитываем процентное изменение курса доллара, а также процентные изменения цены акций на рынках США, Великобритании и Германии.

3. Теперь мы снова вернемся к разложению ошибки предсказаний на дисперсию и смещение.

(a) Приведите набросок графика, где одновременно изображено поведение типичных кривых квадрата смещения, дисперсии, ошибки на обучающих данных, ошибки на контрольных данных и байесовской, или неустранимой, ошибки по мере увеличения гибкости методов статистического обучения. Ось X должна отражать уровень гибкости метода, а ось Y — значения для каждой кривой. Должно быть пять кривых. Убедитесь, что вы пометили каждую кривую.

(b) Объясните, почему каждая из пяти кривых имеет ту форму, которую вы изобразили, выполняя (a).

4. Теперь поразмышляем над некоторыми случаями практического применения методов статистического обучения.

(a) Опишите три прикладные задачи, где полезной могла бы оказаться *классификация*. Опишите отклик и предикторы. Что является целью каждой задачи — статистический вывод или предсказание? Объясните свой ответ.

(b) Опишите три прикладные задачи, где полезной могла бы оказаться *регрессия*. Опишите отклик и предикторы. Что является целью каждой задачи — статистический вывод или предсказание? Объясните свой ответ.

(c) Опишите три прикладные задачи, где полезным мог бы оказаться *кластерный анализ*.

5. В чем заключаются преимущества и недостатки очень гибких методов регрессии и классификации (по сравнению с менее гибкими методами)? В каких условиях можно было бы предпочесть более гибкий метод менее гибкому? Когда предпочтительным мог бы оказаться менее гибкий метод?

6. Опишите различия между параметрическим и непараметрическим подходами к статистическому обучению. Каковы преимущества параметрического подхода к регрессии и классификации (по сравнению с непараметрическим подходом)? Каковы недостатки?

7. В приведенной ниже таблице представлено обучающее множество, состоящее из шести наблюдений, трех предикторов и одной качественной переменной.

Наблюдение	X_1	X_2	X_3	Y
1	0	3	0	Красный
2	2	0	0	Красный
3	0	1	3	Красный
4	0	1	2	Зеленый
5	−1	0	1	Зеленый
6	1	1	1	Красный

Предположим, мы хотим использовать этот набор данных, чтобы предсказать Y по $X_1 = X_2 = X_3 = 0$ при помощи метода K ближайших соседей.

(a) Вычислите евклидово расстояние между каждым наблюдением и контрольной точкой $X_1 = X_2 = X_3 = 0$.

(b) Каково наше предсказание при $K = 1$? Почему?

(c) Каково наше предсказание при $K = 3$? Почему?

(d) Если байесовская решающая граница в этой проблеме является высоконелинейной, будет ли *оптимальное* значение K высоким или низким? Почему?

Практические

8. Это упражнение касается набора данных `College`, который можно найти в файле `College.csv`. Он содержит несколько переменных для 777 университетов и колледжей США. К этим переменным относятся:

- `Private`: индикатор того, является ли заведение публичным или частным;
- `Apps`: количество полученных заявлений на поступление;
- `Accept`: количество принятых абитуриентов;
- `Enroll`: количество новых зарегистрированных студентов;
- `Top10perc`: количество новых студентов, входивших в список 10% лучших учеников своего класса в средней школе;
- `Top25perc`: количество новых студентов, входивших в список 25% лучших учеников своего класса в средней школе;
- `F.Undegrad`: количество студентов, занимающихся полный рабочий день;
- `P.Undegrad`: количество студентов, занимающихся неполный рабочий день;
- `Outstate`: плата за обучение для студентов из других штатов;

- Room.Board: плата за проживание и питание;
- Books: стоимость учебников;
- Personal: примерные затраты на личные нужды;
- PhD: процент преподавателей с ученой степенью;
- Terminal: процент преподавателей с самой высокой ученой степенью в соответствующей области науки;
- S.F.Ratio: соотношение между численностью студентов и преподавателей;
- perc.alumni: процент бывших выпускников, делающих денежные пожертвования;
- Expend: затраты заведения на обучение одного студента;
- Grad.Rate: процент студентов, завершающих полный цикл обучения.

Перед загрузкой данных в R их можно просмотреть в программе Excel или в текстовом редакторе.

(a) Примените функцию read.csv() для загрузки данных в R. Присвойте загруженным данным имя college. Убедитесь, что вы задали правильную директорию, в которой хранятся эти данные.

(b) Просмотрите данные при помощи функции fix(). Вы должны заметить, что первый столбец просто содержит названия университетов. Мы не очень хотим, чтобы этот столбец рассматривался R как данные. Однако эти названия могут оказаться полезными позднее. Попробуйте следующие команды:

```
> rownames(college) = college[, 1]
> fix(college)
```

Вы должны увидеть, что теперь появился столбец row.names с названиями всех исследованных университетов. Это означает, что система R присвоила каждой строке имя, соответствующее тому или иному университету. R не будет пытаться выполнять вычисления над именами строк. Однако нам все еще нужно удалить первый столбец, в котором хранятся названия университетов. Попробуйте

```
> college = college[, -1]
> fix(college)
```

Теперь вы должны видеть, что первым столбцом в данных является Private. Заметьте, что перед столбцом Private появился еще один столбец с заголовком row.names. Однако этот столбец не является частью данных — это имена, присвоенные R каждой строке.

(c) i. Примените функцию summary() для вывода количественной сводки по имеющимся в таблице переменным.

ii. Примените функцию `pairs()` для построения матрицы диаграмм рассеяния по первым десяти столбцам, или переменным. Вспомните, что вы можете обратиться к первым десяти столбцам матрицы `A`, используя `A[, 1:10]`.

iii. Создайте новую качественную переменную `Elite`, *разбив на классы* переменную `Top10perc`. Мы собираемся разделить университеты на две группы в зависимости от того, превышает ли 50% доля студентов, входивших в список 10% лучших учеников своих классов в средней школе.

```
> Elite = rep("No", nrow(college))
> Elite[college$Top10perc > 50] = "Yes"
> Elite = as.factor(Elite)
> college = data.frame(college, Elite)
```

Примените функцию `summary()` для нахождения числа элитных университетов. Теперь примените функцию `plot()` для создания диаграммы размахов по переменным `Outstate` и `Elite`.

iv. Воспользуйтесь функцией `hist()` и для нескольких количественных переменных постройте гистограммы с разным числом классов. Полезной для вас может оказаться команда `par(mfrow = c(2, 2))`: она разделит окно графического устройства на четыре области, что сделает возможным построение четырех графиков одновременно. Изменение аргументов этой функции приведет к разделению экрана иным образом.

v. Продолжите изучение данных и приведите небольшое обобщение ваших находок.

9. Это упражнение касается набора данных `Auto`, рассмотренных в лабораторной работе. Убедитесь, что пропущенные значения в этой таблице были удалены.

(a) Какие предикторы являются количественными, а какие — качественными?

(b) Каков *размах* значений каждого предиктора? Вы можете ответить на этот вопрос при помощи функции `range()`. `range()`

(c) Каковы среднее значение и стандартное отклонение каждого количественного предиктора?

(d) Теперь удалите 10–е и 85–е наблюдения. Чему равны размах, среднее значение и стандартное отклонение каждого предиктора в оставшемся наборе данных?

(e) Используя полный набор данных, исследуйте предикторы графически при помощи диаграмм рассеяния или других инструментов на ваш выбор. Создайте несколько графиков, характеризующих взаимоотношения между предикторами. Прокомментируйте свои находки.

(f) Предположим, что мы хотим предсказать расход топлива (`mpg`) на основе других переменных. Указывают ли ваши графики

на то, что какие–то переменные могли бы оказаться полезными для предсказания mpg? Обоснуйте свой ответ.

10. Это упражнение касается набора данных Boston.

пакет

(a) Данные Boston являются частью *пакета* MASS.

```
> library(MASS)
```

Теперь данные содержатся в объекте Boston.

```
> Boston
```

Почитайте об этом наборе данных:

```
> ?Boston
```

Сколько строк в этом наборе данных? Сколько столбцов? Что представляют собой эти строки и столбцы?

(b) Сделайте несколько парных диаграмм рассеяния для предикторов (столбцов) из этого набора данных. Опишите свои находки.

(c) Связаны ли какие–то из предикторов с уровнем преступности в расчете на душу населения (переменная crim)? Если да, то объясните эту связь.

(d) Похоже ли, что некоторые из пригородов Бостона имеют особенно высокие уровни преступности? Налоговые ставки (tax)? Отношение численности учащихся к численности учителей (ptratio)? Прокомментируйте размах значений каждого предиктора.

(e) Через сколько пригородов из этого набора данных протекает река Чарльз (chas)?

(f) Чему равна медиана отношения численности учащихся к численности учителей в пригородах из этого набора данных?

(g) Какой из пригородов Бостона имеет самое низкое медианное значение числа домов, занятых их владельцами (medv)? Чему равны значения других предикторов у этого пригорода и насколько они отличаются от значений соответствующих предикторов в целом? Прокомментируйте свои находки.

(h) У какого количества пригородов из этого набора данных среднее число комнат в доме (rm) больше семи? Больше восьми? Опишите пригороды, где число комнат в доме в среднем выше восьми.

Глава 3

Линейная регрессия

Эта глава посвящена *линейной регрессии* — очень простому методу обучения с учителем. В частности, линейная регрессия является полезным инструментом для предсказания количественного отклика. Линейная регрессия применялась на протяжении длительного времени, и о ней написано бессчетное количество учебников. Хотя она может показаться несколько скучной в сравнении с некоторыми другими, более современными методами статистического обучения, описанными в последующих главах этой книги, линейная регрессия по–прежнему является полезным и широко применяемым методом. В связи с этим невозможно переоценить важность хорошего понимания линейной регрессии для последующего освоения других, более сложных методов обучения. В этой главе мы даем обзор некоторых ключевых идей, лежащих в основе линейной регрессионной модели, а также метода наименьших квадратов, который чаще всего используется для подгонки этой модели.

Вспомните данные `Advertising` из главы 2. На рис. 2.1 показан объем продаж некоторого продукта (`sales`, тыс. единиц) в зависимости от размера бюджета для рекламы на телевидении (`TV`), радио (`radio`) и в газетах (`newspaper`). Представьте, что нас в качестве консультантов–статистиков попросили воспользоваться этими данными и предложить такой маркетинговый план на следующий год, который приведет к высоким продажам продукта. Какая информация была бы полезной для разработки подобной рекомендации? Вот некоторые важные вопросы, на которые нам, возможно, следует обратить внимание:

1. *Существует ли зависимость между размером рекламного бюджета и объемом продаж?* Наша первая цель должна заключаться в установлении того, указывают ли данные на наличие связи между затратами на рекламу и продажами. Если оснований для такого заключения мало, то можно утверждать, что деньги на рекламу тратить не стоит.

2. *Насколько сильна связь между размером рекламного бюджета и продажами?* Допустив наличие связи между рекламой и объемом продаж, мы хотели бы выяснить силу этой связи. Другими словами, можем ли мы с высокой точностью предсказать объем продаж, исходя из заданного размера рекламного бюджета? Это был бы случай тесной связи. А может быть, прогноз продаж на основе затрат

на рекламу лишь немногим лучше случайного угадывания? Это был бы случай слабой связи.

3. *Какое средство массовой информации способствует продажам?* Способствуют ли продажам все три информационных канала — телевидение, радио и газеты, или только один или два канала? Для ответа на этот вопрос мы должны найти способ выделить эффекты отдельных СМИ при одновременной затрате денег на все три из них.

4. *Насколько точно мы можем оценить влияние каждого СМИ на продажи?* Насколько увеличатся продажи в расчете на каждый доллар, затраченный на рекламу в конкретном СМИ? Насколько точно мы можем предсказать этот прирост?

5. *Насколько точно мы можем спрогнозировать будущие продажи?* Каков наш прогноз объема продаж и насколько он точен для того или иного уровня затрат на рекламу на телевидении, радио и в газетах?

6. *Является ли связь линейной?* Если имеет место примерно прямолинейная зависимость между затратами на рекламу в разных СМИ и объемом продаж, то линейная регрессия будет подходящим методом. Если же это не так, то у нас все еще есть возможность преобразовать предикторы или отклик таким образом, чтобы сделать линейную регрессию применимой.

7. *Существует ли синергия между рекламой в разных СМИ?* Возможно, что затраты в размере 50 000$ на телерекламу и 50 000$ на радио–рекламу приводят к большим продажам, нежели затраты в размере 100 000$ на рекламу только на телевидении или только на радио. В маркетинге этот эффект известен как *синергия*, тогда в статистике его называют эффектом *взаимодействия*.

синергия
взаимодействие

Оказывается, что линейную регрессию можно использовать для получения ответов на каждый из этих вопросов. Сначала мы обсудим все эти вопросы в общем контексте, а затем вернемся к ним в разделе 3.4 в контексте этой конкретной задачи.

3.1 Простая линейная регрессия

простая линейная регрессия

Простая линейная регрессия полностью соответствует своему названию: это очень простой подход для предсказания количественного отклика Y на основе единственной независимой переменной X. Он подразумевает, что между X и Y существует примерно линейная зависимость. Математически мы можем записать линейную регрессию как

$$Y \approx \beta_0 + \beta_1 X. \tag{3.1}$$

Знак $\approx$ означает «*примерно моделируется как*». Иногда мы будем говорить о (3.1) как о *регрессии Y на X* (или Y *по* X). Например, X может представлять собой затраты на телерекламу (`TV`), а Y — объем продаж

(sales). Тогда мы можем построить регрессию sales по TV, подогнав модель

$$\texttt{sales} \approx \beta_0 + \beta_1 \texttt{TV}.$$

В уравнении 3.1 β_0 и β_1 — это константы, известные как *свободный член* и *угол наклона* линейной модели. В более общем виде β_0 и β_1 называют *коэффициентами*, или *параметрами*, модели. Применив наши обучающие данные для нахождения оценок коэффициентов β_0 и β_1, мы можем предсказать будущие объемы продаж на основе того или иного размера затрат на телерекламу путем вычисления

свободный член

угол наклона

коэффициент параметр

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x, \tag{3.2}$$

где $\hat{y}$ — это значение Y, предсказанное по $X = x$. Здесь мы используем символ ˆ («*крышечка*») для обозначения оцененного значения неизвестного параметра, или коэффициента, или для обозначения предсказанной величины отклика.

3.1.1 Оценивание коэффициентов

На практике β_0 и β_1 неизвестны. Поэтому перед тем, как мы сможем применить (3.1) для получения предсказаний, мы должны воспользоваться данными для нахождения оценок коэффициентов. Пусть

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

представляют собой n пар наблюдений, каждая из которых состоит из значения X и значения Y. В примере с данными Advertising у нас есть размер бюджета на телерекламу и объем продаж продукта в $n = 200$ различных регионах. (Напомним, что эти данные представлены на рис. 2.1.) Наша цель заключается в получении оценок коэффициентов β_0 и β_1 таким образом, чтобы линейная модель (3.1) хорошо описывала данные, т. е. чтобы $y_i \approx \hat{\beta}_0 + \hat{\beta}_1 x$ для $i = 1, \ldots, n$. Другими словами, мы хотим найти такие свободный член $\hat{\beta}_0$ и угловой коэффициент $\hat{\beta}_1$, чтобы итоговая линия располагалась максимально близко к $n = 200$ наблюдениям. Существует большое количество способов выразить *близость*. Однако наиболее обычный подход сводится к минимизации критерия *наименьших квадратов*, и мы применяем этот подход в данной главе. Альтернативные методы будут рассмотрены в главе 6.

критерий наименьших квадратов

Пусть $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ представляет собой значение Y, предсказанное на основе i–го значения X. Тогда $e_i = y_i - \hat{y}_i$ — это i–й *остаток*, т. е. разница между i–м наблюденным значением отклика и i–м значением отклика, предсказанным нашей линейной моделью. Мы определяем *сумму квадратов остатков* (RSS)[1] как

остаток

сумма квадратов остатков

$$\text{RSS} = e_1^2 + e_2^2 + \ldots e_n^2,$$

или в эквивалентной форме как

[1] Аббревиатура от «residual sum of squares». — *Прим. пер.*

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \cdots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2. \quad (3.3)$$

Метод наименьших квадратов находит такие $\hat{\beta}_0$ и $\hat{\beta}_1$, которые минимизируют RSS. Применив некоторые вычисления, можно показать, что эти оптимальные оценки представляют собой

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}, \quad (3.4)$$

где $\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i$ и $\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$ — выборочные средние. Другими словами, (3.4) дает определение *оценок коэффициентов* простой линейной регрессии по методу наименьших квадратов.

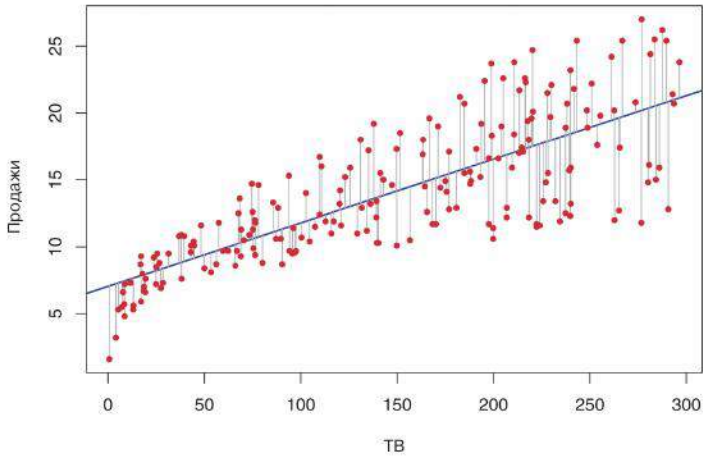

РИСУНОК 3.1. *Для данных* `Advertising` *показана регрессия объема продаж (*`sales`*) по затратам на телерекламу (*`TV`*), подогнанная по методу наименьших квадратов. Модель найдена путем минимизации суммы квадратов остатков. Каждый серый отрезок соответствует остатку, и модель подгоняется на основе компромисса, заключающегося в усреднении квадратов этих остатков. В данном случае линейная модель в целом передает суть зависимости, хотя в левой части графика она несколько неадекватна*

Рисунок 3.1 демонстрирует простую линейную регрессию, подогнанную по данным `Advertising`, где $\hat{\beta}_0 = 7.03$, а $\hat{\beta}_1 = 0.0475$. Иными словами, согласно этой аппроксимации, потраченные на телерекламу дополнительные 1000\$ приводят к продаже примерно 47.5 дополнительной единицы продукта. На рис. 3.2 мы рассчитали RSS для нескольких значений β_0 и β_1 по данным `Advertising`, используя `sales` в качестве отклика и `TV` в качестве предиктора. На каждой диаграмме красная точка представляет собой пару оценок (β_0, β_1), полученных по методу наименьших квадратов в соответствии с (3.4). Хорошо видно, что эти значения минимизируют RSS.

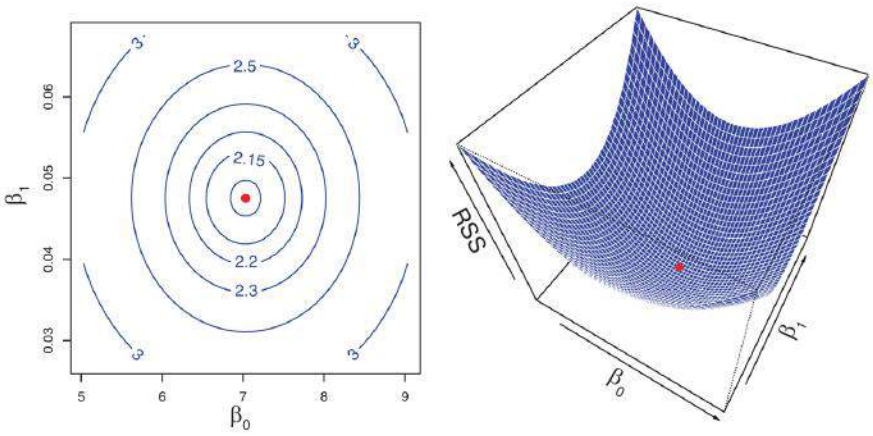

РИСУНОК 3.2. *Контурная и трехмерная диаграммы RSS для данных* `Advertising` *с использованием* `sales` *в качестве отклика и* `TV` *в качестве предиктора. Красные точки соответствуют оценкам* $\hat{\beta}_1$ *и* $\hat{\beta}_2$*, полученным по методу наименьших квадратов согласно (3.4)*

3.1.2 Точность оценок коэффициентов

Согласно (2.1), мы предполагаем, что *истинная* зависимость между X и Y принимает форму $Y = f(X) + \epsilon$ для некоторой неизвестной функции f, где ϵ — случайные ошибки с нулевым средним. Если f аппроксимируется линейной функцией, то мы можем записать эту зависимость как

$$Y = \beta_0 + \beta_1 X + \epsilon. \tag{3.5}$$

Здесь β_0 представляет собой свободный член уравнения, т. е. ожидаемое значение Y при $X = 0$, а β_1 — угловой коэффициент, т. е. среднее приращение Y, связанное с увеличением X на одну единицу. Остатки отражают все, что мы упускаем из вида: возможно, истинная зависимость не является линейной, могут существовать другие переменные, вызывающие вариацию Y, а также могут иметь место ошибки измерения. Обычно мы предполагаем, что остатки независимы от X.

Модель, представленная в (3.5), определяет *регрессионную линию генеральной совокупности*, которая является наилучшей линейной аппроксимацией истинной зависимости между X и Y[2]. Оценки регрессионных коэффициентов, полученные по методу наименьших квадратов согласно (3.4), характеризуют *линию наименьших квадратов* из (3.2). В качестве простого гипотетического примера на рис. 3.3 слева показаны две такие линии. Мы создали 100 случайных значений X и 100 соответствующих значений Y на основе модели

истинная регрессионная линия

линия наименьших квадратов

$$Y = 2 + 3X + \epsilon, \tag{3.6}$$

[2] Предположение о линейной форме зависимости часто является полезной рабочей моделью. Однако, несмотря на то, о чем говорят многие учебники, мы убеждены, что истинная зависимость является линейной только в редких случаях.

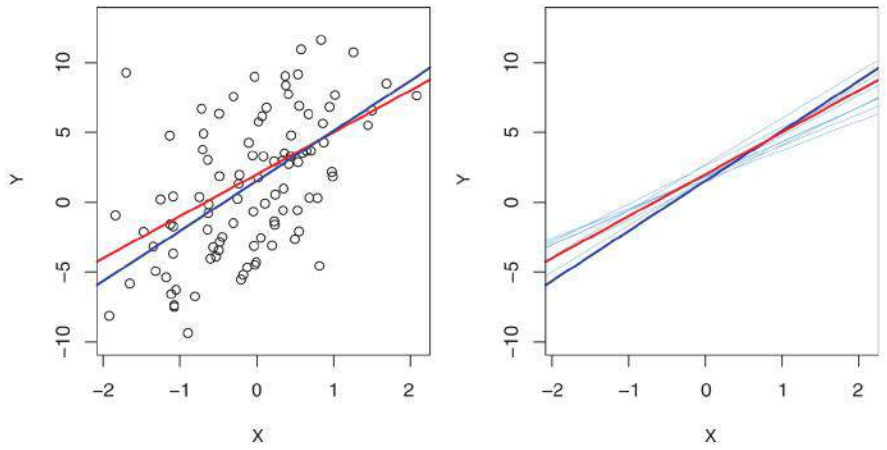

РИСУНОК 3.3. *Набор имитированных данных.* Слева: *красная линия соответствует истинной зависимости* $f(X) = 2 + 3X$, *которая известна как регрессионная линия генеральной совокупности. Синяя линия представляет собой линию наименьших квадратов — она является оценкой* $f(X)$, *полученной по методу наименьших квадратов на основе выборочных данных, показанных в виде полых черных точек.* Справа: *регрессионная линия генеральной совокупности по-прежнему показана красным цветом, а линия наименьших квадратов — темно-синим. Светло-синим цветом показаны десять линий наименьших квадратов, вычисленные на основе отдельных случайных выборок. Каждая линия наименьших квадратов отличается от других таких линий, но в среднем они довольно похожи на регрессионные линии генеральной совокупности*

где значения ϵ были извлечены из нормального распределения с нулевым средним. Красная линия на рис. 3.3 слева изображает *истинную* зависимость $f(X) = 2 + 3X$, тогда как голубая линия получена на основе данных по методу наименьших квадратов. Истинная зависимость для реальных данных обычно неизвестна, однако всегда можно вычислить линию наименьших квадратов, используя оценки коэффициентов, приведенные в (3.4). Иными словами, при решении практических задач у нас имеется доступ к набору наблюдений, по которым мы можем вычислить линию наименьших квадратов, однако регрессионная линия для генеральной совокупности остается ненаблюдаемой. На рис. 3.3 справа мы создали десять различных наборов данных на основе модели (3.6) и изобразили соответствующие десять линий наименьших квадратов. Заметьте, что разные наборы данных, сгенерированные на основе одной той же истинной модели, дают несколько различающиеся линии наименьших квадратов, тогда как ненаблюдаемая регрессионная линия генеральной совокупности остается неизменной.

На первый взгляд, разница между регрессионной линией генеральной совокупности и линией наименьших квадратов может показаться несущественной и сбивающей с толку. У нас есть только одна выборка, и поэтому что значат эти две разные линии, описывающие зависимость между

предиктором и откликом? По существу, концепция этих двух линий является естественным продолжением стандартного статистического подхода по использованию информации о выборке для оценивания характеристик большой совокупности. Представьте, например, что мы заинтересованы в нахождении генерального среднего значения μ некоторой случайной переменной Y. К сожалению, μ неизвестно, но все же у нас есть доступ к n наблюдениям Y, которые мы можем записать как $y_1, \ldots, y_n$ и которые мы можем использовать для нахождения оценки μ. Разумной оценкой является $\hat{\mu} = \hat{y}$, где $\hat{y} = \frac{1}{n}\sum_{i=1}^{n}$ — выборочное среднее. Выборочное среднее и среднее генеральной совокупности различаются, но в целом выборочное среднее обеспечит хорошую оценку генерального среднего. Точно таким же образом неизвестные коэффициенты β_0 и β_1 в линейной регрессии определяют регрессионную линию генеральной совокупности. Мы пытаемся оценить эти неизвестные коэффициенты при помощи $\hat{\beta}_0$ и $\hat{\beta}_1$, представленных в (3.4). Эти оценки коэффициентов определяют линию наименьших квадратов.

Аналогия между линейной регрессией и оцениванием среднего значения случайной переменной является уместной благодаря концепции *смещения*. Если мы используем $\hat{\mu}$ для оценивания μ, то эта оценка является *несмещенной*, в том смысле, что в среднем мы ожидаем равенство между $\hat{\mu}$ и μ. Что именно здесь имеется в виду? Речь здесь идет о том, что на основе некоторого конкретного набора наблюдений $y_1, \ldots, y_m$ $\hat{\mu}$ может дать завышенную оценку μ, а на основе другого набора наблюдений $\hat{\mu}$ может дать заниженную оценку μ. Однако мы могли бы усреднить очень большое число оценок μ, полученных на очень большом количестве выборок, и тогда это среднее значение было бы *в точности* равно μ. Следовательно, несмещенная оценка не проявляет систематических отклонений от истинного значения соответствующего параметра. Свойство несмещенности справедливо также для оценок коэффициентов, получаемых по методу наименьших квадратов в соответствии с (3.4): если мы оцениваем β_0 и β_1 на основе какого–то конкретного набора данных, то наши оценки не будут в точности равны β_0 и β_1. Однако мы могли бы усреднить оценки, полученные на очень большом количестве наборов данных, и тогда средние значения этих оценок попали бы точно «в яблочко»! Фактически на рис. 3.3 справа мы можем увидеть, что результат усреднения нескольких линий наименьших квадратов, каждая из которых была оценена на отдельном наборе данных, довольно близок к истинной линии регрессии.

смещение

Мы продолжаем обсуждение аналогии с оценкой генерального среднего значения случайной переменной Y. Естественным будет следующий вопрос: насколько точным является выборочное среднее $\hat{\mu}$ в качестве оценки μ? Мы выяснили, что среднее из $\hat{\mu}$, рассчитанных по большому количеству выборок, будет близко к μ, однако какая–то одна конкретная оценка $\hat{\mu}$ может существенно недооценивать или переоценивать μ. Насколько большим будет отклонение этой единственной оценки $\hat{\mu}$? Как правило, мы отвечаем на этот вопрос путем вычисления *стандартной ошибки* $\hat{\mu}$, которая обозначается как $\mathrm{SE}(\hat{\mu})$. У нас есть хорошо известная формула

стандартная ошибка

$$\mathrm{Var}(\hat{\mu}) = \mathrm{SE}(\hat{\mu})^2 = \frac{\sigma^2}{n}, \tag{3.7}$$

где σ — это стандартное отклонение каждой из реализаций y_i перемен-

ной Y[3]. Грубо говоря, стандартная ошибка показывает нам, насколько в среднем данная оценка $\hat{\mu}$ отличается от истинного значения μ. Уравнение 3.7 говорит нам также, что при увеличении n это отклонение снижается, т. е. чем больше у нас есть наблюдений, тем меньше будет стандартная ошибка $\hat{\mu}$. Точно так же нам может быть интересно, насколько $\hat{\beta}_0$ и $\hat{\beta}_1$ близки к истинным значениям β_0 и β_1. Для вычисления стандартных ошибок $\hat{\beta}_0$ и $\hat{\beta}_1$ мы применяем следующие формулы:

$$\mathrm{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right], \quad \mathrm{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{3.8}$$

где $\sigma^2 = \mathrm{Var}(\epsilon)$. Для того чтобы эти формулы были применимы, мы должны допустить, что остатки ϵ_i всех наблюдений являются некоррелированными и обладают общей дисперсией σ^2. На рис. 3.1 это со всей очевидностью не так, однако формулы все же обеспечивают хорошую аппроксимацию. Заметьте, что $\mathrm{SE}(\hat{\beta}_1)$ в формуле становится меньше при увеличении размаха x_i; интуитивно это объясняется тем, что в таком случае у нас есть больше информации для оценивания угла наклона. Мы видим также, что $\mathrm{SE}(\hat{\beta}_0)$ была бы равна $\mathrm{SE}(\hat{\mu})$ при $\bar{x} = 0$ (в этом случае $\hat{\beta}_0$ был бы равен $\bar{y}$). Как правило, дисперсия σ^2 не известна, но может быть оценена по данным. Эта оценка σ называется *стандартной ошибкой остатков* и вычисляется по формуле $\mathrm{RSE} = \sqrt{\mathrm{RSS}/(n-2)}$[4]. Строго говоря, для того чтобы показать, что при нахождении σ^2 по данным была получена выборочная оценка, нам следует писать $\widehat{\mathrm{SE}}(\hat{\beta}_1)$, однако для простоты обозначений мы отбросим эту дополнительную «крышечку».

стандартная ошибка остатков

Стандартные ошибки можно использовать для вычисления *доверительных интервалов*. Под 95%–ным доверительным интервалом понимают такой диапазон значений, который с вероятностью 95% будет содержать истинное неизвестное значение параметра. Этот диапазон задается в виде нижнего и верхнего доверительных пределов, вычисляемых по выборочным данным. В случае с линейной регрессией 95%–ный доверительный интервал для $\hat{\beta}_1$ принимает следующую приблизительную форму[5]:

доверительный интервал

$$\hat{\beta}_1 \pm 2 \cdot \mathrm{SE}(\hat{\beta}_1). \tag{3.9}$$

Другими словами, имеет место примерно 95%–ная вероятность того, что интервал

$$\left[\hat{\beta}_1 - 2 \cdot \mathrm{SE}(\hat{\beta}_1), \hat{\beta}_1 + 2 \cdot \mathrm{SE}(\hat{\beta}_1) \right] \tag{3.10}$$

[3] Эта формула работает при условии, что n наблюдений не коррелируют друг с другом.

[4] Аббревиатура от «residual standard error». — *Прим. пер.*

[5] *Приблизительность* этого решения обусловлена несколькими причинами. Уравнение 3.10 основано на допущении о том, что остатки имеют гауссово распределение. Кроме того, множитель 2 перед $\mathrm{SE}(\hat{\beta}_1)$ будет несколько варьировать в зависимости от числа наблюдений n в линейной регрессии. Если быть точнее, то вместо числа 2 уравнение 3.10 должно содержать 97.5%–ный квантиль t–распределения с $n-2$ степенями свободы. Детали, касающиеся вычисления 95%–ного доверительного интервала в R, будут приведены позднее в этой главе.

будет содержать истинное значение $\hat{\beta}_1$. Аналогично доверительный интервал для β_0 будет принимать следующую приблизительную форму:

$$\hat{\beta}_0 \pm 2 \cdot \mathrm{SE}(\hat{\beta}_0). \tag{3.11}$$

В случае с данными по рекламе 95%–ный доверительный интервал для β_0 составляет $[6.130, 7.935]$, а 95%–ный доверительный интервал для β_1 составляет $[0.042, 0.053]$. Следовательно, мы можем заключить, что в отсутствие какой–либо рекламы продажи в среднем будут составлять от 6130 до 7940 единиц. При этом каждый раз при увеличении затрат на телерекламу на 1000\$ рост продаж в среднем составит от 42 до 53 единиц.

проверка гипотезы

Стандартные ошибки можно также использовать для *проверки статистических гипотез* в отношении коэффициентов. Наиболее обычной является проверка *нулевой гипотезы*

нулевая гипотеза

$$H_0 : \text{связи между } X \text{ и } Y \text{ нет} \tag{3.12}$$

против *альтернативной гипотезы*

$$H_a : \text{между } X \text{ и } Y \text{ имеется определенная связь.} \tag{3.13}$$

С математической точки зрения это соответствует проверке гипотезы

$$H_0 : \beta_1 = 0$$

против

$$H_a : \beta_1 \neq 0,$$

поскольку если $\beta_1 = 0$, то модель (3.5) сводится к $Y = \beta_0 + \epsilon$ и связи между X и Y нет. Для проверки нулевой гипотезы нам необходимо выяснить, является ли $\hat{\beta}_1$ — наша оценка β_1 — достаточно удаленной от нуля для того, чтобы быть уверенным в ее ненулевом значении. Какое расстояние является достаточным? Это, конечно, зависит от точности $\hat{\beta}_1$, т. е. от $\mathrm{SE}(\hat{\beta}_1)$. Если $\mathrm{SE}(\hat{\beta}_1)$ мала, то даже относительно небольшие $\hat{\beta}_1$ могут обеспечить веское основание в пользу того, что $\beta_1 \neq 0$, а следовательно, и того, что между X и Y имеется определенная связь. В то же время при большой $\mathrm{SE}(\hat{\beta}_1)$ для отклонения нулевой гипотезы абсолютное значение $\hat{\beta}_1$ также должно быть большим. На практике мы вычисляем t–критерий

t–критерий

$$t = \frac{\hat{\beta}_1 - 0}{\mathrm{SE}(\hat{\beta}_1)}, \tag{3.14}$$

который показывает, на сколько стандартных отклонений $\hat{\beta}_1$ отстоит от 0. Если связи между X и Y действительно не существует, то мы ожидаем, что величина (3.14) будет иметь t–распределение с $n-2$ степенями свободы. t–распределение имеет колоколообразную форму, и при $n > 30$ оно довольно похоже на нормальное распределение. Следовательно, допустив, что $\beta_1 = 0$, можно очень просто вычислить вероятность наблюдения любого значения, равного или превышающего $|t|$. Мы называем эту вероятность *p–значением*. Мы интерпретируем p–значение примерно следующим образом: в отсутствие какой–либо реальной связи между предиктором и откликом небольшое p–значение указывает на низкую вероятность

p–значение

случайного обнаружения существенной связи между этими предиктором и откликом. Следовательно, если наблюдается низкое p–значение, то мы можем сделать заключение о существовании определенной связи между предиктором и откликом. При достаточно низком p–значении мы *отклоняем нулевую гипотезу*, т. е. объявляем существование связи между X и Y. Распространенные пороговые p–значения для отклонения нулевой гипотезы составляют 1% и 5%. При $n = 30$ эти значения соответствуют t–критерию из (3.14), равному примерно 2 и 2.75.

ТАБЛИЦА 3.1. *Коэффициенты регрессионной зависимости количества проданных единиц товара от бюджета на телерекламу, рассчитанные по методу наименьших квадратов на основе данных* `Advertising`. *Увеличение затрат на телерекламу на 1000\$ связано с увеличением продаж примерно на 50 единиц (вспомните, что переменная* `sales` *выражается в тысячах единиц, а переменная* `TV` *— в тысячах долларов)*

	Коэффициент	Ст. ошибка	t	p
Свободный член	7.0325	0.4578	15.36	< 0.0001
`TV`	0.0475	0.0027	17.67	< 0.0001

В табл. 3.1 приведены детали регрессионной зависимости количества продаж от размера бюджета на телерекламу, рассчитанной по методу наименьших квадратов для данных `Advertising`. Заметьте, что коэффициенты $\hat{\beta}_0$ и $\hat{\beta}_1$ очень большие, по сравнению с их стандартными ошибками, и поэтому значения t–критерия также высоки; вероятности наблюдения таких значений при верной H_0 фактически равны нулю. Следовательно, мы заключаем, что $\beta_0 \neq 0$ и $\beta_1 \neq 0$[6].

3.1.3 Оценивание точности модели

После отклонения нулевой гипотезы (3.12) в пользу альтернативной гипотезы (3.13) естественным будет желание количественно оценить степень того, *насколько хорошо модель описывает данные*. Качество линейной регрессионной модели обычно оценивается при помощи двух родственных мер: *стандартной ошибки остатков* (RSE) и коэффициента детерминации R^2.

R^2

В табл. 3.2 приведены RSE, статистика R^2 и F–критерий (будет описан в подразделе 3.2.2) для линейной регрессии количества продаж по размеру бюджета на телерекламу.

Стандартная ошибка остатков

Вспомним, что в модели (3.5) каждому наблюдению соответствует остаток ϵ. Из–за наличия этих остатков мы не смогли бы в точности предсказать

[6] В табл. 3.1 низкое p–значение для свободного члена показывает, что мы можем отклонить нулевую гипотезу о $\beta_0 = 0$, а низкое p–значение для `TV` показывает, что мы можем отклонить нулевую гипотезу о $\beta_1 = 0$. Отклонение последней нулевой гипотезы позволяет нам сделать заключение о существовании связи между `TV` и `sales`. Отклонение же первой гипотезы позволяет нам утверждать, что в отсутствие затрат на телерекламу продажи не равны нулю.

Y по X, даже если бы знали истинную регрессионную линию (т. е. если бы β_0 и β_1 были известны). RSE представляет собой оценку стандартного отклонения ϵ. Грубо говоря, это средняя величина отклонения отклика от истинной регрессионной линии. Она рассчитывается по формуле

$$\text{RSE} = \sqrt{\frac{1}{n-2}\text{RSS}} = \sqrt{\frac{1}{n-2}\sum_{i=n}^{n}(y_i - \hat{y}_i)^2}. \tag{3.15}$$

ТАБЛИЦА 3.2. *Дополнительная информация по регрессионной зависимости количества продаж от размера бюджета на телерекламу, рассчитанной по методу наименьших квадратов для данных* `Advertising`

Мера	Значение
Стандартная ошибка остатков	3.26
R^2	0.612
F–критерий	312.1

Заметьте, что RSS была определена в подразделе 3.1.1 и вычисляется по формуле

$$\text{RSS} = \sum_{i=n}^{n}(y_i - \hat{y}_i)^2. \tag{3.16}$$

По результатам регрессионного анализа, представленным в табл. 3.2, мы видим, что для данных по рекламе RSE составляет 3.26. Другими словами, фактические продажи в каждом регионе в среднем отклоняются от истинной регрессионной линии примерно на 3260 единиц. Об этом можно думать также в следующем ключе: даже если бы модель была правильной и истинные значения неизвестных коэффициентов β_0 и β_1 были в точности известны, любое предсказание продаж на основе размера бюджета на телерекламу все равно отклонялось бы в среднем на 3260 единиц. Является ли ошибка предсказания в 3260 единиц приемлемой, зависит, конечно, от контекста проблемы. В данных по рекламе среднее значение `sales` по всем регионам составляет примерно 14 000 единиц, и поэтому удельная ошибка составляет $3260/14000 = 23\%$.

RSE считается мерой *несоответствия* модели (3.5) данным. Если предсказания, полученные с использованием модели, очень близки к истинным значениям отклика, т. е. $\hat{y}_i \approx y_i$ для $i = 1, \ldots, n$, то величина (3.15) будет небольшой, и мы сможем заключить, что модель описывает данные очень хорошо. С другой стороны, если $\hat{y}_i$ существенно отличается от y_i в одном или нескольких случаях, то RSE может оказаться довольно большой, указывая плохое соответствие модели данным.

Коэффициент детерминации R^2

RSE — это абсолютная мера несоответствия модели (3.5) данным. Но поскольку она выражается в тех же единицах измерения, что и Y, то не

всегда бывает ясно, какая RSE является хорошей. Коэффициент детерминации R^2 представляет собой альтернативную меру соответствия. Этот показатель принимает форму доли — доли объясненной дисперсии, в связи с чем он всегда изменяется от 0 до 1 и не зависит от шкалы измерения Y.

Для вычисления R^2 мы применяем следующую формулу:

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}, \tag{3.17}$$

общая сумма квадратов

где $\text{TSS} = \sum(y_i - \bar{y})^2$ — это *общая сумма квадратов*, а определение RSS дано в (3.16). TSS является мерой общей дисперсии отклика Y, и о ней можно думать как о степени изменчивости, присущей отклику до выполнения регрессионного анализа. В то же время RSS выражает степень изменчивости, которая осталась необъясненной после построения регрессионной модели. Следовательно, TSS − RSS выражает количество дисперсии отклика, объясненное («изъятое») после выполнения регрессионного анализа, а R^2 — *долю дисперсии Y, объясненную при помощи X*. Статистика R^2, близкая к 1, указывает на то, что значительная доля изменчивости отклика была объяснена регрессионной моделью. Число, близкое к 0, указывает на то, что регрессионная модель объяснила лишь незначительную долю изменчивости отклика; это может иметь место либо в силу того, что линейная модель неверна, либо в связи с естественно высокой дисперсией остатков σ^2, либо по обеим причинам одновременно. В табл. 3.2 $R^2 = 0.61$, и значит, линейная регрессия по `TV` объясняет немногим меньше двух третей изменчивости `sales`.

Статистика R^2 из (3.17) имеет преимущество над RSE из (3.15) с точки зрения легкости интерпретации, поскольку, в отличие от RSE, она всегда изменяется от 0 до 1. Однако определение *хорошего значения* R^2 может вызвать затруднения и, как правило, зависит от конкретной задачи. Например, в некоторых задачах физики нам может быть известно, что данные были действительно порождены линейной моделью с небольшой дисперсией остатков. В этом случае мы ожидали бы значение R^2, очень близкое к 1, а существенно меньшее значение R^2 могло бы указывать на серьезную проблему с экспериментом, в котором были получены соответствующие данные. С другой стороны, в типичных задачах биологии, психологии, маркетинга и других областей линейная модель (3.5) в лучшем случае является очень грубой аппроксимацией данных, а остатки часто бывают очень большими в связи с неучтенными факторами. При таком сценарии мы ожидали бы лишь небольшую долю дисперсии отклика, объясненную предиктором, и значение $R^2 = 0.1$ было бы более реалистичным.

корреляция

Статистика R^2 является мерой линейной зависимости между X и Y. Вспомните, что корреляция, определяемая как

$$\text{Cor}(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{3.18}$$

также является мерой линейной зависимости между X и Y[7]. Это подра-

[7] Отметим, что в действительности правая часть выражения (3.18) представляет собой выборочный коэффициент корреляции; следовательно, более корректным способом записи был бы $\widehat{\text{Cor}}(X, Y)$, однако мы опускаем «крышечку» для простоты обозначений.

зумевает, что для оценки степени соответствия линейной модели данным мы могли бы использовать $r = \mathrm{Cor}(X, Y)$, а не R^2. Действительно, можно показать, что в случае с простой линейной регрессией $R^2 = r^2$. Другими словами, квадрат коэффициента корреляции и коэффициент детерминации R^2 идентичны. Однако в следующем разделе мы будем обсуждать проблему множественной линейной регрессии, в которой для предсказания отклика мы одновременно используем несколько предикторов. Идея корреляции между предикторами и откликом на такой сценарий автоматически не переносится, поскольку корреляция выражает силу связи только между парой переменных, а не несколькими переменными. Мы увидим, что эту роль выполняет R^2.

3.2 Множественная линейная регрессия

Простая линейная регрессия является полезным методом для предсказания некоторого отклика на основе единственной независимой переменной. Однако на практике у нас часто есть несколько предикторов. Например, в случае с данными `Advertising` мы изучили связь между количеством продаж и телерекламой. У нас имеются данные также по затратам на рекламу на радио и в газетах, и мы можем быть заинтересованы в установлении связи между количеством продаж и рекламой в каком–либо из этих двух типов СМИ. Как нам расширить наш анализ данных по рекламе, чтобы учесть эти два дополнительных предиктора?

Один из способов заключается в расчете трех отдельных простых регрессионных моделей, каждая из которых в качестве предиктора использует затраты на рекламу в соответствующем СМИ. Например, мы можем подогнать простую линейную регрессию для предсказания продаж на основе затрат на радиорекламу. Результаты представлены в табл. 3.3 (верхняя часть). Мы видим, что увеличение затрат на радиорекламу на 1000\$ связано с увеличением продаж примерно на 203 единицы. Нижняя часть табл. 3.3 содержит коэффициенты простой линейной регрессии количества продаж по размеру бюджета на рекламу в газетах. Увеличение затрат на рекламу в газетах на 1000\$ связано с увеличением продаж примерно на 55 единиц.

Однако метод подгонки простой линейной регрессионной модели отдельно для каждого предиктора не является в полной мере удовлетворительным. Прежде всего не ясно, как сделать единый прогноз продаж при заданных размерах бюджета на рекламу в каждом из трех типов СМИ, поскольку каждому из этих размеров соответствует отдельное регрессионное уравнение. Во–вторых, каждое из трех регрессионных уравнений игнорирует два других вида СМИ при расчете оценок регрессионных коэффициентов. Вскоре мы увидим, что если затраты на рекламу в разных СМИ в тех 200 регионах, которые составляют наш набор данных, коррелируют друг с другом, то это может привести к очень обманчивым оценкам эффектов отдельных СМИ на продажи.

Вместо подгонки отдельной регрессионной модели для каждого предиктора более подходящим методом является расширение простой линейной регрессионной модели (3.5) таким образом, чтобы непосредственно учесть наличие нескольких предикторов. Для этого мы можем присвоить

ТАБЛИЦА 3.3. *Дополнительные простые линейные регрессионные модели для данных* Advertising. Вверху: *коэффициенты простой линейной регрессии количества продаж по размеру бюджета на радиорекламу.* Внизу: *то же, но для рекламы в газетах. Увеличение затрат на радиорекламу на 1000$ связано с увеличением продаж в среднем примерно на 203 единицы, тогда как аналогичное увеличение затрат на рекламу в газетах связано с увеличением продаж в среднем примерно на 55 единиц (заметьте, что переменная* sales *выражается в тысячах единиц, а переменные* radio *и* newspaper *— в тысячах долларов)*

Простая линейная регрессия sales по radio

	Коэффициент	Ст. ошибка	t	p
Свободный член	9.312	0.563	16.54	< 0.0001
radio	0.203	0.020	9.92	< 0.0001

Простая линейная регрессия sales по newspaper

	Коэффициент	Ст. ошибка	t	p
Свободный член	12.351	0.621	19.88	< 0.0001
newspaper	0.055	0.017	3.30	< 0.0001

каждому предиктору свой угловой коэффициент в рамках одной модели. Допустим, что в общем виде у нас имеется p отдельных предикторов. Тогда множественная регрессионная модель принимает форму

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon, \tag{3.19}$$

где X_j — это j–й предиктор, а β_j выражает силу связи между соответствующей переменной и откликом. Мы интерпретируем β_j как *среднее* изменение Y, вызванное увеличением X_j на одну единицу при *удержании всех остальных предикторов на фиксированном уровне.* Для примера с рекламой уравнение (3.19) превращается в

$$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{newspaper} + \epsilon. \tag{3.20}$$

3.2.1 Оценивание регрессионных коэффициентов

Подобно тому, как это было в случае с простой линейной регрессией, регрессионные коэффициенты $\beta_0, \beta_1, \ldots, \beta_p$ в (3.19) неизвестны и подлежат оцениванию. Имея оценки $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$, мы можем сделать предсказания при помощи формулы

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p. \tag{3.21}$$

Параметры оцениваются с использованием того же метода наименьших квадратов, который мы видели в контексте простой линейной регрессии. Мы выбираем $\beta_0, \beta_1, \ldots, \beta_p$ с целью минимизации суммы квадратов остатков

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2. \quad (3.22)$$

Значения $\beta_0, \beta_1, \ldots, \beta_p$, которые минимизируют (3.22), представляют собой оценки коэффициентов множественной регрессии по методу наименьших квадратов. В отличие от оценок простой линейной регрессии (3.4), оценки коэффициентов множественной регрессии имеют довольно сложные формы, которые проще всего представить с использованием матричной алгебры. По этой причине мы их здесь не приводим. Для вычисления этих оценок можно использовать любой статистический пакет, и позже в этой главе мы покажем, как это сделать в R. Рисунок 3.4 иллюстрирует пример подгонки модели наименьших квадратов к гипотетическим данным с $p = 2$ предикторами.

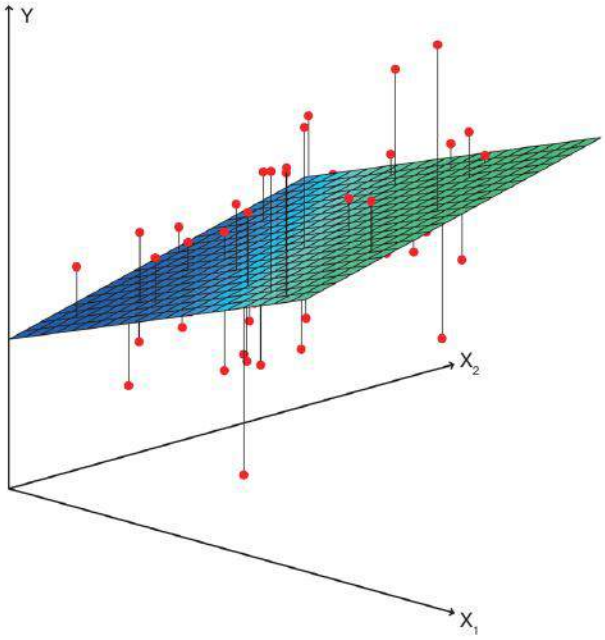

РИСУНОК 3.4. *В трехмерном пространстве, представленном двумя предикторами и одной независимой переменной, линия наименьших квадратов становится плоскостью. Эта плоскость выбирается путем минимизации суммы возведенных в квадрат вертикальных расстояний от каждого наблюдения (показаны красными точками) до плоскости*

В табл. 3.4 приведены оценки коэффициентов множественной регрессионной модели для данных Advertising, в которой затраты на рекламу на телевидении, радио и в газетах используются для предсказания количества продаж. Мы интерпретируем эти результаты следующим образом: при некотором фиксированном бюджете на рекламу на телевидении

и в газетах трата дополнительных 1000$ на радиорекламу приводит к увеличению продаж примерно на 189 единиц. Сравнивая эти коэффициенты с таковыми в табл. 3.1 и 3.3, мы можем заметить, что оценки коэффициентов множественной регрессии для TV и radio довольно похожи на оценки коэффициентов простой линейной регрессии. Однако, несмотря на то что оценка регрессионного коэффициента для newspaper в табл. 3.3 значительно отличалась от нуля, оценка коэффициента newspaper в модели множественной регрессии оказалась близка к нулю, а соответствующее p–значение, примерно равное 0.86, перестало быть значимым. Данный пример показывает, что коэффициенты простой и множественной регрессий могут быть довольно разными. Эта разница обусловлена тем фактом, что в простой регрессии угловой коэффициент отражает средний эффект увеличения затрат на рекламу в газетах на 1000$ без учета эффектов других предикторов (TV и radio). В случае же со множественной регрессией коэффициент newspaper соответствует среднему эффекту увеличения затрат на рекламу в газетах на 1000$ при фиксированных уровнях TV и radio.

ТАБЛИЦА 3.4. *Коэффициенты множественной регрессии количества продаж по размеру затрат на рекламу на телевидении, радио и в газетах, рассчитанные по методу наименьших квадратов на основе данных* Advertising

	Коэффициент	Ст. ошибка	t	p
Свободный член	2.939	0.3119	9.42	< 0.0001
TV	0.046	0.0014	32.81	< 0.0001
radio	0.189	0.0086	21.89	< 0.0001
newspaper	−0.001	0.0059	−0.18	< 0.8599

Имеет ли смысл тот факт, что множественная регрессия указывает на отсутствие связи между sales и newspaper, тогда как простая линейная регрессия предполагает обратное? Оказывается, что имеет. Обратите внимание на корреляционную матрицу для трех предикторов и независимой переменной, представленную в табл. 3.5. Заметьте, что корреляция между radio и newspaper составляет 0.35. Это указывает на наличие повышенных затрат на рекламу в газетах в тех регионах, где много тратится на радиорекламу. Теперь представьте, что множественная регрессия является корректной и реклама в газетах не имеет прямого влияния на продажи, тогда как реклама на радио ведет к их росту. Тогда в регионах, где мы тратим больше на радиорекламу, наши продажи в целом будут выше, и, согласно нашей корреляционной матрице, в тех же регионах в целом мы также будем тратить больше на рекламу в газетах. Следовательно, в простой линейной регрессии, которая описывает связь только между sales и newspaper, мы будем наблюдать более высокие значения sales при высоких значениях newspaper, даже если реклама в газетах в действительности не оказывает влияния на продажи. Таким образом, продажи, связанные с рекламой в газетах, являются суррогатом продаж, вызванных рекламой на радио, — переменная newspaper как бы получает «кредит» за счет влияния radio на sales.

Этот несколько неожиданный результат часто встречается во многих ситуациях из реальной жизни. Представьте себе следующий абсурдный

ТАБЛИЦА 3.5. *Корреляционная матрица для переменных* TV, radio, newspaper *и* sales *из набора данных* Advertising

	TV	radio	newspaper	sales
TV	1.0000	0.0548	0.0567	0.7822
radio		1.0000	0.3541	0.5762
newspaper			1.0000	0.2283
sales				1.0000

пример, иллюстрирующий суть этого явления. Регрессия числа нападений акул на купающихся по количеству продаж мороженого на основе данных, собранных в течение некоторого времени для определенного пляжа, показала бы наличие положительной связи, подобной той, что наблюдается между sales и newspaper. Конечно, никто (пока) не заявил, что для снижения числа нападений акул нужно запретить продавать мороженое на пляжах. В действительности более высокие температуры заставляют большее количество людей посещать пляж, что в итоге приводит к более высоким продажам мороженого и более частым нападениям акул. Множественная регрессия числа атак по количеству продаж мороженого и температуре ожидаемо показывает, что с поправкой на температуру уровень продаж мороженого перестает быть значимым предиктором.

3.2.2 Некоторые важные вопросы

При выполнении множественной линейной регрессии мы обычно заинтересованы в ответе на несколько важных вопросов.

1. *Является ли хотя бы один из предикторов* $X_1, X_2, \ldots, X_p$ *полезным для предсказания отклика?*

2. *Все ли предикторы помогают объяснить* Y*, или полезным является только некоторое подмножество предикторов?*

3. *Насколько хорошо модель описывает данные?*

4. *Имея некоторый набор значений предикторов, как нам предсказать отклик, и насколько точным будет наше предсказание?*

Теперь мы рассмотрим каждый из этих вопросов по порядку.

Вопрос 1: Имеется ли связь между откликом и предикторами?

Вспомните, что в случае с простой линейной регрессией для установления существования связи между откликом и предиктором мы можем просто проверить равенство β_1 нулю. В случае множественной регрессии с p предикторами нам необходимо выяснить, равны ли нулю все регрессионные коэффициенты, т. е. $\beta_1 = \beta_2 = \cdots = \beta_p = 0$. Как и в простой линейной регрессии, для ответа на этот вопрос мы выполняем проверку статистической гипотезы. Мы проверяем нулевую гипотезу

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

против

$$H_a : \text{как минимум один } \beta_j \text{ не равен нулю.}$$

F–критерий

Проверка этой гипотезы выполняется путем вычисления *F–критерия*

$$F = \frac{(\text{TSS} - \text{RSS})/p}{\text{RSS}/(n-p-1)}, \tag{3.23}$$

где, как и в простой линейной регрессии, $\text{TSS} = \sum(y_i - \bar{y})^2$, а $\text{RSS} = \sum(y_i - \hat{y})^2$. Если предположения в отношении линейной модели справедливы, то можно показать, что

$$E\{\text{RSS}/(n-p-1)\} = \sigma^2,$$

а также (при справедливой H_0), что

$$E\{(\text{TSS} - \text{RSS})/p\} = \sigma^2.$$

Следовательно, в отсутствие связи между откликом и предикторами можно было бы ожидать, что F–критерий примет значение, близкое к 1. С другой стороны, если верна H_a, то $E\{(\text{TSS} - \text{RSS})/p\} > \sigma^2$, и мы ожидаем $F > 1$.

ТАБЛИЦА 3.6. *Дополнительная информация по регрессии количества продаж от затрат на рекламу на телевидении, радио и в газетах, рассчитанной по методу наименьших квадратов для данных* `Advertising`. *Другая информация об этой модели была приведена в табл. 3.4*

Мера	Значение
Стандартная ошибка остатков	1.69
R^2	0.897
F–критерий	570

F–критерий для множественной линейной регрессии, описывающей зависимость `sales` от `radio`, `TV` и `newspaper`, приведен в табл. 3.6. В этом примере F–критерий составляет 570. Поскольку это значение намного больше 1, оно является убедительным свидетельством против нулевой гипотезы H_0. Другими словами, большой F–критерий указывает на то, что как минимум одно средство рекламы должно быть связано с продажами. Но что, если бы F–критерий был ближе к 1? Насколько большим должно быть значение F, чтобы мы смогли отклонить H_0 и сделать заключение о существовании зависимости? Оказывается, что ответ зависит от значений n и p. При большом n значение F–критерия, ненамного превышающее 1, уже могло бы служить достаточным свидетельством против H_0. В то же время при небольшом n для отклонения H_0 требуется более высокий F–критерий. Если H_0 верна и остатки ϵ_i имеют нормальное распределение, то

F–критерий подчиняется F–распределению[8]. Имея некоторые значения n и p, можно воспользоваться любым статистическим пакетом для вычисления по этому распределению p–значения, связанного с соответствующим F–критерием. На основе этого p–значения мы можем определить, нужно ли отклонять H_0. Для данных по рекламе p–значение F–критерия в сущности равно нулю (табл. 3.6), и поэтому у нас есть очень убедительное свидетельство в пользу того, что как минимум одно СМИ связано с увеличением продаж.

В (3.23) мы проверяем H_0 о том, что нулю равны все коэффициенты. Иногда мы хотим проверить равенство нулю лишь некоторого подмножества, образованного q коэффициентами. Это соответствует нулевой гипотезе

$$H_0 = \beta_{p-q+1} = \beta_{p-q+2} = \cdots = \beta_p = 0,$$

где для удобства мы поместили подлежащие исключению переменные в конце списка. В этом случае мы подгоняем вторую модель, которая использует все переменные, *за исключением* этих последних q переменных. Пусть сумма квадратов остатков для такой модели обозначается как RSS_0. Тогда подходящим F–критерием будет

$$F = \frac{(\mathrm{RSS}_0 - \mathrm{RSS})/q}{\mathrm{RSS}/(n-p-1)}. \tag{3.24}$$

Заметьте, что в табл. 3.4 для каждого предиктора были приведены отдельные t–критерий и p–значение. Они предоставляют информацию о том, связан ли каждый предиктор с откликом в присутствии других предикторов. Оказывается, что каждый из этих t–критериев эквивалентен[9] F–критерию, который рассчитывается после удаления соответствующей переменной из модели, т. е. $q = 1$ в (3.24). Таким образом, каждый t–критерий характеризует *частный эффект* от добавления соответствующей переменной в модель. Например, как мы обсуждали ранее, эти p–значения говорят о том, что `TV` и `radio` связаны с `sales`, а также о том, что в присутствии двух других предикторов указаний на связь между `newspaper` и `sales` нет.

Имея эти отдельные p–значения для каждой переменной, зачем нам вообще смотреть на общий F–критерий? В конце концов, вполне вероятно, что если какое–либо из p–значений является очень низким, то *как минимум один из предикторов связан с откликом.* Однако это ложная логика, особенно при наличии большого числа предикторов p.

Представьте себе пример, в котором $p = 100$ и $H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$ верна, т. е. ни одна из переменных в действительности не связана с откликом. В этой ситуации примерно у 5% переменных p–значения (вроде тех, которые представлены в табл. 3.4) будут ниже 0.05 просто в силу случая. Другими словами, мы ожидаем увидеть примерно пять *низких* p–значений даже в отсутствие какой–либо истинной связи между предикторами и откликом. На самом деле мы почти гарантированно обнаружим

[8] Даже если остатки имеют распределение, отличное от нормального, при наличии большого числа наблюдений F–критерий приближенно будет подчиняться F–распределению.

[9] Квадрат каждого t–критерия представляет собой соответствующий F–критерий.

как минимум одно p–значение ниже 0.05, возникшее по чистой случайности. Следовательно, если мы будем применять отдельные t–критерии и связанные с ними p–значения для принятия решения о наличии взаимосвязи между переменными и откликом, то столкнемся с очень большой вероятностью сделать неправильное заключение. Однако F–критерий не страдает от этой проблемы, поскольку он делает поправку на количество предикторов. Следовательно, при верной H_0 есть лишь 5%–ная вероятность того, что F–критерий будет сопровождаться p–значением, меньшим 0.05, вне зависимости от количества предикторов или объема наблюдений.

Использование F–критерия для проверки наличия какой–либо связи между предикторами и откликом работает при относительно небольшом p, и конечно небольшом, по сравнению с n. Однако иногда у нас есть большое количество переменных. При $p > n$ число подлежащих оцениванию коэффициентов β_j превышает число имеющихся для этого наблюдений. В этом случае мы даже не сможем подогнать линейную регрессионную модель по методу наименьших квадратов, и поэтому невозможно будет использовать ни F–критерий, ни большинство других концепций, с которыми мы столкнулись в данной главе. При большом p можно применять некоторые подходы, обсуждаемые в следующем подразделе, например *пошаговый отбор с включением переменных*. Этот *многомерный* сценарий обсуждается более подробно в главе 6.

Вопрос 2: Отбор важных переменных

Как обсуждалось в предыдущем подразделе, первым шагом множественного регрессионного анализа является вычисление F–критерия и изучение соответствующего p–значения. Если на основе этого p–значения мы делаем вывод о том, что как минимум один из предикторов связан с откликом, то естественным будет желание выяснить, *какие именно* предикторы «виноваты» в этом. Мы могли бы исследовать отдельные p–значения вроде тех, которые представлены в табл. 3.4, но, как было отмечено ранее, при большом p велика вероятность, что мы сделаем неверные заключения.

Наличие связи между всеми предикторами и откликом возможно, однако чаще встречаются ситуации, когда отклик связан только с некоторым подмножеством предикторов. Задачу по нахождению предикторов, связанных с откликом, с целью последующей подгонки единой модели, включающей только такие предикторы, называют *отбором информативных переменных*. Проблема отбора информативных переменных детально исследуется в главе 6, и поэтому здесь мы приведем только краткое описание некоторых классических подходов.

отбор информативных переменных

В идеале мы хотели бы выполнить отбор переменных путем перебора большого количества различных моделей, каждая из которых включает свой набор предикторов. Например, при $p = 2$ мы можем рассмотреть четыре модели: (1) модель без переменных, (2) модель, содержащую лишь X_1, (3) модель, содержащую только X_2, и (4) модель, содержащую X_1 и X_2. Далее мы можем выбрать *оптимальную* из всех рассмотренных моделей. Как мы определяем, какая из моделей является оптимальной? Для суждения о качестве модели можно использовать разные критерии. Они включают C_p *Мэллоу*, *информационный критерий Акаике* (AIC), *байесовский информационный критерий* (BIC), а также *скорректированный*

C_p Мэллоу

AIC

BIC

коэффициент детерминации R^2. Эти критерии детально обсуждаются в главе 6. Кроме того, мы можем определить оптимальную модель путем визуализации результатов моделирования (например, остатков) для обнаружения соответствующих закономерностей.

скорректированный R^2

К сожалению, существует 2^p моделей, содержащих подмножества p переменных. Это означает, что даже для умеренных p перебор всех возможных подмножеств предикторов невыполним. Например, мы видели, что при $p = 2$ рассмотрению подлежат $2^2 = 4$ модели. Однако при $p = 30$ нам придется рассмотреть $2^{30} = 1\,073\,741\,824$ моделей! Это непрактично. Следовательно, за исключением случаев с очень небольшим p, мы не можем рассмотреть все 2^p моделей, и вместо этого нам нужен какой–то автоматизированный и эффективный способ для выбора меньшего набора подлежащих рассмотрению моделей. Для решения этой задачи существует три классических метода:

- *Отбор с включением*. Мы начинаем с *нулевой модели* — модели, которая включает свободный член, но не содержит предикторов. Далее мы подгоняем p простых линейных регрессий и добавляем к нулевой модели ту переменную, которая приводит к наименьшей RSS. Затем мы добавляем ту переменную, которая приводит к наименьшей RSS для новой модели с двумя переменными. Этот процесс продолжается до удовлетворения некоторого правила остановки.

- *Отбор с исключением*. Мы начинаем с модели, включающей все переменные, и исключаем из нее переменную с наибольшим p–значением, т. е. наименее значимую переменную. Далее подгоняется новая модель с $(p - 1)$ переменными, и из нее удаляется переменная с наибольшим p–значением. Этот процесс продолжается до удовлетворения некоторого правила остановки. Например, мы можем остановиться, когда p–значения у всех переменных оказываются ниже определенного порога.

- *Комбинированный отбор*. Этот метод является сочетанием отбора с включением и отбора с исключением. Мы начинаем с нулевой модели и, как при отборе с включением, добавляем переменную, обеспечивающую наилучшее качество. Далее мы продолжаем добавлять переменные одну за другой. Конечно, как мы уже видели в примере с данными `Advertising`, по мере добавления новых предикторов в модель некоторые p–значения могут становиться большими. Следовательно, если в какой–то момент p–значение какой–то из переменных в модели начинает превышать некоторый порог, мы удаляем эту переменную из модели. Мы продолжаем выполнять эти шаги вперед и назад до тех пор, пока p–значения у всех переменных в модели не оказываются достаточно низкими, а p–значения у исключенных переменных — высокими при добавлении в модель.

Отбор с исключением неприменим при $p > n$, тогда как отбор с включением применим в любых ситуациях. Отбор с включением является жадным методом и может на ранних этапах добавить переменные, которые позднее становятся излишними. Комбинированный отбор может это исправить.

Вопрос 3: Качество модели

Двумя наиболее распространенными количественными мерами качества модели являются RSE и R^2 (доля объясненной дисперсии). Эти показатели вычисляются и интерпретируются тем же образом, как и для простой линейной регрессии.

Вспомните, что в простой линейной регрессии коэффициент детерминации R^2 равен квадрату корреляции между откликом и независимой переменной. Оказывается, что в случае множественной линейной регрессии он равен $\mathrm{Cor}(Y, \hat{Y})^2$, т. е. квадрату корреляции между откликом и его модельным значением; по сути, одним из свойств подогнанной линейной модели является то, что из всех возможных моделей она максимизирует эту корреляцию.

Значение R^2, близкое к 1, указывает на то, что модель объясняет значительную часть дисперсии независимой переменной. Например, в табл. 3.6 мы видели, что для данных `Advertising` модель, предсказывающая продажи на основе всех трех типов рекламы, имела $R^2 = 0.8972$. С другой стороны, у модели, которая использует для предсказания `sales` только `TV` и `radio`, R^2 составляет 0.89719. Другими словами, имеет место *небольшое* увеличение R^2 при добавлении `newspaper` в модель, которая уже содержит `TV` и `radio`, хотя раньше в табл. 3.4 мы видели, что p–значение для `newspaper` было незначимым. Оказывается, что R^2 всегда будет возрастать при добавлении в модель дополнительных переменных, даже если эти переменные очень слабо связаны с откликом. Это обусловлено тем фактом, что добавление новой переменной в формулу наименьших квадратов обязано приводить к более точному описанию обучающих данных (хотя это не обязательно так для контрольной выборки). Поэтому статистика R^2, также вычисленная по обучающим данным, обязана возрастать. То обстоятельство, что добавление газетной рекламы в модель, включающую рекламу на телевидении и радио, вызывает лишь незначительное увеличение R^2, является дополнительным свидетельством в пользу того, что `newspaper` можно исключить из модели. В сущности, `newspaper` не вызывает реального улучшения качества модели, подогнанной по обучающей выборке, и включение этой переменной может привести к плохим результатам на независимых контрольных выборках в силу переобучения.

В то же время модель, содержащая в качестве предиктора только `TV`, имела $R^2 = 0.61$ (табл. 3.2). Добавление переменной `radio` в эту модель приводит к существенному улучшению R^2. Это предполагает, что модель, которая предсказывает продажи на основе рекламы на телевидении и радио, гораздо лучше модели, использующей только телевизионную рекламу. Мы могли бы также охарактеризовать это улучшение количественно, исследовав p–значение коэффициента для `radio` в модели, которая содержит в качестве предикторов только `TV` и `radio`.

RSE у модели, содержащей в качестве предикторов только `TV` и `radio`, составляет 1.681, а у модели, которая включает также `newspaper`, RSE = 1.686 (табл. 3.6). В то же время модель, включающая только `TV`, имеет RSE = 3.26 (табл. 3.2). Это подтверждает наше предыдущее заключение о том, что модель, использующая для предсказания продаж затраты на рекламу на телевидении и радио, является гораздо более точной (на обучающей выборке), чем та, которая использует только затраты на телере-

кламу. Более того, при условии, что в качестве предикторов используются затраты на рекламу на телевидении и радио, нет смысла в добавлении в качестве предиктора также затрат на рекламу в газетах. Внимательный читатель может задаться вопросом о том, как при добавлении в модель переменной `newspaper` RSE может увеличиться, учитывая, что RSS обязана снижаться? В общем виде RSE вычисляется по формуле

$$\text{RSE} = \sqrt{\frac{1}{n-p-1}\text{RSS}}, \tag{3.25}$$

которая упрощается до (3.15) в случае простой линейной регрессии. Таким образом, модели с большим количеством переменных могут иметь более высокую RSE, если снижение RSS мало, по сравнению с увеличением p.

Помимо анализа рассмотренных только что RSE и R^2, полезной может оказаться также визуализация данных. Графические сводки могут выявить проблемы с моделью, незаметные по количественным показателям. Например, на рис. 3.5 представлен трехмерный график зависимости `sales` от `TV` и `radio`. Мы видим, что некоторые наблюдения лежат выше, а некоторые — ниже плоскости наименьших квадратов. Похоже, в частности, что линейная модель переоценивает `sales` в случаях, когда большая часть рекламного бюджета была потрачена только на телевидение или только на радио. При этом она недооценивает `sales` в случаях, когда бюджет был разделен между этими двумя СМИ. Такую выраженную нелинейную зависимость невозможно смоделировать с высокой точностью при помощи линейной регрессии. Данная зависимость предполагает наличие *синергии*, или эффекта *взаимодействия* между разными типами рекламы, когда определенное сочетание СМИ приводит к большему росту продаж, нежели использование любого из них по отдельности. В подразделе 3.3.2 мы обсудим расширение линейной модели, позволяющее учесть подобные синергичные эффекты путем добавления новых переменных, описывающих взаимодействия.

Вопрос 4: Предсказания

Подогнав множественную регрессионную модель, можно легко применить (3.21) для предсказания значения отклика Y по набору значений предикторов $X_1, X_2, \ldots, X_p$. Однако имеют место три вида неопределенности в отношении такого предсказания:

1. Коэффициенты $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ являются оценками $\beta_0, \beta_1, \ldots, \beta_p$. Иными словами, *плоскость наименьших квадратов*

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \ldots \hat{\beta}_p X_p$$

является лишь оценкой *истинной регрессионной плоскости*

$$f(X) = \beta_0 + \beta_1 X_1 + \ldots \beta_p X_p.$$

Неточность в оценках коэффициентов есть проявление *устранимой ошибки* (глава 2). Мы можем вычислить доверительный интервал с целью определения того, насколько близка оценка $\hat{Y}$ к $f(X)$.

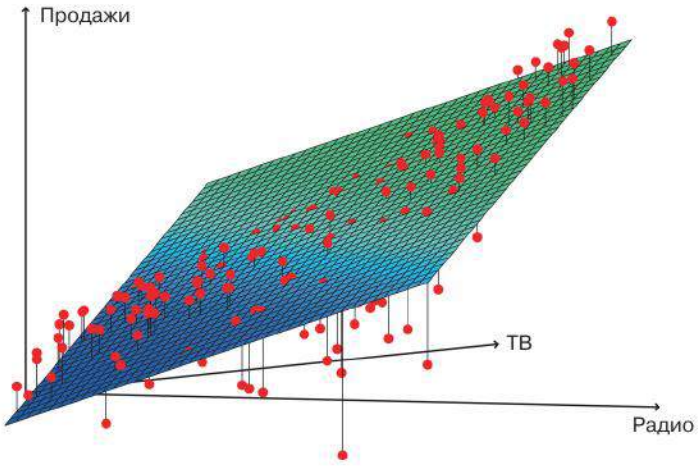

РИСУНОК 3.5. *Линейная зависимость* `sales` *от* `TV` *и* `radio`*, подогнанная по данным* `Advertising`*. По характеру расположения остатков видна четко выраженная нелинейная зависимость. Положительные остатки (располагаются над плоскостью) преимущественно выстраиваются вдоль линии, проходящей под углом 45°, где затраты на теле– и радиорекламу распределены равномерно. При неравномерном распределении бюджетов отрицательные остатки (большинство из них не видны) в основном отклоняются от этой линии*

2. Конечно, на практике предположение о линейной форме модели для $f(X)$ почти всегда является упрощением реальности, в связи с чем имеет место дополнительный источник потенциально устранимой ошибки, который мы называем *смещением модели*. Поэтому, применяя линейную модель, мы фактически оцениваем наилучшую линейную аппроксимацию истинной регрессионной плоскости. Однако здесь мы проигнорируем это несоответствие и будем действовать, как будто линейная модель верна.

3. Даже если бы мы знали $f(X)$, т. е. если бы нам были известны истинные значения $\beta_0, \beta_1, \ldots, \beta_p$, зависимую переменную невозможно было бы предсказать с абсолютной точностью из–за наличия в модели (3.21) случайной ошибки ϵ. В главе 2 мы называли это *неустранимой ошибкой*. Насколько большим будет отклонение Y от $\hat{Y}$? Для ответа на этот вопрос мы применяем *интервалы предсказания*. Интервалы предсказания всегда шире доверительных интервалов, поскольку они включают как (устранимую) ошибку оценки $f(X)$, так и неопределенность в отношении того, насколько отдельная точка будет отличаться от истинной регрессионной плоскости (неустранимая ошибка).

доверительный интервал

Мы применяем *доверительный интервал*, чтобы количественно выразить неопределенность в отношении *среднего* объема продаж в большом числе городов. Например, при затрате в каждом городе 100 000$ на те-

лерекламу и 20 000$ на радиорекламу, 95%–ный доверительный интервал составит [10 985, 11 528]. В 95% случаев интервалы данного типа будут содержать истинное значение $f(X)$.[10] С другой стороны, *интервал предсказания* можно использовать для количественного описания неопределенности в отношении объема продаж в *конкретном* городе. При затрате в некотором городе 100 000$ на телерекламу и 20 000$ на радиорекламу, 95%–ный интервал предсказания составит [7930, 14 580]. В 95% случаев интервалы этого типа будут содержать истинное значение Y для данного города. Заметьте, что интервалы обоих типов центрированы относительно значения 11 256, однако интервал предсказания значительно шире доверительного интервала, что отражает повышенную неопределенность в отношении объема продаж в конкретном городе, по сравнению со средним уровнем продаж в большом количестве городов.

интервал предсказания

3.3 Другие аспекты регрессионной модели

3.3.1 Качественные предикторы

До сих пор мы полагали, что все переменные в нашей линейной регрессионной модели являются *количественными*. Однако на практике это не всегда так — часто некоторые предикторы являются качественными.

Например, набор данных `Credit`, изображенный на рис. 3.6, содержит данные по среднему долгу по кредитной карте (`balance`) для ряда клиентов банка, а также несколько количественных предикторов: `age` (возраст), `cards` (количество карт), `education` (количество лет, потраченных на образование), `income` (доход, тыс. долларов), `limit` (кредитный лимит) и `rating` (кредитный рейтинг). Каждый график на рис. 3.6 представляет собой диаграмму рассеяния для пары переменных, чьи названия указаны в метках соответствующих строк и столбцов. Например, диаграмма, расположенная сразу справа от слова «Баланс», описывает связь между `balance` и `age`, тогда как диаграмма справа от слова «Возраст» соответствует связи между `age` и `cards`. Помимо этих количественных переменных, у нас есть также четыре качественные переменные: `gender` (пол), `student` (является ли клиент студентом), `status` (состоит ли клиент в браке) и `ethnicity` (белый, афроамериканец или азиат).

Предикторы, имеющие только два уровня

Представим, что мы хотим выяснить различия по балансу на кредитной карте между мужчинами (`Male`) и женщинами (`Female`), игнорируя пока другие переменные. Если качественный предиктор (известный также как «*фактор*») имеет только два *уровня*, или возможных значения, то добавить его в регрессионную модель очень просто. Мы просто создаем *индикаторную*, или *фиктивную*, переменную, которая принимает только два возможных количественных значения. Например, на основе переменной `gender` (пол) мы можем создать новую переменную вида

фактор

уровень

индикаторная переменная

[10] Другими словами, если мы получим много наборов данных, аналогичных `Advertising`, и построим доверительный интервал для среднего значения `sales` на основе каждого из этих наборов (затрачивая 100 000$ на `TV`– и 20 000$ на `radio`–рекламу), то 95% полученных доверительных интервалов будут включать истинное значение `sales`.

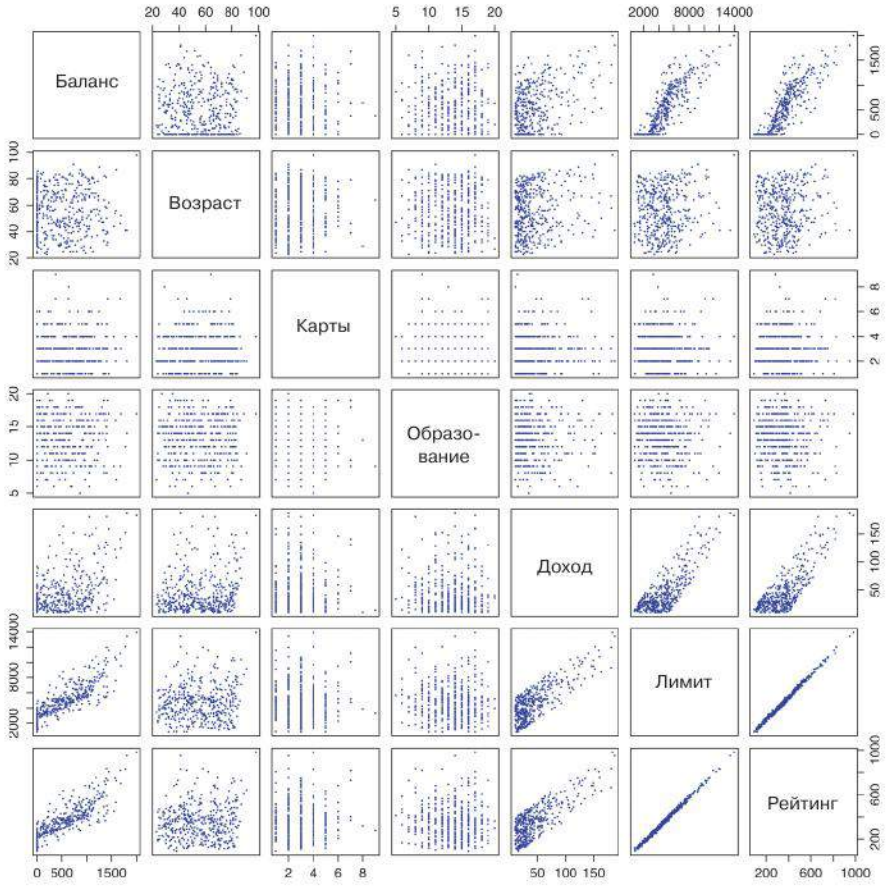

РИСУНОК 3.6. *Набор данных* `Credit` *содержит переменные* `balance`, `age`, `cards`, `education`, `income`, `limit` *и* `rating` *для нескольких потенциальных клиентов банка*

$$x_i = \begin{cases} 1, & \text{если } i\text{–й клиент — женщина} \\ 0, & \text{если } i\text{–й клиент — мужчина} \end{cases} \tag{3.26}$$

и использовать эту переменную в качестве предиктора в уравнении регрессии. Это приводит к следующей модели:

$$y_i = \beta_0 + \beta_1 x_1 + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i, & \text{если } i\text{–й клиент — женщина} \\ \beta_0 + \epsilon_i, & \text{если } i\text{–й клиент — мужчина.} \end{cases} \tag{3.27}$$

Здесь β_0 можно интерпретировать как средний баланс на кредитной карте у мужчин, $\beta_0+\beta_1$ — как средний баланс на кредитной карте у женщин, а β_1 — как среднюю разницу по балансу на кредитной карте между женщинами и мужчинами.

В табл. 3.7 приведены оценки коэффициентов и другая информация по модели (3.27). Средний долг на кредитной карте у мужчин оценен в 509.80\$, тогда как у женщин он оказался на 19.73\$ выше, т. е. 509.80\$ + 19.73\$ = 529.53\$. Заметьте, однако, что p–значение у индикаторной переменной очень высокое. Это указывает на отсутствие статистически значимой разницы между мужчинами и женщинами по среднему балансу на кредитной карте.

ТАБЛИЦА 3.7. *Коэффициенты регрессии* `balance` *по* `gender`*, рассчитанные по методу наименьших квадратов на основе данных* `Credit`

	Коэффициент	Ст. ошибка	t	p
Свободный член	509.80	33.13	15.389	< 0.0001
`gender[Female]`	19.73	46.05	0.429	0.6690

Решение закодировать пол женщин в (3.27) при помощи 1, а мужчин при помощи 0 является произвольным и не влияет на качество итоговой модели, однако оно определяет интерпретацию коэффициентов. Если бы мы закодировали пол мужчин при помощи 1, а пол женщин при помощи 0, то оценки β_0 и β_1 составили бы 529.53\$ и -19.73\$ соответственно, что снова привело бы к предсказанному долгу на кредитной карте в 529.53\$ − 19.73\$ = 509.80\$ у мужчин и 529.53\$ у женщин. В качестве альтернативы кодированию с помощью 0 и 1 мы могли бы создать индикаторную переменную

$$x_i = \begin{cases} 1, & \text{если } i\text{–й клиент — женщина} \\ -1, & \text{если } i\text{–й клиент — мужчина} \end{cases}$$

и использовать эту переменную в регрессионном уравнении. Это приводит к следующей модели:

$$y_i = \beta_0 + \beta_1 x_1 + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i, & \text{если } i\text{–й клиент — женщина} \\ \beta_0 - \beta_1 + \epsilon_i, & \text{если } i\text{–й клиент — мужчина.} \end{cases}$$

Теперь β_0 можно интерпретировать как общий средний долг на кредитной карте без учета пола клиентов, а β_1 — как величину, на которую долг у женщин и мужчин выше и ниже этого общего среднего значения соответственно. В данном примере оценка β_0 составляет 519.665\$ — среднее от средних значений у мужчин (509.80\$) и женщин (529.53\$). Оценка β_1 составила бы 9.865\$, т. е. половину от 19.73\$ — средней величины различий между женщинами и мужчинами. Важно подчеркнуть, что итоговые предсказания кредитного баланса для мужчин и женщин будут идентичными вне зависимости от использованной схемы кодирования. Различия появляются только в интерпретации коэффициентов.

Качественные предикторы, имеющие более двух уровней

Когда качественный предиктор имеет более двух уровней, одна индикаторная переменная не может представить все возможные значения. В та-

кой ситуации мы можем создать дополнительные индикаторные переменные. Так, для `ethnicity` (этническая принадлежность) мы создаем две такие переменные. Первой из них могла бы быть

$$x_{i1} = \begin{cases} 1, & \text{если } i\text{–й клиент — азиат} \\ 0, & \text{если } i\text{–й клиент — не азиат,} \end{cases} \tag{3.28}$$

а второй —

$$x_{i2} = \begin{cases} 1, & \text{если } i\text{–й клиент — белый} \\ 0, & \text{если } i\text{–й клиент — не белый.} \end{cases} \tag{3.29}$$

Тогда обе эти переменные можно было бы использовать в уравнении регрессии для получения следующей модели:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i, & \text{если } i\text{–й клиент — азиат} \\ \beta_0 + \beta_2 + \epsilon_i, & \text{если } i\text{–й клиент — белый} \\ \beta_0 + \epsilon_i, & \text{если } i\text{–й клиент — афроамериканец.} \end{cases} \tag{3.30}$$

Теперь β_0 можно интерпретировать как средний кредитный баланс у афроамериканцев, β_1 — как разницу по среднему балансу у азиатов и афроамериканцев, а β_2 — как разницу по среднему балансу у белых и афроамериканцев. Число индикаторных переменных всегда будет на единицу меньше числа уровней фактора. Уровень без соответствующей индикаторной переменной — афроамериканцы в данном примере — известен как *базовый*.

базовый уровень

Из табл. 3.8 мы видим, что оцененное среднее значение `balance` для базового уровня (афроамериканцы) составляет 531.00\$. Оцененный долг в группе азиатов на 18.69\$ меньше, чем в группе афроамериканцев, а долг в группе белых на 12.50\$ меньше, чем в группе афроамериканцев. Однако p–значения у коэффициентов этих двух индикаторных переменных очень большие, что указывает на отсутствие статистических оснований для вывода о существовании реальной разницы между этническими группами по балансу на кредитной карте. Повторим еще раз, что уровень, выбранный в качестве базового, является произвольным, и предсказания для каждой группы в итоге будут одинаковыми вне зависимости от этого выбора. Однако коэффициенты и их p–значения все же зависят от выбранного способа кодирования индикаторных переменных. Вместо того чтобы полагаться на отдельные коэффициенты, мы можем использовать F–критерий для проверки гипотезы $H_0 = \beta_1 = \beta_2 = 0$ — данный тест не будет зависеть от способа кодирования. P–значение для этого F–критерия составляет 0.96, на основании чего мы не можем отклонить нулевую гипотезу об отсутствии связи между `balance` и `ethnicity`.

ТАБЛИЦА 3.8. *Коэффициенты регрессии* `balance` *по* `ethnicity`*, рассчитанные по методу наименьших квадратов на основе данных* `Credit`*. Эта линейная модель определена в (3.30), т. е. этническая принадлежность закодирована при помощи двух индикаторных переменных, представленных в (3.28) и (3.29)*

	Коэффициент	Ст. ошибка	t	p
Свободный член	531.00	46.32	11.464	< 0.0001
`ethnicity[Asian]`	−18.69	65.02	−0.287	0.7740
`ethnicity[Caucasian]`	−12.50	56.68	−0.221	0.8260

Использование метода индикаторных переменных не составляет труда при включении в модель как количественных, так и качественных предикторов. Например, для нахождения регрессионной зависимости `balance` от количественной переменной вроде `income` и качественной переменной вроде `student` мы должны просто создать индикаторную переменную для `student`, а затем подогнать множественную регрессионную модель, используя `income` и эту индикаторную переменную в качестве предикторов кредитного баланса.

Помимо рассмотренного здесь метода индикаторных переменных, существует множество других способов кодирования качественных переменных. Все эти способы приводят к созданию эквивалентных моделей, однако коэффициенты и их интерпретация различаются, поскольку эти коэффициенты предназначены для отображения разных *контрастов*. Эта тема выходит за рамки данной книги, и мы не будем дальше ее рассматривать.

контрасты

3.3.2 Расширения линейной модели

Стандартная линейная регрессионная модель (3.19) позволяет получать интерпретируемые результаты и довольно хорошо работает при решении многих практических проблем. Однако она основана на ряде строгих ограничивающих допущений, которые на практике часто не выполняются. Два наиболее важных допущения утверждают, что зависимость между предикторами и откликом является *аддитивной* и *линейной*. Условие аддитивности означает, что влияние предиктора X_j на отклик Y не зависит от значений других предикторов. Условие линейности заключается в том, что изменение отклика Y при изменении X_j на одну единицу одинаково вне зависимости от значения X_j. В этой книге мы рассматриваем несколько сложных методов, которые ослабляют эти допущения. Здесь мы кратко рассмотрим некоторые классические подходы, позволяющие расширить линейную модель.

аддитивность
линейность

Снятие ограничения на аддитивность

В нашем предыдущем анализе данных `Advertising` мы сделали заключение о том, что объем продаж (`sales`) связан с рекламой как на радио (`radio`), так и телевидении (`TV`). Линейные модели, легшие в основу этого вывода, подразумевали, что эффект на `sales` от увеличения затрат на рекламу в одном СМИ не зависит от затрат на другое СМИ. Например,

линейная модель (3.20) утверждает, что среднее изменение sales при увеличении TV на одну единицу всегда равно β_1, вне зависимости от затрат на радиорекламу.

Однако эта простая модель может быть неверной. Представьте, что на самом деле затраты на радиорекламу увеличивают эффективность телерекламы, в связи с чем угловой коэффициент для TV должен возрастать одновременно с radio. В такой ситуации при фиксированном бюджете в 100 000$ трата половины денег на радиорекламу, а второй половины на телерекламу может увеличить продажи в большей степени, нежели при направлении всех денег отдельно на теле– или радиорекламу. В маркетинге этот эффект известен как *синергия*, а в статистике его называют эффектом *взаимодействия*. Как видно на рис. 3.5, подобный эффект может иметь место в данных по рекламе. Заметьте, что при низких уровнях TV и radio действительные значения sales ниже тех значений, которые предсказывает модель. Однако при равномерном распределении бюджета на оба типа рекламы модель имеет тенденцию завышать продажи.

Рассмотрим стандартную линейную регрессионную модель с двумя переменными:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon.$$

Согласно этой модели, при увеличении X_1 на одну единицу Y увеличится в среднем на β_1 единиц. Заметьте, что присутствие X_2 не изменяет этого утверждения, т. е. вне зависимости от значения X_2 увеличение X_1 на одну единицу приведет к увеличению Y на β_1 единиц. Один из способов расширения этой модели для учета синергии заключается в добавлении третьего предиктора, который называется *эффектом взаимодействия* и вычисляется как произведение X_1 и X_2. Это приводит к модели

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon. \tag{3.31}$$

Каким образом добавление этой новой переменной ослабляет условие аддитивности? Обратите внимание на то, что (3.31) можно переписать в виде

$$Y = \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \epsilon = \beta_0 + \widetilde{\beta}_1 X_1 + \beta_2 X_2 + \epsilon, \tag{3.32}$$

где $\widetilde{\beta}_1 = \beta_1 + \beta_3 X_2$. Поскольку $\widetilde{\beta}_1$ изменяется одновременно с X_2, то влияние X_1 на Y больше не будет постоянным: изменение X_2 приведет к изменению эффекта X_1 на Y.

Представьте, например, что мы заинтересованы в изучении производительности некоторой фабрики. Мы хотим предсказать количество единиц продукции (units) на основе числа производственных линий (lines) и общего числа работников (workers). Вполне вероятно, что эффект от увеличения числа производственных линий будет зависеть от числа работников, поскольку если не будет работников для управления линиями, то увеличение числа этих линий не приведет к росту продукции. Это предполагает, что в линейную модель для предсказания units следует включить эффект взаимодействия между lines и workers. Предположим, что при подгонке модели мы получаем

$$\begin{aligned}\texttt{units} \approx 1.2 + 3.4 \times \texttt{lines} + 0.22 \times \texttt{workers} + 1.4 \times (\texttt{lines} \times \texttt{workers}) = \\ = 1.2 + (3.4 + 1.4 \times \texttt{workers}) \times \texttt{lines} + 0.22 \times \texttt{workers}.\end{aligned}$$

Другим словами, добавление новой линии увеличит количество единиц продукции на $3.4 + 1.4 \times \texttt{workers}$. Следовательно, чем больше у нас работников, тем сильнее будет эффект переменной lines.

Теперь мы вернемся к примеру с данными по рекламе. Линейная модель, которая использует radio, TV и их взаимодействие для предсказания sales, принимает следующую форму:

$$\begin{aligned}\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times (\texttt{radio} \times \texttt{TV}) + \epsilon = \\ = \beta_0 + (\beta_1 + \beta_3 \times \texttt{radio}) \times \texttt{TV} + \beta_2 \times \texttt{radio} + \epsilon.\end{aligned} \tag{3.33}$$

Мы можем интерпретировать β_3 как прирост эффективности телерекламы в ответ на увеличение радиорекламы на одну единицу (и наоборот). Коэффициенты, получаемые при подгонке модели (3.33), представлены в табл. 3.9.

ТАБЛИЦА 3.9. *Коэффициенты регрессионной зависимости* sales *от* TV, radio *и их взаимодействия (см. уравнение 3.33), рассчитанные по методу наименьших квадратов на основе данных* Advertising

	Коэффициент	Ст. ошибка	t	p
Свободный член	6.7502	0.248	27.23	< 0.0001
TV	0.0191	0.002	12.70	< 0.0001
radio	0.0289	0.009	3.24	0.0014
TV × radio	0.0011	0.000	20.73	< 0.0001

Результаты, приведенные в табл. 3.9, со всей очевидностью указывают на то, что модель, включающая эффект взаимодействия, превосходит модель, которая содержит только *главные эффекты*. P–значение для эффекта взаимодействия (TV × radio) является крайне низким, что говорит о наличии веских оснований в пользу $H_a : \beta_3 \neq 0$. Иными словами, четко видно, что истинная зависимость не является аддитивной. Коэффициент детерминации R^2 для модели (3.33) составляет 96.8%, тогда как для модели, предсказывающей sales на основе TV и radio без эффекта их взаимодействия, он составляет лишь 89.7%. Это означает, что $(96.8 - 89.7)/(100 - 89.7) = 69\%$ дисперсии sales, остающейся после подгонки аддитивной модели, были объяснены эффектом взаимодействия. Оценки коэффициентов, приведенные в табл. 3.9, указывают на то, что увеличение затрат на телерекламу на 1000\$ связано с ростом продаж на $(\hat{\beta}_1 + \hat{\beta}_3 \times \texttt{radio}) \times 1000 = 19 + 1.1 \times \texttt{radio}$ единиц. В свою очередь увеличение на 1000\$ затрат на радиорекламу будет связано с ростом продаж на $(\hat{\beta}_2 + \hat{\beta}_3 \times \texttt{TV}) \times 1000 = 29 + 1.1 \times \texttt{TV}$ единиц.

главный эффект

В этом примере p–значения для TV, radio и эффекта их взаимодействия являются статистически значимыми (табл. 3.9), и поэтому ясно, что в модель должны быть включены все три переменные. Однако иногда

принцип иерархии

бывает так, что эффект взаимодействия имеет очень низкое p–значение, а соответствующие главные эффекты (в данном случае TV и radio) — нет. *Принцип иерархии* гласит, что *при включении в модель эффекта взаимодействия нам следует также включать главные эффекты, даже если p–значения их коэффициентов незначимы.* Другими словами, если взаимодействие между X_1 и X_2 выглядит важным, тогда нам следует включить в модель и X_1, и X_2, даже если оценки их коэффициентов имеют высокие p–значения. Обоснование этого принципа состоит в том, что если переменная $X_1 \times X_2$ связана с откликом, то проверка равенства коэффициентов X_1 и X_2 нулю практически не представляет интереса. Кроме того, переменная $X_1 \times X_2$ обычно коррелирует с X_1 и X_2, и поэтому исключение главных эффектов изменяет смысл взаимодействия.

В предыдущем примере мы рассмотрели взаимодействие между переменными TV и radio, обе из которых являются количественными. Однако концепция взаимодействия одинаково применима также к качественным переменным и к комбинациям количественных и качественных переменных. Более того, взаимодействие между качественной и количественной переменными имеет особенно привлекательную интерпретацию. Обратимся к набору данных Credit из подраздела 3.3.1 и предположим, что мы хотим предсказать balance на основе количественной переменной income и качественной переменной student.

В отсутствие эффекта взаимодействия модель принимает следующую форму:

$$\begin{aligned}
&\texttt{balance}_i \approx \beta_0 + \beta_1 \times \texttt{income}_i + \\
&\begin{cases} \beta_2, & \text{если } i\text{–й клиент — студент} \\ 0, & \text{если } i\text{–й клиент — не студент} \end{cases} = \\
&\beta_1 \times \texttt{income}_i + \\
&\begin{cases} \beta_0 + \beta_2, & \text{если } i\text{–й клиент — студент} \\ \beta_0, & \text{если } i\text{–й клиент — не студент.} \end{cases}
\end{aligned} \tag{3.34}$$

Заметьте, что это сводится к подгонке двух параллельных линий — одной для студентов, а другой для не студентов. Уравнения линий для студентов и не студентов имеют разные свободные члены — $\beta_0 + \beta_2$ против β_0, — но одинаковый угол наклона β_1. Эта ситуация показана на рис. 3.7 слева. Факт параллельного расположения линий означает, что средняя величина изменения balance при увеличении income на одну единицу не зависит от того, является ли тот или иной клиент студентом. Потенциально это вносит в модель серьезное ограничение, поскольку в действительности изменение дохода может оказывать очень разные эффекты на долг по кредитной карте у студента и не студента.

Это ограничение можно упразднить, добавив эффект взаимодействия — переменную, созданную путем умножения income на индикаторную переменную student. Тогда наша модель превратится в следующую:

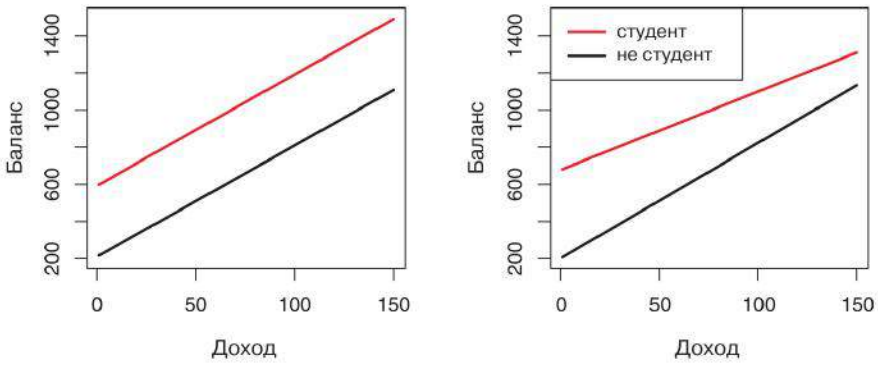

РИСУНОК 3.7. *Коэффициенты регрессионной зависимости* `balance` *от* `income` *у студентов и не студентов, подогнанной по методу наименьших квадратов на основе данных* `Credit`. Слева: *подогнана модель (3.34). Взаимодействия между* `income` *и* `student` *нет.* Справа: *подогнана модель (3.35). Имеет место взаимодействие между* `income` *и* `student`

$$\begin{aligned}
&\texttt{balance}_i \approx \beta_0 + \beta_1 \times \texttt{income}_i + \\
&\begin{cases} \beta_2 + \beta_3 \times \texttt{income}_i, & \text{если } i\text{–й клиент — студент} \\ 0, & \text{если } i\text{–й клиент — не студент} \end{cases} = \\
&\begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times \texttt{income}_i, & \text{если } i\text{–й клиент — студент} \\ \beta_0 + \beta_1 \times \texttt{income}_i, & \text{если } i\text{–й клиент — не студент.} \end{cases}
\end{aligned} \tag{3.35}$$

Подчеркнем еще раз, что у нас есть две разные регрессионные линии для студентов и не студентов. Однако теперь эти регрессионные линии имеют разные свободные члены ($\beta_0 + \beta_2$ против β_0) и коэффициенты угла наклона ($\beta_1 + \beta_3$ против β_1). Это позволяет учесть возможность того, что изменения в доходе могут по–разному влиять на долг по кредитной карте у студентов и не студентов. График на рис. 3.7 справа показывает зависимости между `income` и `balance` у студентов и не студентов, подогнанные в соответствии с моделью (3.35). Видно, что угол наклона прямой у студентов ниже, чем у не студентов. Это предполагает, что увеличение дохода у студентов связано с меньшим ростом долга по кредитной карте.

Нелинейные зависимости

Как обсуждалось ранее, линейная регрессионная модель (3.19) предполагает линейную связь между откликом и предикторами. Однако в некоторых случаях истинная связь между откликом и предикторами может быть нелинейной. Здесь мы представим очень простой способ непосредственного расширения линейной модели, позволяющий подгонять нелинейные зависимости при помощи *полиномиальной регрессии*. В последующих главах мы представим более сложные подходы для подгонки нелинейных моделей, основанные на более универсальной методологии.

полиномиальная регрессия

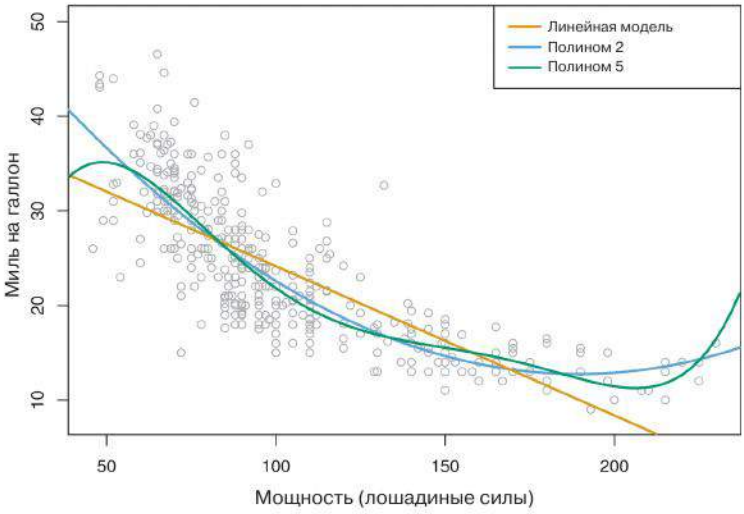

РИСУНОК 3.8. *Данные из таблицы* `Auto`. *Приведены* `mpg` *(расход топлива) и* `horsepower` *(мощность двигателя) для нескольких автомобилей. Линейная регрессионная модель показана в виде оранжевой линии. Линейная регрессионная модель, включающая* $\texttt{horsepower}^2$, *показана в виде голубой линии. Линейная регрессионная модель, включающая все полиномы* `horsepower` *вплоть до 5–й степени, показана в виде зеленой линии*

Рассмотрим рис. 3.8, где показана зависимость `mpg` (расход топлива, миль на галлон) от `horsepower` (мощность двигателя, лошадиные силы) для нескольких автомобилей из набора данных `Auto`. Оранжевая линия соответствует линейной регрессионной модели. Имеет место выраженная связь между `mpg` и `horsepower`, однако хорошо видно, что эта связь нелинейна: данные указывают на искривленную форму зависимости. Простой подход для включения нелинейных зависимостей в линейную модель состоит в добавлении преобразованных предикторов. Например, характер расположения точек на рис. 3.8 указывает на *квадратичную* форму зависимости, из чего следует, что модель вида

$$\texttt{mpg} = \beta_0 + \beta_1 \times \texttt{horsepower} + \beta_2 \times \texttt{horsepower}^2 + \epsilon \qquad (3.36)$$

может дать более приемлемое описание данных. Уравнение 3.36 предполагает предсказание `mpg` на основе нелинейной функции `horsepower`. *Однако это все еще линейная модель!* Другими словами, (3.36) — это просто множественная регрессионная модель, в которой $X_1 = \texttt{horsepower}$, а $X_2 = \texttt{horsepower}^2$. Поэтому для нахождения β_0, β_1 и β_2 и подгонки искривленной линии мы можем воспользоваться стандартным программным обеспечением для линейной регрессии. Результат подгонки квадратичной модели к этим данным представлен в виде голубой линии на рис. 3.8. Похоже, что квадратичное уравнение описывает данные намного лучше уравнения, содержащего только линейный член. Статистика R^2 у квадратичной модели составляет 0.688 против 0.606 у линейной модели, а p–значение квадратичного члена является высоко значимым (табл. 3.10).

ТАБЛИЦА 3.10. *Коэффициенты регрессионной зависимости* mpg *от* horsepower *и* $\texttt{horsepower}^2$*, рассчитанные по методу наименьших квадратов на основе данных* Auto

	Коэффициент	Ст. ошибка	t	p
Свободный член	56.9001	1.8004	31.6	< 0.0001
horsepower	−0.4662	0.0311	−15.0	< 0.0001
$\texttt{horsepower}^2$	0.0012	0.0001	10.1	< 0.0001

Если включение $\texttt{horsepower}^2$ привело к такому существенному улучшению модели, то почему бы не добавить также $\texttt{horsepower}^3$, $\texttt{horsepower}^4$ или даже $\texttt{horsepower}^5$? Зеленая кривая на рис. 3.8 представляет собой результат подгонки модели, включающей все полиномы вплоть до 5–й степени. Полученная модель выглядит излишне извилистой, т. е. не ясно, действительно ли включение дополнительных членов привело к более качественному описанию данных.

Описанный нами выше метод расширения линейной модели для учета нелинейных зависимостей известен как *полиномиальная регрессия*, поскольку мы добавляем в регрессионную модель полиномиальные функции предикторов. В главе 7 мы продолжим рассмотрение этого, а также других способов нелинейных расширений линейной модели.

3.3.3 Потенциальные проблемы

При подгонке линейной регрессионной модели к тому или иному набору данных может возникнуть целый ряд проблем. Наиболее распространенные из этих проблем заключаются в следующем.

1. Нелинейность связи между откликом и предикторами.
2. Корреляция остатков.
3. Непостоянная дисперсия остатков.
4. Выбросы.
5. Влиятельные наблюдения.
6. Коллинеарность.

На практике обнаружение и преодоление этих проблем являются в равной степени искусством и наукой. На эту тему написано огромное количество страниц в многочисленных книгах. Поскольку в этой книге линейная регрессионная модель не является основным центром внимания, мы приведем лишь краткое описание некоторых ключевых аспектов.

1. Нелинейность данных

Линейная регрессионная модель предполагает наличие прямолинейной связи между предикторами и откликом. Если истинная зависимость далека от линейной, то фактически все заключения, которые мы делаем на основе подогнанной модели, будут сомнительными. Кроме того, существенно более низкой может оказаться точность предсказаний такой модели.

графики остатков

Графики остатков представляют собой полезный графический инструмент для обнаружения нелинейности. Имея простую линейную регрессионную модель, мы можем изобразить остатки $e_i = y_i - \hat{y}_i$ в зависимости от предиктора x_i. В случае множественной регрессионной модели, которая содержит несколько предикторов, мы вместо этого изображаем зависимость остатков от предсказанных, или модельных, значений $\hat{y}_i$. В идеале график остатков не выявит никакой различимой структуры в расположении точек. Наличие же такой структуры может указывать на определенную проблему с линейной моделью.

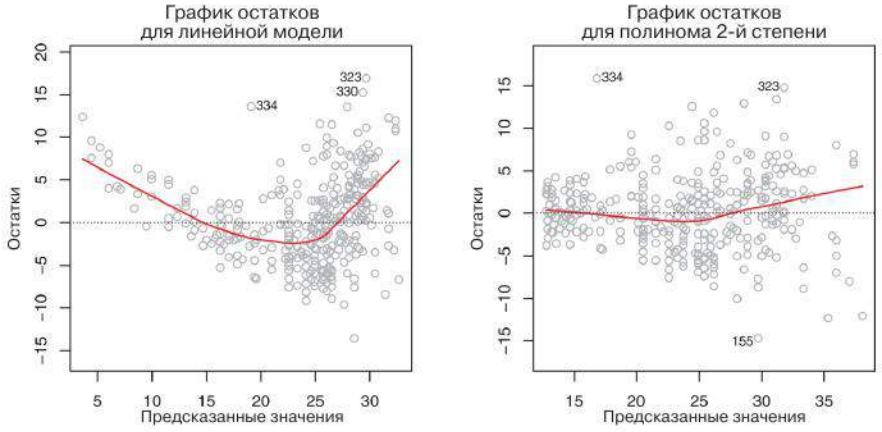

РИСУНОК 3.9. *Графики зависимости остатков от предсказанных, или модельных, значений для набора данных* `Auto`. *На каждом графике красная кривая представляет собой сглаживающую линию, подогнанную к остаткам с целью облегчить выявление характера их расположения.* Слева: *линейная регрессия* `mpg` *по* `horsepower`. *Выраженная закономерность в расположении остатков указывает на нелинейность в данных.* Справа: *линейная регрессия* `mpg` *по* `horsepower` *и* $\texttt{horsepower}^2$. *Закономерности в расположении остатков почти нет*

Слева на рис. 3.9 изображены остатки линейной регрессии `mpg` по `horsepower`, подогнанной по данным `Auto` (см. рис. 3.8). Красная кривая представляет собой сглаживающую линию, подогнанную к остаткам с целью облегчить обнаружение каких–либо закономерностей в их расположении. Остатки демонстрируют выраженную U–образную форму, что является надежным свидетельством нелинейности в данных. Справа на рис. 3.9 показаны остатки, полученные в результате подгонки модели (3.36) с квадратичным членом. Здесь закономерности в расположении остатков почти нет, что указывает на более адекватное описание данных, вызванное добавлением квадратичного члена.

Если график остатков свидетельствует о наличии нелинейной связи в данных, то простым решением проблемы является добавление в модель нелинейных преобразований предикторов, таких как $\log X$, $\sqrt{X}$ и X^2. В последующих главах этой книги мы будем обсуждать другие, более продвинутые нелинейные методы, позволяющие преодолеть эту проблему.

2. Корреляция остатков

Важное допущение линейной регрессионной модели состоит в том, что остатки $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ являются некоррелированными. Что это означает? Например, если в остатках нет корреляции, то положительное значение ϵ_i практически не несет никакой информации о знаке ϵ_{i+1}. Стандартные ошибки, которые рассчитываются для оценок регрессионных коэффициентов или предсказанных значений отклика, основаны на предположении о некоррелированных остатках. Если же корреляция в остатках присутствует, то оцененные стандартные ошибки обычно будут занижать истинные значения стандартных ошибок. В результате этого доверительные интервалы и интервалы предсказаний будут уже, чем следует. Например, 95%-ный доверительный интервал может включать истинное значение параметра с вероятностью, намного меньшей, чем 0.95. P–значения параметров модели также будут ниже, чем следует; это могло бы привести к ложному заключению о том, что тот или иной параметр является статистически значимым. Короче говоря, коррелированные остатки могут привести к необоснованному чувству уверенности в нашей модели.

В качестве экстремального примера предположим, что мы случайно продублировали наши данные и это привело к возникновению пар идентичных наблюдений и остатков. Проигнорировав это, мы вычисляли бы стандартные ошибки, как будто у нас была выборка объемом $2n$, хотя на самом деле у нас есть только n наблюдений. Наши оценки параметров для $2n$ и n наблюдений были бы одинаковыми, но доверительные интервалы оказались бы в $\sqrt{2}$ раз ниже!

Отчего возникает корреляция в остатках? Подобные корреляции обычны при работе с *временными рядами*, которые состоят из наблюдений, полученных через определенные промежутки времени. Часто наблюдения, полученные в соседних временных точках, будут иметь положительно коррелирующие остатки. Для обнаружения этого в некотором наборе данных мы можем изобразить остатки из нашей модели в зависимости от времени. В отсутствие корреляции в остатках на графике не должно быть видно никаких закономерностей. С другой стороны, при наличии положительной корреляции в остатках может наблюдаться *трекинг* — явление, при котором соседние остатки имеют похожие значения. Рисунок 3.10 иллюстрирует эти идеи. На верхнем графике мы видим остатки из линейной регрессии, подогнанной к имитированным данным, в которых остатки не коррелируют. Указаний на наличие временного тренда в этом случае нет. Остатки же, показанные на нижнем графике, происходят из набора данных, в котором корреляция составляет 0.9. Здесь имеет место выраженная закономерность — соседние остатки имеют тенденцию принимать похожие значения. Наконец, график, представленный в центре, иллюстрирует умеренный случай, в котором корреляция остатков составила 0.5. Трекинг все еще заметен, но выражен не так четко.

временной ряд

трекинг

Разработано большое количество методов, позволяющих корректно учесть корреляцию в остатках при работе с временными рядами. Корреляции в остатках могут встречаться также при работе с данными, которые не являются временными рядами. Например, представьте себе ситуацию, в которой рост людей предсказывается по их весу. Условие некоррелированных остатков могло бы нарушаться в случаях, когда некоторые участ-

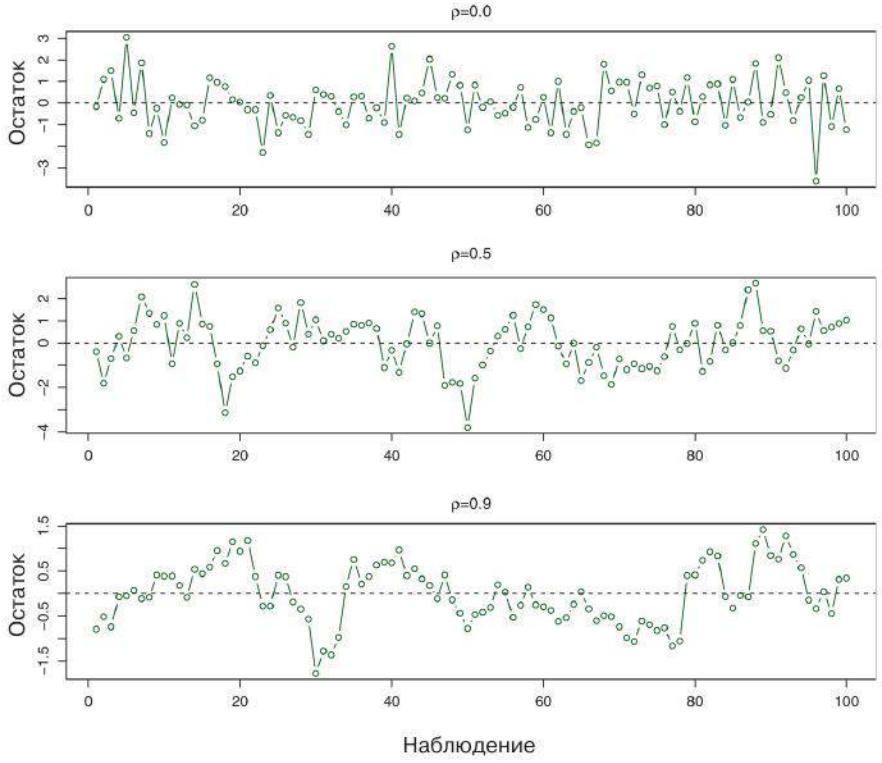

РИСУНОК 3.10. *Графики остатков для временных рядов, которые были сгенерированы с использованием разных уровней корреляции ρ между соседними временными точками*

ники исследования были членами одной семьи, или употребляли одинаковую диету, или были подвержены действию одинаковых факторов среды. В целом предположение о некоррелированных остатках является чрезвычайно важным для линейной регрессии и других статистических методов, и ключом к снижению риска возникновения подобных корреляций является хорошее планирование эксперимента.

3. Непостоянная дисперсия остатков

Другое важное допущение линейной регрессионной модели состоит в том, что ее остатки имеют постоянную дисперсию, т. е. $\mathrm{Var}(\epsilon_i) = \sigma^2$. На этом допущении основано вычисление соответствующих стандартных ошибок и доверительных интервалов, а также проверка статистических гипотез.

гетероскедастичность

К сожалению, очень часто дисперсия остатков оказывается непостоянной. Например, она может возрастать с увеличением значений отклика. Непостоянство, или *гетероскедастичность*, дисперсии можно выявить по наличию воронкообразной формы на графике остатков. На рис. 3.11 слева приведен пример, в котором разброс остатков возрастает по мере увеличения предсказанных значений. При возникновении такой проблемы одним

из возможных решений является преобразование отклика Y с использованием вогнутой функции вроде $\log Y$ или $\sqrt{Y}$. Подобное преобразование сопровождается более сильным «сжатием» более высоких значений остатков, что приводит к снижению гетероскедастичности. Справа на рис. 3.11 показан график остатков после преобразования отклика путем логарифмирования. Теперь остатки демонстрируют постоянную дисперсию, хотя и имеется некоторое указание на слабовыраженную нелинейную зависимость в данных.

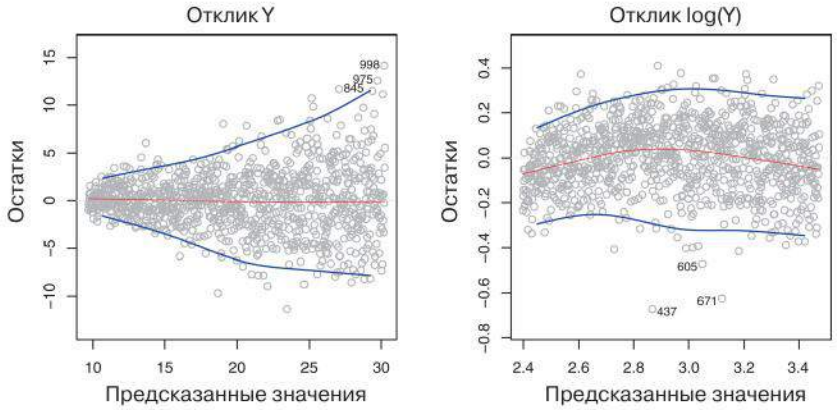

РИСУНОК 3.11. *Графики остатков. На каждом графике красная кривая представляет собой сглаживающую линию, подогнанную к остаткам для выявления характера их расположения. Синие линии соответствуют квартилям остатков и подчеркивают закономерность в их расположении.* Слева: *воронкообразная форма указывает на гетероскедастичность.* Справа: *отклик был прологарифмирован, и теперь указаний на гетероскедастичность нет*

Иногда у нас есть хорошее представление о дисперсии каждого значения зависимой переменной. Например, i–е значение отклика могло бы представлять собой среднее значение из n_i исходных наблюдений. Если каждое из этих исходных наблюдений не коррелирует с дисперсией σ^2, то их среднее значение имеет дисперсию $\sigma_i^2 = \sigma^2/n_i$. В этом случае простым решением проблемы является подгонка модели по методу *взвешенных наименьших квадратов*, в котором веса пропорциональны обратным величинам дисперсий ($w_i = n_i$ в данном примере). Большинство программ для расчета линейной регрессии позволяет работать с весами наблюдений.

метод взвешенных наименьших квадратов

4. Выбросы

Выброс — это наблюдение, у которого y_i существенно отличается от значения, предсказанного моделью. Выбросы могут возникать по ряду причин — например, неверный учет наблюдения во время сбора данных.

выброс

Красная точка (наблюдение № 20), показанная слева на рис. 3.12, является примером типичного выброса. Красная сплошная линия — это регрессионная модель, подогнанная по методу наименьших квадратов, а синяя

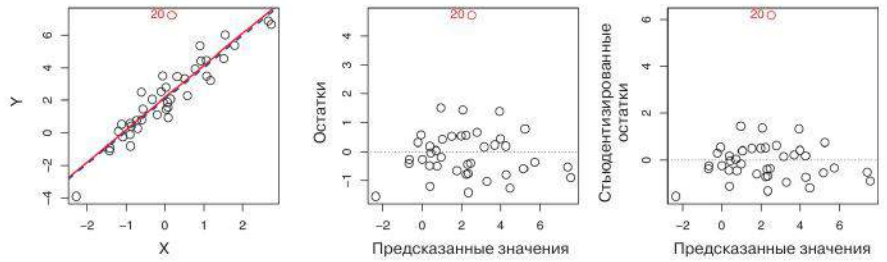

РИСУНОК 3.12. Слева: *линия наименьших квадратов показана красным цветом, а регрессионная линия, подогнанная после удаления выброса, — синим.* В центре: *график остатков четко показывает наличие выброса.* Справа: *стьюдентизированный остаток выброса равен 6, тогда как обычно мы ожидаем значения от −3 до 3.*

прерывистая линия — это модель, подогнанная по методу наименьших квадратов после удаления выброса. В данном случае удаление выброса оказывает незначительный эффект на модель: оно почти не изменяет угла наклона и приводит лишь к незначительному снижению коэффициента свободного члена. Для выброса, который не имеет необычного значения предиктора, подобное незначительное влияние на модель типично. Однако даже если выброс не оказывает большого эффекта на модель, он может привести к возникновению других проблем. Так, в данном примере RSE составляет 1.09 при включении выброса в модель и лишь 0.77 после его удаления. Поскольку RSE используется для вычисления всех доверительных интервалов и p–значений, подобное резкое увеличение, вызванное лишь одним наблюдением, может иметь последствия для интерпретации всей модели. Похожим образом включение выброса в модель вызывает снижение R^2 с 0.892 до 0.805.

Для обнаружения выбросов можно воспользоваться графиками остатков. В этом примере выброс виден очень хорошо (см. график остатков, показанный в центре на рис. 3.12). Однако на практике бывает сложно решить, насколько большим должен быть остаток, чтобы считать соответствующее наблюдение выбросом. Для преодоления этой проблемы вместо исходных остатков мы можем изобразить *стьюдентизированные остатки*, рассчитанные путем деления каждого остатка e_i на его оцененное стандартное отклонение. Наблюдения, чьи стьюдентизированные остатки в абсолютном выражении превышают 3, потенциально являются выбросами. Справа на рис. 3.12 стьюдентизированный остаток выброса превышает 6, тогда как стьюдентизированные остатки остальных наблюдений изменяются от -2 до 2.

стьюдентизированный остаток

Если мы убеждены, что выброс появился из–за ошибки, допущенной в ходе сбора данных или документирования, то простым решением будет исключение этого наблюдения из анализа. Однако следует соблюдать осторожность, поскольку выброс может указывать и на недостаток в самой модели (например, отсутствие в ней важного предиктора).

5. Наблюдения с высокой разбалансировкой

высокая разбалансировка

Как мы только что выяснили, выбросы — это наблюдения, у которых для заданного предиктора x_i наблюдается необычное значение отклика y_i. В то же время наблюдения с *высокой разбалансировкой*[11] имеют необычное значение x_i. Например, наблюдение № 41, показанное справа на рис. 3.13, обладает высокой разбалансировкой в том смысле, что значение предиктора у этого наблюдения велико в сравнении с другими наблюдениями. (Обратите внимание, что на рис. 3.13 показаны те же данные, что и на рис. 3.12, но с добавлением одного наблюдения, обладающего высокой разбалансировкой). Красной сплошной линией показана модель, подогнанная по методу наименьших квадратов, а прерывистой синей линией — модель, подогнанная после удаления наблюдения № 41. Сравнивая графики, представленные слева на рис. 3.12 и 3.13, мы видим, что удаление наблюдения с высокой разбалансировкой оказывает гораздо более сильное влияние на линию наименьших квадратов, нежели удаление выброса. И действительно, наблюдения с высокой разбалансировкой обычно оказывают значительное влияние на регрессионную линию. Влияние нескольких наблюдений на линию наименьших квадратов служит причиной беспокойства, поскольку такие наблюдения могут сделать непригодной всю модель. По этой причине обнаружению наблюдений с высокой разбалансировкой следует уделять серьезное внимание.

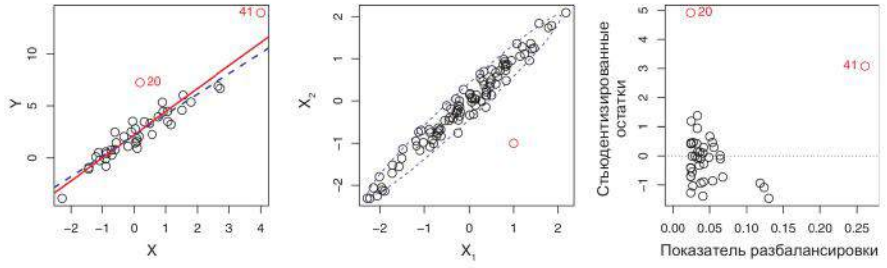

РИСУНОК 3.13. Слева: *наблюдение № 41 является наблюдением с высокой разбалансировкой, тогда как № 20 — нет. Красная линия соответствует модели, подогнанной ко всем данным, а синяя линия — модели, подогнанной после удаления наблюдения № 41.* В центре: *красная точка не является необычной по значению X_1 или значению X_2, но все же выходит за пределы основного массива данных и поэтому обладает высоким показателем разбалансировки.* Справа: *Наблюдение № 41 имеет высокий показатель разбалансировки и большой остаток*

В случае простой линейной регрессии наблюдения с высокой разбалансировкой обнаружить довольно легко, поскольку мы просто можем найти точки, у которых значение предиктора выходит за пределы нормального интервала значений. В случае же множественной регрессии со многими предикторами возможна ситуация, когда некоторое наблюдение вполне находится в пределах диапазонов значений отдельных предикторов, но является необычным по всем предикторам сразу. Пример набора

[11] В оригинале используется термин «high leverage points». — *Прим. пер.*

данных с двумя предикторами показан в центре на рис. 3.13. У большинства наблюдений значения предикторов находятся в пределах эллипса, ограниченного голубой прерывистой линией, однако красная точка выходит далеко за пределы этого диапазона. Тем не менее это наблюдение не является необычным ни по значению X_1, ни по значению X_2 в отдельности. Поэтому если мы исследуем только X_1 или только X_2, нам не удастся обнаружить, что это наблюдение обладает высокой разбалансировкой. Эта проблема более выражена в случае множественной регрессии, поскольку при наличии более двух предикторов не существует простого способа одновременно изобразить все имеющиеся в данных переменные.

показатель разбалансировки

Для количественного выражения степени разбалансировки того или иного наблюдения мы вычисляем *показатель разбалансировки*[12]. Высокое значение этого показателя говорит о том, что наблюдение обладает высокой разбалансировкой. Для простой регрессии

$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2}. \tag{3.37}$$

Из этого уравнения видно, что h_i возрастает по мере увеличения разности между x_i и $\bar{x}$. Существует простое обобщение h_i для случая со многими предикторами, однако соответствующую формулу мы здесь не приводим. Показатель разбалансировки всегда изменяется от $1/n$ до 1, а его среднее значение для всех наблюдений всегда равно $(p+1)/n$. Поэтому если показатель разбалансировки у некоторого значения намного превосходит $(p+1)/n$, то мы можем ожидать, что это наблюдение обладает высокой разбалансировкой.

Справа на рис. 3.13 показан график зависимости стьюдентизированных остатков от h_i для данных, представленных на том же рисунке слева. Наблюдение № 41 четко выделяется за счет очень высокого показателя разбалансировки, а также высокого стьюдентизированного остатка. Другими словами, оно является и выбросом, и наблюдением с высокой разбалансировкой. Это особенно опасная комбинация! Данный график объясняет также, почему наблюдение № 20 имеет относительно слабый эффект на результат подгонки линии наименьших квадратов (см. рис. 3.12) — оно обладает низкой разбалансировкой.

6. Коллинеарность

коллинеарность

Под коллинеарностью понимают ситуацию, в которой два или более предиктора тесно связаны друг с другом. Концепция коллинеарности проиллюстрирована на рис. 3.14 на примере набора данных `Credit`. Слева на этом рисунке видно, что два предиктора `limit` (лимит кредита) и `age` (возраст клиента) практически не связаны друг с другом. В то же время справа на этом рисунке предикторы `limit` и `rating` (рейтинг клиента) тесно коррелируют, и мы говорим, что они *коллинеарны*. Наличие коллинеарности может вызвать проблемы в ходе выполнения регрессионного анализа в связи с трудностью выделения эффектов отдельных коллинеарных

[12] В оригинале используется термин «leverage statistic». Речь идет о диагональных элементах h_i т. н. *матрицы проекции на пространство предикторов* $\mathbf{H}$ (применяются также названия «матрица влияния» и «матрица воздействия»; англ. «hat matrix», «influence matrix»). — *Прим. пер.*

предикторов на отклик. Иными словами, поскольку limit и rating имеют тенденцию к согласованному увеличению или снижению, установление связи каждой из этих переменных с откликом (balance) может оказаться затруднительным.

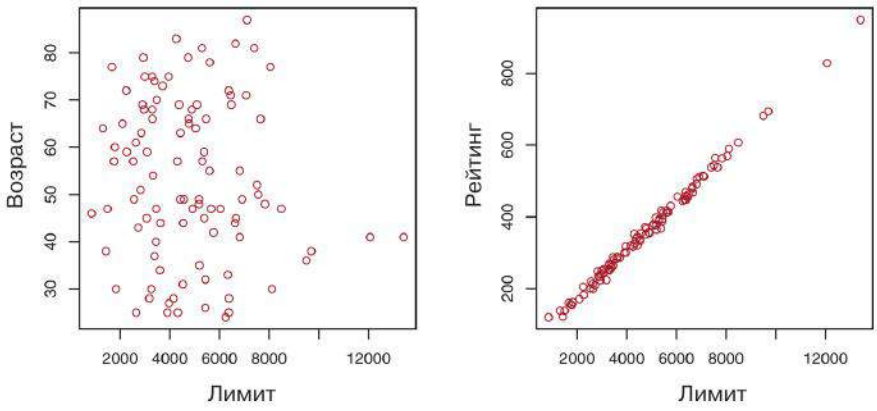

РИСУНОК 3.14. *Диаграммы рассеяния для наблюдений из набора данных* Credit. Слева: *график зависимости между возрастом клиента (*age*) и кредитным лимитом (*limit*). Эти две переменные не являются коллинеарными.* Справа: *график зависимости между рейтингом клиента* rating *и кредитным лимитом (*limit*). Имеет место высокая коллинеарность*

Рисунок 3.15 иллюстрирует некоторые из трудностей, которые могут возникнуть из–за коллинеарности. Слева на этом рисунке показана контурная диаграмма RSS (3.22), рассчитанной для разных возможных оценок коэффициентов регрессионной зависимости balance от limit и age. Каждый эллипс соответствует набору коэффициентов для заданного значения RSS, причем эллипсы, расположенные близко к центру, соответствуют наименьшим значениям RSS. Черная точка и прерывистые линии показывают оценки коэффициентов, которые сопровождаются наименьшей RSS, т. е. это оценки, полученные по методу наименьших квадратов. Оси limit и age были масштабированы таким образом, чтобы изобразить на графике все возможные оценки коэффициентов, которые находятся на расстоянии вплоть до четырех стандартных ошибок по обе стороны от оценок, полученных по методу наименьших квадратов. Таким образом, график изображает все правдоподобные значения коэффициентов. Например, мы видим, что истинное значение коэффициента для limit почти наверняка лежит где–то между 0.15 и 0.20.

В то же время на рис. 3.15 справа показана контурная диаграмма RSS для возможных оценок коэффициентов регрессионной зависимости balance от rating и limit — предикторов, которые, как мы знаем, высоко коллинеарны. Теперь эллипсы принимают очень узкую форму: имеет место большой размах значений оценок коэффициентов с одинаковыми значениями RSS. Следовательно, небольшое изменение в данных может заставить пару значений коэффициентов, которые обеспечивают наимень-

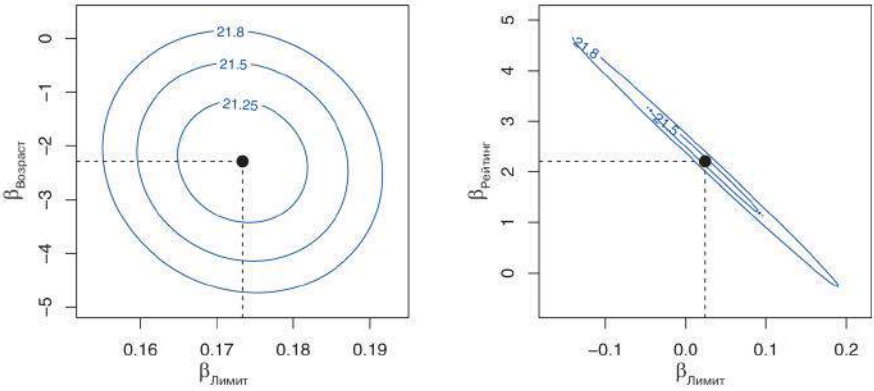

РИСУНОК 3.15. *Контурные диаграммы для значений RSS как функции от параметров β различных регрессионных моделей, рассчитанных по данным* `Credit`. *На каждом графике черная точка соответствует значениям коэффициентов с наименьшей RSS.* Слева: *контурная диаграмма RSS для регрессионной зависимости баланса на кредитной карте (*`balance`*) от возраста клиента (*`age`*) и кредитного лимита (*`limit`*). Минимальное значение хорошо определено.* Справа: *контурная диаграмма RSS для регрессионной зависимости* `balance` *от* `rating` *(рейтинг клиента) и* `limit`. *Из–за наличия коллинеарности имеет место большое количество пар $(\beta_{Лимит}, \beta_{Рейтинг})$ с одинаковым значением RSS*

шую RSS (т. е. оценки, полученные по методу наименьших квадратов), переместиться куда–угодно вдоль эллипса. Это приводит к очень большой неопределенности в отношении оценок коэффициентов. Заметьте, что шкала значений коэффициента для `limit` теперь изменяется от -0.2 до 0.2 — это восьмикратное увеличение диапазона вероятных значений данного коэффициента, по сравнению с регрессионной моделью, включающей `age`. Интересно, что хотя по отдельности коэффициенты `limit` и `rating` теперь обладают гораздо большей неопределенностью, вместе они почти наверняка будут находится в пределах изображенной узкой области. Например, не стоит ожидать, что истинные значения коэффициентов `limit` и `rating` составят -0.1 и 1 соответственно, хотя для каждого коэффициента в отдельности эти значения возможны.

Поскольку коллинеарность снижает точность оценок регрессионных коэффициентов, она вызывает увеличение стандартной ошибки $\hat{\beta}_j$. Вспомните, что t–критерий для каждого предиктора рассчитывается путем деления $\hat{\beta}_j$ на соответствующую стандартную ошибку. Следовательно, коллинеарность приводит к снижению t–критерия. В результате этого в присутствии коллинеарности у нас может не получиться отклонить $H_0 : \beta_j = 0$. Это означает, что из–за коллинеарности *мощность* статистического теста, т. е. вероятность безошибочного обнаружения *ненулевого* коэффициента, снижается.

мощность

Табл. 3.11 сравнивает оценки коэффициентов двух отдельных множественных регрессионных моделей. Первая представляет собой регрессию

ТАБЛИЦА 3.11. *Приведены результаты для двух регрессионных моделей, построенных по данным* Credit. *Модель M1 — это регрессия* balance *по* age *и* limit, *а M2 — регрессия* balance *по* rating *и* limit. *Стандартная ошибка* $\hat{\beta}_{\texttt{limit}}$ *во второй модели выше в 12 раз из–за коллинеарности*

Модель	Параметр	Коэффициент	Ст. ошибка	t	p
M1	Свободный член	−173.411	43.828	−3.957	< 0.0001
	age	−2.292	0.672	−3.407	0.0007
	limit	0.173	0.005	34.496	< 0.0001
M2	Свободный член	−377.537	45.254	−8.343	< 0.0001
	rating	2.202	0.952	2.312	0.0213
	limit	0.025	0.064	0.384	0.7012

balance по age и limit, а вторая — регрессию balance по rating и limit. В первой модели как age, так и limit высокозначимы и имеют очень низкие p–значения. Во второй модели коллинеарность между limit и rating привела к увеличению стандартной ошибки оценки коэффициента limit в 12 раз, а p–значение увеличилось до 0.701. Другими словами, из–за наличия коллинеарности роль переменной limit была замаскирована. Во избежание подобной ситуации при подгонке модели желательно обнаружить и устранить потенциальные проблемы, связанные с коллинеарностью.

Простой способ обнаружения коллинеарности заключается в изучении корреляционной матрицы предикторов. Высокое (по модулю) значение в такой матрице указывает на наличие пары высоко коррелирующих переменных, а следовательно, и на наличие в данных проблемы коллинеарности. К сожалению, не все связанные с коллинеарностью проблемы можно обнаружить путем инспектирования корреляционной матрицы: коллинеарность может существовать между тремя и более переменными, даже когда не установлено ни одной существенно высокой парной корреляции. Мы называем такую ситуацию *мультиколлинеарностью*. Вместо инспектирования корреляционной матрицы более надежным способом обнаружения мультиколлинеарности является вычисление *фактора инфляции дисперсии* (VIF[13]). VIF представляет собой отношение дисперсии $\hat{\beta}_j$ в полной модели к дисперсии $\hat{\beta}_j$ в модели, содержащей только один этот предиктор j. Наименьшее возможное значение VIF составляет 1 и указывает на полное отсутствие коллинеарности. На практике обычно имеет место небольшая степень коллинеарности между предикторами. В качестве эмпирического правила можно считать, что VIF, превышающий 5 или 10, указывает на излишне высокую степень коллинеарности. VIF для каждой переменной можно вычислить по следующей формуле:

мультиколлинеарность

$$\mathrm{VIF}(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}},$$

где $R^2_{X_j|X_{-j}}$ — это R^2 из регрессии X_j по всем остальным предикторам. Если значение $R^2_{X_j|X_{-j}}$ близко к 1, то коллинеарность присутствует и поэтому VIF будет большим.

[13] Аббревиатура от «variance inflation factor». — *Прим. пер.*

В случае с данными Credit регрессия balance по age, rating и limit показывает, что эти предикторы имеют VIF, равные 1.01, 160.67 и 160.59 соответственно. Как мы и ожидали, в данных есть значительная коллинеарность!

При столкновении с проблемой коллинеарности можно воспользоваться двумя простыми решениями. Первое заключается в удалении проблематичных переменных из регрессии. Обычно это можно сделать без значительного снижения качества модели, поскольку коллинеарность подразумевает, что заключенная в соответствующей переменной информация об отклике в присутствии других переменных избыточна. Например, если мы построим регрессию balance по age и limit, исключив предиктор rating, то итоговые значения VIF будут близки к минимально возможному значению — 1, а R^2 снизится с 0.754 до 0.75. Поэтому исключение rating из совокупности предикторов успешно решит проблему коллинеарности без снижения качества модели. Второе решение заключается в объединении коллинеарных переменных в один предиктор. Например, мы могли бы взять среднее значение стандартизованных версий limit и rating для создания новой переменной, которая измеряет *кредитоспособность*.

3.4 Маркетинговый план

Теперь мы вкратце вернемся к семи вопросам в отношении данных Advertising, которые мы задали в начале этой главы.

1. *Есть ли зависимость между затратами на рекламу и объемом продаж?*

   На этот вопрос можно ответить, построив множественную регрессию sales по TV, radio и newspaper как в (3.20) и проверив гипотезу $H_0 : \beta_{\texttt{TV}} = \beta_{\texttt{radio}} = \beta_{\texttt{newspaper}} = 0$. В подразделе 3.2.2 мы показали, что о необходимости отклонения этой нулевой гипотезы можно судить по F–критерию. В данном случае p–значение для F–критерия очень низкое (табл. 3.6), что является надежным свидетельством существования связи между объемом продаж и рекламой.

2. *Насколько сильна связь между затратами на рекламу и объемом продаж?*

   В подразделе 3.1.3 мы обсудили два показателя точности модели. Во–первых, RSE оценивает стандартное отклонение отклика от истинной линии регрессии. Для данных Advertising RSE составляет 1681 единицу, тогда как среднее значение отклика равно 14 022, что указывает на удельную ошибку около 22%. Во–вторых, коэффициент детерминации R^2 выражает долю дисперсии отклика, объясненную предикторами. Предикторы объясняют почти 90% дисперсии sales. RSE и статистика R^2 приведены в табл. 3.6.

3. *Какой тип рекламы способствует продажам?*

   Для ответа на этот вопрос мы можем проанализировать p–значения, связанные с t–критерием каждого предиктора (см. подраздел 3.1.2). В множественной линейной регрессии из табл. 3.4 p–значения для TV

и radio низки, а p–значение для newspaper — нет. Это говорит о том, что на sales влияет только реклама на телевидении и на радио. В главе 6 мы рассмотрим этот вопрос подробнее.

4. *Каков размер эффекта каждого типа рекламы на продажи?*

   В подразделе 3.1.2 мы видели, что при помощи стандартной ошибки $\hat{\beta}_j$ можно сконструировать доверительные интервалы для β_j. В случае с данным Advertising мы имеем следующие доверительные интервалы: (0.043, 0.049) для TV, (0.172, 0.206) для radio и (−0.013, 0.011) для newspaper. Доверительные интервалы для TV и radio узкие и далеки от нуля, что указывает на связь рекламы в этих СМИ с количеством продажам. Однако интервал для newspaper содержит ноль, указывая на то, что эта переменная не является статистически значимой при заданных значениях TV и radio.

   В подразделе 3.3.3 мы видели, что коллинеарность может привести к очень большим стандартным ошибкам. Могла ли коллинеарность стать причиной очень широкого доверительного интервала для newspaper? Значения VIF для TV, radio и newspaper составляют 1.005, 1.145 и 1.145 соответственно, что указывает на отсутствие коллинеарности.

   Для оценки степени влияния каждого типа рекламы на продажи мы можем отдельно рассчитать три простые линейные регрессии. Результаты приведены в табл. 3.1 и 3.3. Есть свидетельство очень тесной связи между TV и sales, а также между radio и sales. При игнорировании значений TV и radio наблюдается умеренно выраженная связь между newspaper и sales.

5. *Насколько точно мы можем спрогнозировать будущие продажи?*

   Отклик можно предсказать при помощи уравнения (3.21). Точность этой оценки зависит от того, хотим ли мы предсказать отдельное значение отклика — $Y = f(X) + \epsilon$, или его среднее значение — $f(X)$ (см. подраздел 3.2.2). В первом случае мы применяем интервал предсказаний, а во втором — доверительный интервал. Интервалы предсказаний всегда будут шире доверительных интервалов, поскольку они учитывают неопределенность, связанную с ϵ — неустранимой ошибкой.

6. *Является ли связь линейной?*

   В подразделе 3.3.3 мы видели, что для обнаружения нелинейности можно использовать графики остатков. Если зависимость является линейной, то в расположении остатков не должно наблюдаться никакой закономерности. В случае с данными Advertising на рис. 3.5 мы наблюдаем нелинейную зависимость, хотя эту зависимость можно было бы увидеть также и на графике остатков. В подразделе 3.3.2 мы обсудили добавление преобразованных предикторов в линейную регрессионную модель для учета подобных нелинейных зависимостей.

7. *Существует ли синергия между разными типами рекламы?*

Стандартная линейная регрессионная модель подразумевает аддитивную связь между предикторами и откликом. Аддитивную модель легко интерпретировать, поскольку эффект одного предиктора на отклик не зависит от значений других предикторов. Однако допущение аддитивности может оказаться нереалистичным для некоторых наборов данных. В подразделе 3.3.2 мы показали, как добавить в регрессионную модель эффект взаимодействия для учета неаддитивных зависимостей. Низкое p–значение эффекта взаимодействия указывает на наличие подобных зависимостей. По рис. 3.5 следовало, что данные `Advertising` могут обладать свойством неаддитивности. Включение в модель эффекта взаимодействия приводит к существенному увеличению R^2 — примерно с 90% до почти 97%.

3.5 Сравнение линейной регрессии с методом K ближайших соседей

Как обсуждалось в главе 2, линейная регрессия представляет собой пример *параметрического* метода, поскольку она предполагает линейную функциональную форму для $f(X)$. Параметрические методы имеют несколько преимуществ. Обычно подгонка таких моделей не составляет труда, т. к. требуется оценить только небольшой набор коэффициентов. В случае линейной регрессии коэффициенты имеют простую интерпретацию, и можно легко выполнить проверку их статистической значимости. Однако параметрические методы все же обладают недостатком: в силу своего устройства они делают строгие предположения относительно формы $f(X)$. Если выбранная функциональная форма далека от истинной, а наша цель заключается в получении точных предсказаний, то параметрический метод сработает плохо. Например, если мы предположим линейную связь между X и Y, а истинная зависимость существенно отличается от линейной, то полученная модель будет плохо соответствовать данным и любые сделанные на ее основе выводы будут ненадежными.

В то же время непараметрические методы не делают однозначного предположения относительно формы $f(X)$ и, следовательно, предоставляют альтернативный и более гибкий подход для выполнения регрессионного анализа. В этой книге мы обсуждаем разные непараметрические подходы. Здесь мы рассмотрим один из наиболее простых и хорошо известных непараметрических методов — *метод K ближайших соседей* (KNN–регрессия[14]). KNN–регрессия очень похожа на KNN–классификатор, описанный в главе 2. Для некоторого заданного значения K и прогнозируемого наблюдения x_0 KNN–регрессия сначала находит K обучающих наблюдений (обозначаются как $\mathcal{N}_0$), которые ближе всего расположены к x_0. Затем она оценивает $f(x_0)$, усредняя значения отклика всех обучающих наблюдений из $\mathcal{N}_0$. Другими словами,

метод K ближайших соседей

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} y_i.$$

[14] Аббревиатура от «K–nearest neighbors». — *Прим. пер.*

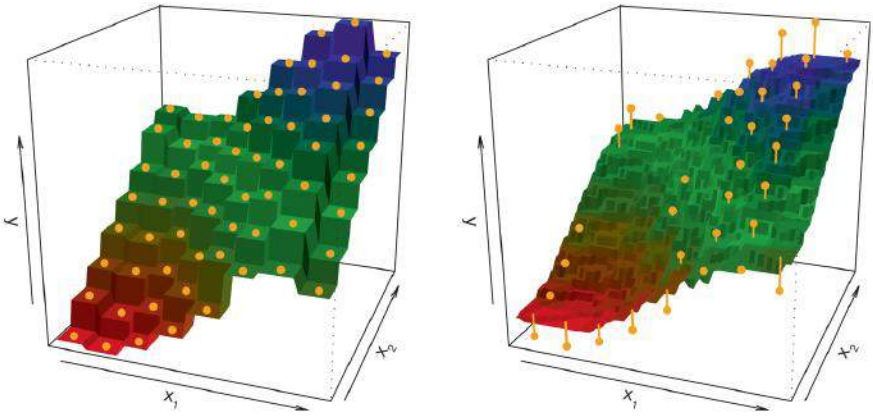

РИСУНОК 3.16. *Графики* $\hat{f}(X)$, *полученные путем подгонки KNN–регрессии к двумерным данным с 64 наблюдениями (оранжевые точки).* Слева: $K = 1$ *приводит к грубой ступенчатой функции.* Справа: $K = 9$ *приводит к гораздо более гладкой функции*

На рис. 3.16 показаны две модели KNN, подогнанные к данным с двумя предикторами. Модель с $K = 1$ показана на графике слева, а график справа соответствует модели с $K = 9$. Мы видим, что при $K = 1$ модель KNN идеально интерполирует обучающие наблюдения и, следовательно, принимает вид ступенчатой функции. При $K = 9$ модель KNN все еще является ступенчатой функцией, однако усреднение по девяти наблюдениям приводит к гораздо меньшим областям с одинаковыми прогнозными значениями, а следовательно, и к более гладкой функции. В целом оптимальное значение K будет зависеть от *компромисса между смещением и дисперсией*, который мы описали в главе 2. Небольшое значение K обеспечивает наиболее гибкую модель, которая будет обладать низким смещением, но высокой дисперсией. Эта дисперсия обусловлена тем фактом, что предсказание в некоторой конкретной области полностью определяется лишь одним наблюдением. В то же время более высокие значения K обеспечивают более гладкую и менее изменчивую модель: прогнозное значение в некоторой области является результатом усреднения нескольких точек, в связи с чем изменение одного наблюдения имеет незначительный эффект. Однако сглаживание может замаскировать некоторые особенности структуры $f(X)$. В главе 5 мы рассмотрим несколько методов для оценивания величины ошибок предсказаний. Эти методы можно использовать для нахождения оптимального значения K для KNN–регрессии.

В каких условиях параметрический метод, такой как обычная линейная регрессия, будет превосходить непараметрический метод, такой как KNN–регрессия? Ответ прост: *параметрический метод превзойдет непараметрический метод, если выбранная параметрическая форма будет близка к истинной форме* f. На рис. 3.17 показан пример данных, сгенерированных на основе одномерной линейной регрессионной модели. Черные сплошные линии соответствуют $f(X)$, тогда как синие линии показывают KNN–модели, подогнанные с использованием $K = 1$ и $K = 9$. В этом

случае предсказания при $K = 1$ слишком изменчивы, тогда как более гладкая модель с $K = 9$ намного ближе к истинной $f(X)$. Однако в связи с тем, что истинная зависимость является линейной, непараметрическому методу сложно конкурировать с линейной регрессией: непараметрический метод сопровождается увеличением дисперсии, которое не компенсируется пониженным смещением. Слева на рис. 3.18 синяя прерывистая линия показывает линейную регрессию, подогнанную к тем же данным. Это почти идеальная модель. Справа на рис. 3.18 видно, что на этих данных линейная регрессия превосходит метод KNN. Зеленая сплошная линия, показанная как функция от $1/K$, описывает изменение среднеквадратичной ошибки (MSE) KNN–регрессии на контрольной выборке. Кривая ошибок KNN–регрессии находится намного выше черной прерывистой линии, которая соответствует MSE на контрольной выборке у линейной регрессии. При высоком значении K метод KNN работает несколько хуже линейной регрессии (в смысле MSE). При небольших K он работает намного хуже.

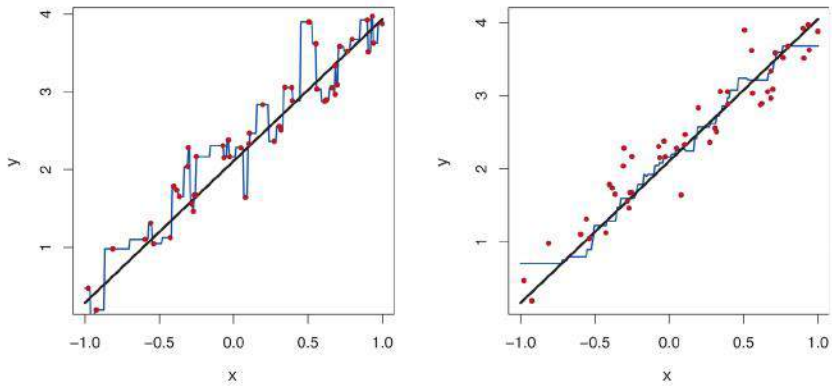

РИСУНОК 3.17. *Графики $\hat{f}(X)$, полученные путем подгонки KNN–регрессии к одномерному набору данных со 100 наблюдениями. Истинная зависимость показана черной сплошной линией.* Слева: *синяя линия соответствует $K = 1$ и интерполирует (т. е. проходит через) обучающие наблюдения.* Справа: *синяя линия соответствует $K = 9$ и представляет собой более гладкую модель*

На практике истинная зависимость Y от X редко бывает в точности линейной. Рисунок 3.19 сравнивает относительную эффективность линейной регрессии и метода KNN при возрастающей степени нелинейности связи между X и Y. В верхнем ряду графиков истинная зависимость почти линейна. В этом случае мы видим, что MSE на контрольных данных у линейной регрессии лучше, чем у KNN–регрессии с низкими значениями K. Однако при $K \geq 4$ KNN превосходит линейную регрессию. Во втором ряду показано более существенное отклонение от линейности. В этой ситуации метод KNN значительно превосходит линейную регрессию при всех значениях K. Заметьте, что по мере увеличения степени нелинейности MSE на контрольной выборке у непараметрического метода KNN почти не изменяется, тогда как MSE на контрольной выборке у линейной регрессии демонстрирует существенный рост.

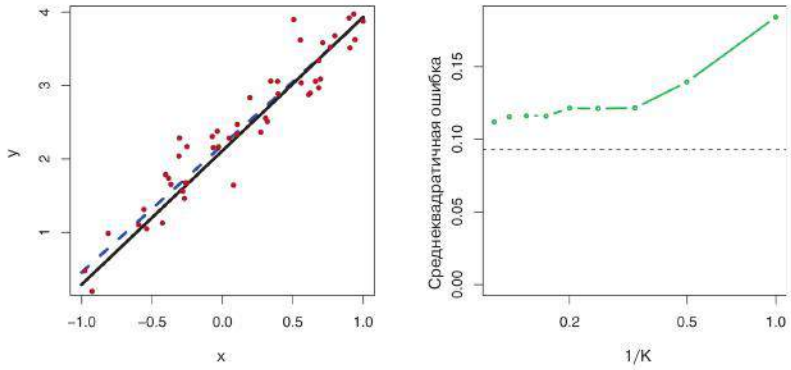

РИСУНОК 3.18. *Набор данных, показанный на рис. 3.17, исследован более подробно.* Слева: *синяя линия изображает модель, подогнанную по методу наименьших квадратов. Поскольку* $f(X)$ *действительно является линейной функцией (показана черной прерывистой линией), линейная регрессия обеспечивает очень хорошую оценку* $f(X)$. Справа: *горизонтальная прерывистая линия изображает MSE линейной регрессии на контрольной выборке, а зеленая сплошная линия соответствует MSE KNN–регрессии в зависимости от* $1/K$ *(на логарифмической шкале). Линейная регрессия достигает более низкой ошибки на контрольных данных по сравнению с KNN–регрессией, поскольку* $f(X)$ *в действительности является линейной функцией. Наилучший результат для KNN–регрессии имеет место при очень большом значении* K*, которое соответствует небольшому значению* $1/K$

Рисунки 3.18 и 3.19 иллюстрируют ситуации, когда метод KNN работает несколько хуже линейной регрессии при линейной истинной зависимости и намного лучше при нелинейных зависимостях. На практике, когда истинная зависимость неизвестна, можно было бы заключить, что метод KNN следует предпочесть линейной регрессии, поскольку он сработает лишь незначительно хуже в случае линейной истинной зависимости, но в то же время может дать намного более высокие результаты при нелинейной истинной зависимости. Однако в действительности даже при выраженной нелинейной истинной зависимости метод KNN может дать менее качественные решения, по сравнению с линейной регрессией. В частности, на рис. 3.18 и 3.19 показаны ситуации с $p = 1$ предиктором. При более высоких размерностях метод KNN часто работает хуже линейной регрессии.

На рис. 3.20 представлена та же ситуация с выраженной нелинейностью, как и во втором ряду на рис. 3.19, однако мы добавили дополнительные *шумовые* предикторы, которые не связаны с откликом. При $p = 1$ или $p = 2$ метод KNN превосходит линейную регрессию. Однако при $p = 3$ результаты неоднозначны, а при $p \geq 4$ линейная регрессия превосходит KNN. Более того, увеличение размерности вызвало лишь небольшое увеличение MSE на контрольной выборке у линейной регрессии, но более чем десятикратное увеличение MSE у KNN–регрессии. Подобное снижение качества по мере увеличения размерности является обычной проблемой метода KNN и обусловлено тем, что при высоких размерностях, по сути,

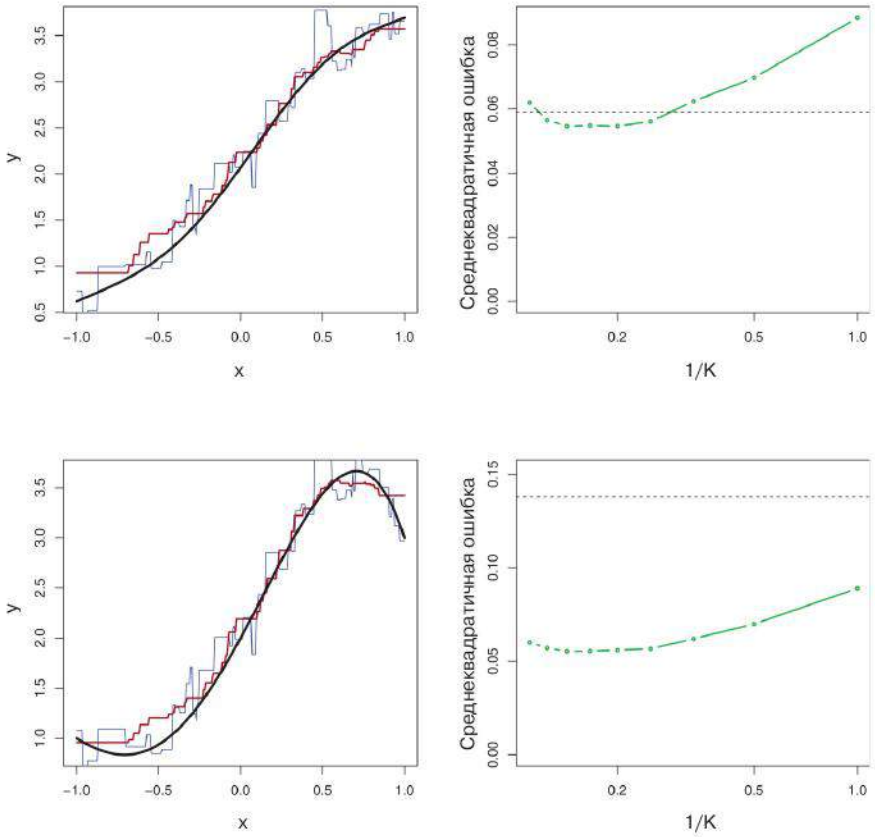

РИСУНОК 3.19. Слева вверху: *приведены KNN–модели с $K = 1$ (голубая кривая) и $K = 9$ (красная кривая) для случая со слабой нелинейной зависимостью между X и Y (черная сплошная кривая).* Справа вверху: *показаны MSE на контрольной выборке для линейной регрессии по методу наименьших квадратов (горизонтальная черная линия) и для KNN–моделей с разными значениями $1/K$ (зеленая кривая), подогнанных к данным со слабой нелинейной зависимостью.* Слева и справа внизу: *то же, что и вверху, но для случая с выраженной нелинейной зависимостью между X и Y*

происходит снижение объема выборки. В рассматриваемом наборе данных имеется 100 обучающих наблюдений; при $p = 1$ это дает достаточно информации, чтобы верно оценить $f(X)$. Однако распределение 100 наблюдений в 20–мерном пространстве приводит к явлению, когда то или иное наблюдение не имеет *близлежащих соседей*, — это так называемое *проклятие размерности*. Иными словами, K наблюдений, являющихся ближайшими соседями контрольного наблюдения x_0, при большом p могут находиться очень далеко от x_0 в p–мерном пространстве, что приводит к очень плохому предсказанию $f(x_0)$, а следовательно, и плохому качеству модели KNN. Как правило, параметрические методы превосходят непараметрические методы при небольшом числе наблюдений, приходящихся на один предиктор.

проклятие размерности

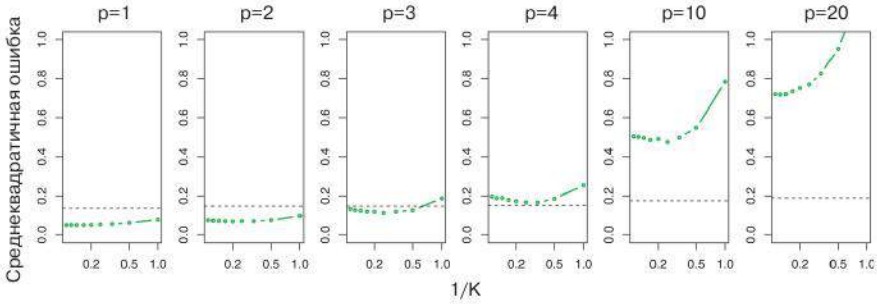

РИСУНОК 3.20. *MSE на контрольной выборке у линейной регрессии (черные прерывистые линии) и KNN–регрессии (зеленые кривые) при возрастающем числе переменных p. Истинная функция нелинейна по первой переменной (как на рис. 3.19) и не зависит от дополнительных переменных. Качество линейной регрессии при наличии этих дополнительных шумовых переменных снижается медленно, тогда как качество KNN–регрессии при увеличении p падает намного быстрее*

Даже в задачах малой размерности мы можем предпочесть линейную регрессию методу KNN из–за легкости интерпретации. Если MSE на контрольной выборке у KNN–регрессии лишь незначительно ниже ошибки линейной регрессии, то, возможно, имеет смысл пожертвовать точностью прогноза в пользу более простой модели, которую можно описать при помощи всего нескольких коэффициентов с низкими p–значениями.

3.6 Лабораторная работа: линейная регрессия

3.6.1 Библиотеки

Функция `library()` используется для загрузки библиотек, или групп функций и наборов данных, которые не включены в базовый дистрибутив R[15]. Обычные функции, выполняющие расчет регрессии по методу

`library()`

[15] К сожалению, имя функции `library()` выбрано не совсем удачно и поначалу вводит в заблуждение. Она загружает *пакеты* (packages), а не *библиотеки* (libraries).

наименьших квадратов и другие анализы, входят в состав стандартного дистрибутива R, однако более экзотические функции требуют дополнительных библиотек. Здесь мы загружаем пакет MASS, который представляет собой очень большую коллекцию наборов данных и функций. Кроме того, мы загружаем пакет ISLR, который содержит наборы данных для этой книги.

```
> library(MASS)
> library(ISLR)
```

Если вы получаете сообщение об ошибке при загрузке любой из этих библиотек, это, скорее всего, указывает на то, что соответствующая библиотека еще не инсталлирована на вашей системе. Некоторые библиотеки, такие как MASS, поставляются вместе с R и не требуют отдельной инсталляции на вашем компьютере. Однако другие пакеты, такие как ISLR, должны быть загружены перед их первым использованием. Это можно сделать непосредственно из R. Например, на компьютере с операционной системой Windows выберите опцию Install package из закладки Packages. После того как вы выберете тот или иной сайт–зеркало, появится список доступных пакетов. Просто выберите пакет, который вы хотите установить, и R автоматически загрузит его. В качестве альтернативы это можно сделать из командной строки R, выполнив install.packages("ISLR"). Такая инсталляция должна быть выполнена только перед первым использованием пакета. Однако функцию library() необходимо вызывать каждый раз, когда вы хотите воспользоваться соответствующим пакетом.

3.6.2 Простая линейная регрессия

Библиотека MASS содержит набор данных Boston, который включает значения medv (медианная стоимость дома) для 506 окрестностей Бостона. Мы попытаемся предсказать medv на основе 13 предикторов, таких как rm (среднее количество комнат в доме), age (средний возраст дома) и lstat (процент домохозяйств с низким социоэкономическим статусом).

```
> fix(Boston)
> names(Boston)
[1]  "crim"  "zn"    "indus"  "chas"    "nox"    "rm"     "age"
[8]  "dis"   "rad"   "tax"    "ptratio" "black"  "lstat"  "medv"
```

Для получения дополнительной информации по этому набору данных, мы можем набрать команду ?Boston.

Мы начнем с использования функции lm() для подгонки простой линейной регрессионной модели, в которой medv будет откликом, а lstat — предиктором. Основной синтаксис выглядит как lm(y ∼ x, data), где y — это зависимая переменная, x — предиктор, а data — набор данных, в котором содержатся эти переменные.

```
> lm.fit = lm(medv ∼ lstat)
Error in eval(expr, envir, enclos) : Object "medv" not found
```

Строго говоря, пакет — это «группа функций и наборов данных, которые не включены в базовый дистрибутив R». Под библиотекой же обычно понимают директорию, в которой инсталлированы пакеты. В этой книге авторы используют термины «пакет» и «библиотека» в качестве синонимов. — *Прим. пер.*

Эта команда вызывает ошибку, поскольку R не знает, где найти переменные medv и lstat. Следующая строка говорит R о том, что эти переменные находятся в таблице Boston. Если мы прикрепим таблицу Boston[16], то первая команда сработает правильно, поскольку теперь R распознает необходимые переменные.

```
> lm.fit = lm(medv ~ lstat, data = Boston)
> attach(Boston)
> lm.fit = lm(medv ~ lstat)
```

При вводе lm.fit будет выведена основная информация по модели. Для получения более детальной информации мы применяем summary(lm.fit). Это даст нам p–значения и стандартные ошибки коэффициентов, а также коэффициент детерминации R^2 и F–критерий для модели.

```
> lm.fit

Call:
lm(formula = medv ~ lstat)

Coefficients:
(Intercept)       lstat
      34.55       -0.95

> summary(lm.fit)

Call:
lm(formula = medv ~ lstat)

Residuals:
   Min      1Q  Median      3Q     Max
-15.17   -3.99   -1.32    2.03   24.50

Coefficients:
             Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)    34.5538      0.5626     61.4    <2e-16 ***
lstat          -0.9500      0.0387    -24.5    <2e-16 ***
---
Signif. codes:    0 ***  0.001 **  0.01 *  0.05 . 0.1   1

Residual standard error: 6.22 on 504 degrees of freedom
Multiple R-squared: 0.544,    Adjusted R-squared: 0.543
F-statistic:  602 on 1 and 504 DF,  p-value: <2e-16
```

Мы можем воспользоваться функцией names(), чтобы выяснить, что еще хранится в объекте lm.fit. Хотя мы можем извлечь эти количественные показатели по их именам (например, lm.fit$coefficients), для получения доступа к ним надежнее будет использовать функции–экстракторы вроде coef(). names() coef()

[16] Имеется в виду операция, в результате которой содержимое объекта Boston будет добавлено в т. н. *поисковый путь* R (search path). — *Прим. пер.*

```
> names(lm.fit)
[1]  "coefficients"   "residuals"      "effects"
[4]  "rank"           "fitted.values"  "assign"
[7]  "qr"             "df.residual"    "xlevels"
[10] "call"           "terms"          "model"

> coef(lm.fit)
(Intercept)       lstat
      34.55       -0.95
```

confint() Для вычисления доверительных интервалов оценок коэффициентов мы можем применить команду `confint()`.

```
> confint(lm.fit)
              2.5 %  97.5 %
(Intercept)  33.45  35.659
lstat        -1.03 -0.874
```

Функцию `predict()` можно использовать для расчета доверительных интервалов и интервалов предсказаний при прогнозировании `medv` по заданному значению `lstat`.

```
> predict(lm.fit, data.frame(lstat = (c(5, 10, 15))),
          interval = "confidence")
     fit    lwr    upr
1  29.80  29.01  30.60
2  25.05  24.47  25.63
3  20.30  19.73  20.87

> predict(lm.fit, data.frame(lstat = (c(5, 10, 15))),
          interval ="prediction")
     fit     lwr    upr
1  29.80  17.57   42.04
2  25.05  12.83   37.28
3  20.30   8.08   32.53
```

Например, 95%–ный доверительный интервал, связанный со значением `lstat` = 10, составляет (24.47, 25.63), а 95%–ный интервал предсказания — (12.828, 37.28). Как и следовало ожидать, оба этих интервала центрированы относительно одной и той же точки (предсказанное значение `medv` = 25.05 при `lstat` = 10), однако интервал предсказания является намного более широким.

abline() Теперь мы изобразим `lstat` и `medv` вместе с регрессионной прямой, используя функции `plot()` и `abline()`.

```
> plot(lstat, medv)
> abline(lm.fit)
```

Имеется указание на некоторую нелинейность связи между `lstat` и `medv`. Мы исследуем данное обстоятельство позднее в этой лабораторной работе.

Функцию `abline()` можно использовать для изображения любой линии, не только линии наименьших квадратов. Чтобы нарисовать линию со свободным членом `a` и углом наклона `b` мы выполняем команду `abline(a,`

b). Ниже мы экспериментируем с некоторыми дополнительными настройками для изображения линий и точек. Команда `lwd = 3` увеличивает толщину регрессионной линии в 3 раза; это работает и в случае функций `plot()` и `lines()`. Мы можем также использовать опцию `pch`, чтобы выбрать различные символы для изображения точек.

```
> abline(lm.fit, lwd = 3)
> abline(lm.fit, lwd = 3, col = "red")
> plot(lstat, medv, col = "red")
> plot(lstat, medv, pch = 20)
> plot(lstat, medv , pch = "+")
> plot (1:20, 1:20, pch = 1:20)
```

Далее мы изучим некоторые диагностические графики, часть из которых обсуждалась в подразделе 3.3.3. Четыре таких графика автоматически создаются путем непосредственного применения функции `plot()` к результату работы функции `lm()`. Как правило, эта команда будет выводить только один график за раз, и при нажатии *Enter* будет выводиться следующий график. Однако часто удобнее бывает просмотреть все четыре графика одновременно. Мы можем сделать это при помощи функции `par()`, которая говорит `R` разделить графическое окно на отдельные панели так, чтобы можно было просмотреть несколько графиков одновременно. Например, `par(mfrow = c(2, 2))` разделяет графическое окно на сетку из 2×2 панелей.

`par()`

```
> par(mfrow = c(2, 2))
> plot(lm.fit)
```

В качестве альтернативы мы можем рассчитать остатки линейной регрессионной модели, используя функцию `residuals()`. Функция `rstudent()` выдаст стьюдентизированные остатки, и мы сможем использовать ее для построения графика зависимости этих остатков от подогнанных значений.

`residuals()`
`rstudent()`

```
> plot(predict(lm.fit), residuals(lm.fit))
> plot(predict(lm.fit), rstudent(lm.fit))
```

Судя по графикам остатков, имеет место некоторая нелинейность. Значения показателя разбалансировки можно рассчитать для любого количества предикторов при помощи функции `hatvalues()`.

`hatvalues()`

```
> plot(hatvalues(lm.fit ))
> which.max(hatvalues(lm.fit))
375
```

Функция `which.max()` определяет индексный номер наибольшего элемента некоторого вектора. В данном случае она говорит нам, какое наблюдение имеет максимальное значение показателя разбалансировки.

`which.max()`

3.6.3 Множественная линейная регрессия

Для подгонки множественной линейной регрессии по методу наименьших квадратов мы снова воспользуемся функцией `lm()`. Синтаксис `lm(y ~ x1 + x2 + x3)` применяется для подгонки модели с тремя предикторами —

x1, x2 и x3. Функция summary() теперь возвращает регрессионные коэффициенты для всех предикторов.

```
> lm.fit = lm(medv ~ lstat + age, data = Boston)
> summary(lm.fit)

Call:
lm(formula = medv ~ lstat + age, data = Boston)

Residuals :
    Min      1Q  Median      3Q     Max
-15.98   -3.98   -1.28    1.97   23.16

Coefficients:
             Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)   33.2228      0.7308    45.46    <2e-16 ***
lstat         -1.0321      0.0482   -21.42    <2e-16 ***
age            0.0345      0.0122     2.83    0.0049 **
---
Signif. codes:   0 ***  0.001 **  0.01 *  0.05 . 0.1 1

Residual standard error: 6.17 on 503 degrees of freedom
Multiple R-squared: 0.551,    Adjusted R-squared: 0.549
F-statistic:  309 on 2 and 503 DF, p-value: <2e-16
```

Набор данных Boston содержит 13 переменных, и поэтому было бы утомительно набирать их названия для выполнения регрессионного анализа на основе всех предикторов. Вместо этого мы можем использовать следующий сокращенный вариант:

```
> lm.fit = lm(medv ~ ., data = Boston)

> summary(lm.fit)

Call:
lm(formula = medv ~ ., data = Boston)

Residuals :
    Min       1Q   Median       3Q      Max
-15.594   -2.730   -0.518    1.777   26.199

Coefficients:
              Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  3.646e+01   5.103e+00    7.144  3.28e-12 ***
crim        -1.080e-01   3.286e-02   -3.287  0.001087 **
zn           4.642e-02   1.373e-02    3.382  0.000778 ***
indus        2.056e-02   6.150e-02    0.334  0.738288
chas         2.687e+00   8.616e-01    3.118  0.001925 **
nox         -1.777e+01   3.820e+00   -4.651  4.25e-06 ***
rm           3.810e+00   4.179e-01    9.116   < 2e-16 ***
age          6.922e-04   1.321e-02    0.052  0.958229
dis         -1.476e+00   1.995e-01   -7.398  6.01e-13 ***
```

```
rad          3.060e-01   6.635e-02    4.613  5.07e-06 ***
tax         -1.233e-02   3.761e-03   -3.280  0.001112 **
ptratio     -9.527e-01   1.308e-01   -7.283  1.31e-12 ***
black        9.312e-03   2.686e-03    3.467  0.000573 ***
lstat       -5.248e-01   5.072e-02  -10.347   < 2e-16 ***
---
Signif. codes: 0 ***  0.001 **  0.01 *  0.05 .  0.1  1

Residual standard error:  4.745 on 492 degrees of freedom
Multiple R-Squared:  0.7406,     Adjusted R-squared:  0.7338
F-statistic:  108.1 on 13 and 492 DF, p-value: < 2.2e-16
```

Мы можем получить доступ к отдельным компонентам объекта с результатами анализа по их имени (наберите `?summary.lm` для получения списка доступных компонентов). Следовательно, `summary(lm.fit)$r.sq` даст нам R^2, а `summary(lm.fit)$sigma` даст нам RSE. Функцию `vif()` из пакета `car` можно применить для вычисления значений фактора инфляции дисперсии. Для этого набора данных большинство VIF имеет низкие или умеренные значения. Пакет `car` не является частью базового дистрибутива `R`, и поэтому перед первым использованием его нужно установить при помощи функции `install.packages()`.

```
> library(car)
> vif(lm.fit)
  crim      zn   indus    chas     nox      rm     age
  1.79    2.30    3.99    1.07    4.39    1.93    3.10
   dis     rad     tax ptratio   black   lstat
  3.96    7.48    9.01    1.80    1.35    2.94
```

Что, если мы захотели бы выполнить регрессионный анализ по всем переменным, за исключением одной? Например, в приведенных выше результатах `age` имеет высокое p–значение. Поэтому, возможно, мы хотим построить регрессию, исключив этот предиктор. Следующий синтаксис приводит к созданию регрессионной модели, включающей все переменные, кроме `age`.

```
> lm.fit1 = lm(medv ~ . -age, data = Boston)
> summary(lm.fit1)
...
```

В качестве альтернативы можно воспользоваться функцией `update()`. `update()`

```
> lm.fit1 = update(lm.fit, ~ . -age)
```

3.6.4 Эффекты взаимодействия

Включение эффектов взаимодействий в линейную модель при помощи функции `lm()` не составляет труда. Синтаксис `lstat:black` говорит `R` добавить эффект взаимодействия между `lstat` и `black`. Синтаксис `lstat*age` одновременно включает в качестве предикторов `lstat`, `age` и взаимодействие `lstat` × `age` — это сокращенная форма для `lstat + age + lstat:age`.

```
> summary(lm(medv ~ lstat * age, data = Boston))

Call:
lm(formula = medv ~ lstat * age, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max
-15.81   -4.04   -1.33    2.08   27.55

Coefficients:
             Estimate  Std. Error  t value   Pr(>|t|)
(Intercept) 36.088536    1.469835    24.55    < 2e-16 ***
lstat       -1.392117    0.167456    -8.31    8.8e-16 ***
age         -0.000721    0.019879    -0.04      0.971
lstat:age    0.004156    0.001852     2.24      0.025 *
---
Signif. codes: 0 ***  0.001 **  0.01 *  0.05 .  0.1  1

Residual standard error: 6.15 on 502 degrees of freedom
Multiple R-squared: 0.556,    Adjusted R-squared: 0.553
F-statistic:  209 on 3 and 502 DF,  p-value: <2e-16
```

3.6.5 Нелинейные преобразования предикторов

Функция `lm()` может принимать также нелинейно преобразованные предикторы. Например, имея предиктор X, мы можем создать предиктор X^2 при помощи `I(X^2)`. Функция `I()` необходима, поскольку `^` имеет особое значение в формуле; изоляция при помощи `I()` позволяет использовать `^` в обычном значении, т. е. возведение `X` в квадрат. Теперь мы построим регрессию `medv` по `lstat` и $\texttt{lstat}^2$.

```
> lm.fit2 = lm(medv ~ lstat + I(lstat^2))
> summary(lm.fit2)

Call:
lm(formula = medv ~ lstat + I(lstat^2))

Residuals:
    Min      1Q  Median      3Q     Max
-15.28   -3.83   -0.53    2.31   25.41

Coefficients:
             Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  42.86201     0.87208     49.1    <2e-16 ***
lstat        -2.33282     0.12380    -18.8    <2e-16 ***
I(lstat^2)    0.04355     0.00375     11.6    <2e-16 ***
---
Signif. codes: 0 ***  0.001 **  0.01 *  0.05 .  0.1  1

Residual standard error: 5.52 on 503 degrees of freedom
```

```
Multiple R-squared: 0.641,    Adjusted R-squared: 0.639
F-statistic:  449 on 2 and 503 DF,  p-value: <2e-16
```

Близкое к нулю p–значение квадратичного члена говорит о том, что его включение приводит к более качественной модели. Воспользуемся функцией `anova()` для дальнейшего количественного описания степени преимущества квадратичной модели над линейной. anova()

```
> lm.fit = lm(medv ~ lstat)
> anova(lm.fit, lm.fit2)
Analysis of Variance Table

Model 1: medv ~ lstat
Model 2: medv ~ lstat + I(lstat^2)
  Res.Df    RSS  Df  Sum of Sq    F  Pr(>F)
1    504  19472
2    503  15347   1       4125  135  <2e-16 ***
---
Signif. codes: 0 ***  0.001 **  0.01 *  0.05 .  0.1   1
```

Здесь `Model 1` соответствует линейной модели, которая содержит только один предиктор (`lstat`), а `Model 2` — большей квадратичной модели с двумя предикторами (`lstat` и $\texttt{lstat}^2$). Функция `anova()` выполняет проверку определенной гипотезы для сравнения двух моделей. Эта нулевая гипотеза заключается в том, что обе модели описывают данные одинаково хорошо, а альтернативная гипотеза — в том, что полная модель[17] описывает данные лучше. Здесь F–критерий составляет 135, а его p–значение практически равно нулю. Это является четким свидетельством в пользу того, что модель с предикторами `lstat` и $\texttt{lstat}^2$ существенно превосходит модель, в которую входит только предиктор `lstat`. Это неудивительно, поскольку ранее мы наблюдали признаки нелинейной зависимости между `medv` и `lstat`. Если мы выполним

```
> par(mfrow = c(2, 2))
> plot(lm.fit2)
```

то увидим, что при добавлении $\texttt{lstat}^2$ в модель какая–либо различимая закономерность в расположении остатков исчезает.

Для создания кубической модели мы можем добавить предиктор в форме `I(X^3)`. Однако такой подход может стать трудоемким в случае полиномов более высокой степени. Более подходящим способом для создания полиномов при вызове `lm()` является использование функции `poly()`. poly() Например, следующая команда даст полином пятой степени:

```
> lm.fit5 = lm(medv ~ poly(lstat, 5))
> summary(lm.fit5)

Call:
lm(formula = medv ~ poly(lstat, 5))

Residuals:
```

[17] Под полной («full model») подразумевается модель, включающая все имеющиеся предикторы. — *Прим. пер.*

```
    Min      1Q  Median      3Q     Max
-13.543  -3.104  -0.705   2.084  27.115

Coefficients:
                Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)       22.533       0.232    97.20   < 2e-16 ***
poly(lstat, 5)1 -152.460       5.215   -29.24   < 2e-16 ***
poly(lstat, 5)2   64.227       5.215    12.32   < 2e-16 ***
poly(lstat, 5)3  -27.051       5.215    -5.19   3.1e-07 ***
poly(lstat, 5)4   25.452       5.215     4.88   1.4e-06 ***
poly(lstat, 5)5  -19.252       5.215    -3.69   0.00025 ***
---
Signif. codes: 0 ***  0.001 **  0.01 *  0.05 .  0.1   1
Residual standard error:  5.21 on 500 degrees of freedom
Multiple R-squared: 0.682,     Adjusted R-squared: 0.679
F-statistic: 214 on 5 and 500 DF,  p-value: <2e-16
```

Полученный результат свидетельствует о том, что включение дополнительных полиномиальных членов (вплоть до пятой степени) приводит к улучшению модели. Однако дальнейшее исследование данных показывает, что ни один из полиномиальных членов со степенью выше пяти не имеет достаточно низких p–значений при включении в модель.

Конечно, мы никоим образом не ограничены использованием только полиномиальных преобразований предикторов. Здесь мы пробуем логарифмирование:

```
> summary(lm(medv ~ log(rm), data = Boston))
...
```

3.6.6 Качественные предикторы

Теперь мы исследуем набор данных `Carseats` из библиотеки `ISLR`. Мы попробуем предсказать `Sales` (продажи детских автомобильных сидений) в 400 регионах на основе ряда предикторов.

```
> fix(Carseats)
> names(Carseats)
 [1] "Sales"       "CompPrice"   "Income"      "Advertising"
 [5] "Population"  "Price"       "ShelveLoc"   "Age"
 [9] "Education"   "Urban"       "US"
```

Данные `Carseats` включают категориальные предикторы, в частности `Shelveloc` — индикатор качества расположения стеллажа, т. е. место в каждом магазине, где выставлено сидение. Предиктор `Shelveloc` принимает три возможных значения: *Bad* (плохое), *Medium* (среднее) и *Good* (хорошее). Встречая качественную переменную вроде `Shelveloc`, `R` автоматически создает индикаторные переменные. Ниже мы подгоняем множественную регрессионную модель, которая включает эффекты взаимодействия между некоторыми переменными.

```
> lm.fit = lm(Sales ~ . + Income:Advertising + Price:Age,
              data = Carseats)
```

```
> summary(lm.fit)

Call:
lm(formula = Sales ~ . + Income:Advertising + Price:Age,
   data = Carseats)

Residuals:
   Min      1Q  Median      3Q     Max
-2.921  -0.750   0.018   0.675   3.341

Coefficients:
                   Estimate       Std. Error  t value  Pr(>|t|)
(Intercept)          6.575565    1.008747     6.52   2.2e-10 ***
CompPrice            0.092937    0.004118    22.57   < 2e-16 ***
Income               0.010894    0.002604     4.18   3.6e-05 ***
Advertising          0.070246    0.022609     3.11   0.00203 **
Population           0.000159    0.000368     0.43   0.66533
Price               -0.100806    0.007440   -13.55   < 2e-16 ***
ShelveLocGood        4.848676    0.152838    31.72   < 2e-16 ***
ShelveLocMedium      1.953262    0.125768    15.53   < 2e-16 ***
Age                 -0.057947    0.015951    -3.63   0.00032 ***
Education           -0.020852    0.019613    -1.06   0.28836
UrbanYes             0.140160    0.112402     1.25   0.21317
USYes               -0.157557    0.148923    -1.06   0.29073
Income:Advertising   0.000751    0.000278     2.70   0.00729 **
Price:Age            0.000107    0.000133     0.80   0.42381
---
Signif. codes: 0 ***  0.001 **  0.01 *  0.05 .  0.1   1

Residual standard error: 1.01 on 386 degrees of freedom
Multiple R-squared: 0.876,    Adjusted R-squared: 0.872
F-statistic: 210 on 13 and 386 DF, p-value: <2e-16
```

Функция `contrasts()` возвращает матрицу, которая показывает, как именно R кодирует индикаторные переменные.

```
> attach(Carseats)

> contrasts(ShelveLoc)
       Good Medium
Bad       0      0
Good      1      0
Medium    0      1
```

Для получения информации о других типах контрастов и о том, как их задать, воспользуйтесь командой `?contrasts`.

Программа R создала индикаторную переменную `ShelveLocGood`, которая принимает значение 1, если расположение стеллажа хорошее, и 0 в остальных случаях. Она также создала индикаторную переменную `ShelveLocMedium`, которая равна 1 при среднем качестве расположения стеллажа и 0 в остальных случаях. У каждой из этих двух индикаторных переменных плохое качество расположения соответствует нулю. Тот факт,

что коэффициент для ShelveLocGood в результатах расчета регрессионной модели является положительным, указывает на то, что хорошее расположение стеллажа связано с высокими продажами (по сравнению с плохим расположением). ShelveLocMedium имеет меньший положительный коэффициент, указывая на то, что среднее качество расположения сопровождается более высокими продажами, по сравнению с плохим расположением, но меньшими продажами, по сравнению с хорошим расположением.

3.6.7 Написание функций

Как мы уже видели, R поставляется с большим количеством полезных функций, и еще большее их количество доступно в составе библиотек для R. Однако часто нам будет интересно выполнить некоторую операцию, функции для которой нет. При таком сценарии мы можем написать свою собственную функцию. Например, ниже мы приводим пример простой функции LoadLibraries(), которая подключает библиотеки ISLR и MASS. При попытке вызова этой функции до того, как мы ее создали, R выдает ошибку.

```
> LoadLibraries
Error: object 'LoadLibraries' not found
> LoadLibraries()
Error: could not find function "LoadLibraries"
```

Теперь мы создадим эту функцию. Заметьте, что символы + выводятся на экран программой и их не нужно набирать. Символ { информирует R о том, что будет введено несколько команд. Нажатие *Enter* после { заставит R вывести знак +. После этого мы можем ввести столько команд, сколько захотим, нажимая *Enter* после каждой из них. Наконец, символ } информирует R о том, что больше команды вводиться не будут.

```
> LoadLibraries = function() {
+ library(ISLR)
+ library(MASS)
+ print("The libraries have been loaded.")
+ }
```

Если мы наберем LoadLibraries теперь, то R покажет нам содержимое этой функции.

```
> LoadLibraries
function() {
    library(ISLR)
    library(MASS)
    print("The libraries have been loaded.")
}
```

Вызов этой функции приведет к подключению библиотек и выводу на экран соответствующего уведомления.

```
> LoadLibraries()
[1] "The libraries have been loaded."
```

3.7 Упражнения

Теоретические

1. Опишите нулевые гипотезы, соответствующие p–значениям в табл. 3.4. Объясните, какие выводы вы можете сделать на основе этих p–значений. Ваше объяснение должно быть сформулировано в отношении связи между объемом продаж и затратами на телерекламу, радиорекламу и рекламу в газетах, а не в отношении коэффициентов линейной модели.

2. Приведите подробное объяснение различий в применении метода KNN для задач классификации и регрессии.

3. Предположим, что у нас есть набор данных с пятью предикторами: X_1 — средний балл за школьные предметы (GPA[18]), X_2 — коэффициент IQ, X_3 — пол (1 для женщин и 0 для мужчин), X_4 — взаимодействие между GPA и IQ, и X_5 — взаимодействие между GPA и полом. Зависимой переменной является начальная заработная плата после окончания университета (тыс. долларов). Представьте, что мы используем метод наименьших квадратов для подгонки модели и получаем $\hat{\beta}_0 = 50$, $\hat{\beta}_1 = 20$, $\hat{\beta}_2 = 0.07$, $\hat{\beta}_3 = 35$, $\hat{\beta}_4 = 0.01$ и $\hat{\beta}_5 = -10$.

   (a) Какой из приведенных ниже ответов правильный и почему?

      i. При фиксированных значениях IQ и GPA мужчины в среднем зарабатывают больше женщин.

      ii. При фиксированных значениях IQ и GPA женщины в среднем зарабатывают больше мужчин.

      iii. При фиксированных значениях IQ и GPA мужчины в среднем зарабатывают больше женщин при условии, что GPA достаточно высок.

      iv. При фиксированных значениях IQ и GPA женщины в среднем зарабатывают больше мужчин при условии, что GPA достаточно высок.

   (b) Дайте прогнозную оценку заработной платы женщины с IQ = 111 и GPA = 4.0.

   (c) Верно ли следующее утверждение: поскольку коэффициент для взаимодействия между GPA и IQ очень небольшой, то у нас есть мало оснований сделать вывод о наличии эффекта взаимодействия. Объясните свой ответ.

4. Я получил данные ($n = 100$ наблюдений), содержащие один предиктор и одну независимую количественную переменную. Затем я подогнал линейную регрессионную модель к этим данным, а также кубическую регрессию вида $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$.

   (a) Представьте, что истинная зависимость между X и Y является линейной, т. е. $Y = \beta_0 + \beta_1 X + \epsilon$. Подумайте о сумме квадратов остатков (RSS) линейной регрессии на обучающей выборке,

[18] Эта аббревиатура происходит от «gross point average» — важного показателя, учитываемого при принятии абитуриентов в университеты США. — *Прим. пер.*

а также о RSS кубической регрессии на обучающей выборке. Стоит ли нам ожидать, что одна из этих величин будет ниже другой, обе величины будут равны, или имеющейся у нас информации недостаточно, чтобы сделать какое–либо заключение? Обоснуйте свой ответ.

(b) Ответьте на вопрос (a) в отношении RSS для контрольной, а не обучающей выборки.

(c) Представьте, что истинная зависимость между X и Y не является линейной, но мы не знаем, как сильно она отличается от линейной зависимости. Подумайте о RSS линейной регрессии на обучающей выборке и о RSS кубической регрессии на обучающей выборке. Стоит ли нам ожидать, что одна из этих величин будет ниже другой, обе величины будут равны, или имеющейся у нас информации недостаточно, чтобы сделать какое–либо заключение? Обоснуйте свой ответ.

(d) Ответьте на вопрос (c) в отношении RSS для контрольной, а не обучающей выборки.

5. Предположим, что у нас имеются предсказанные значения, полученные на основе линейной регрессионной модели без свободного члена. При таком сценарии i–е предсказанное значение принимает форму

$$\hat{y}_i = x_i\hat{\beta},$$

где

$$\hat{\beta} = \left(\sum_{i=1}^{n} x_i y_i\right) / \left(\sum_{i=1}^{n} x_{i'}^2\right). \tag{3.38}$$

Покажите, что верно следующее выражение:

$$\hat{y}_i = \sum_{i'=1}^{n} a_{i'} y_{i'}.$$

Что представляет собой $a_{i'}$?

Обратите внимание: мы интерпретируем этот результат, исходя из того, что предсказанные на основе линейной регрессии значения являются линейными комбинациями *значений отклика.*

6. Используя (3.4), докажите, что в случае простой линейной регрессии линия наименьших квадратов всегда проходит через точку $(\bar{x}, \bar{y})$.

7. В тексте главы утверждается, что в случае простой линейной регрессии Y по X статистика R^2 из (3.17) равна квадрату коэффициента корреляции между X и Y (3.18). Докажите, что это так и есть. Для облегчения задачи вы можете допустить, что $\bar{x} = \bar{y} = 0$.

Практические

8. Этот вопрос касается применения простой линейной регрессии к данным `Auto`.

   (a) Воспользуйтесь функцией `lm()` для подгонки простой линейной регрессии с `mpg` в качестве отклика и `horsepower` в качестве предиктора. Примените функцию `summary()` для вывода результатов на экран. Прокомментируйте эту выведенную информацию. Например:

      i. Есть ли линейная связь между предиктором и откликом?
      ii. Насколько сильна связь между предиктором и откликом?
      iii. Каково направление связи между предиктором и откликом?
      iv. Чему равно модельное значение `mpg` при `horsepower` $= 98$? Каковы соответствующие 95%–ные доверительные интервалы для регрессионной прямой и для предсказанного значения?

   (b) Постройте график зависимости отклика от предиктора. Используйте функцию `abline()` для добавления регрессионной линии, найденной по методу наименьших квадратов.

   (c) Воспользуйтесь функцией `plot()` для построения диагностических графиков для модели, подогнанной по методу наименьших квадратов. Прокомментируйте любые обнаруженные вами проблемы с этой моделью.

9. Этот вопрос касается применения множественной линейной регрессии к данным `Auto`.

   (a) Постройте матрицу диаграмм рассеяния, которая включает все переменные из этой таблицы данных.

   (b) Вычислите корреляционную матрицу для всех переменных, используя функцию `cor()`. Вам потребуется исключить из расчетов качественную переменную `name`. `cor()`

   (c) Воспользуйтесь функцией `lm()` для подгонки множественной линейной регрессии с `mpg` в качестве отклика и всех других переменных, за исключением `name`, в качестве предикторов. Примените функцию `summary()` для вывода результатов на экран. Прокомментируйте эти результаты. Например:

      i. Имеется ли связь между предикторами и откликом?
      ii. Какие предикторы проявляют статистически значимую связь с откликом?
      iii. О чем говорит коэффициент переменной `year`?

   (d) Воспользуйтесь функцией `plot()` для построения диагностических графиков для этой линейной регрессионной модели. Прокомментируйте любые обнаруженные вами проблемы с этой моделью. Указывают ли остатки на наличие каких–либо необычно больших выбросов? Указывает ли график значений показателя разбалансировки на наличие наблюдений с необычно высокой разбалансировкой?

(e) Используйте символы `*` и `:` для подгонки регрессионных моделей с эффектами взаимодействия. Являются ли какие–либо взаимодействия статистически значимыми?

(f) Попробуйте применить несколько разных преобразований к переменным, таких как $\log(X)$, $\sqrt{X}$, X^2. Прокомментируйте свои находки.

10. Для ответа на следующий вопрос необходимо воспользоваться набором данных `Carseats`.

(a) Постройте множественную регрессионную модель для предсказания `Sales` на основе `Price`, `Urban` и `US`[19].

(b) Приведите интерпретацию каждого коэффициента в модели. Будьте осторожны — некоторые переменные являются качественными!

(c) Запишите модель в виде уравнения (помня о необходимости должным образом обходиться с качественными переменными).

(d) Для каких из предикторов вы можете отклонить нулевую гипотезу $H_0 : \beta_j = 0$?

(e) Учитывая свой ответ на предыдущий вопрос, постройте модель только с теми предикторами, для которых было обнаружено свидетельство в пользу их связи с откликом.

(f) Насколько хорошо модели из (a) и (e) описывают данные?

(g) Используя модель из (e), рассчитайте 95%–ные доверительные интервалы для каждого коэффициента.

(h) Имеются ли указания на наличие выбросов или наблюдений с высокой разбалансировкой в модели из (e)?

11. В этой задаче мы исследуем t–критерий для нулевой гипотезы $H_0 : \beta = 0$ в отношении простой линейной регрессии без свободного члена. Для начала мы сгенерируем значения предиктора `x` и отклика `y`:

```
> set.seed(1)
> x = rnorm(100)
> y = 2 * x + rnorm(100)
```

(a) Постройте простую линейную регрессию `y` по `x` *без* свободного члена. Приведите оценку коэффициента $\hat{\beta}$, стандартную ошибку оценки этого коэффициента, а также t–критерий и p–значение, связанные с нулевой гипотезой $H_0 : \beta = 0$. Прокомментируйте эти результаты. (Вы можете построить регрессию без свободного члена, используя команду `lm(y ~ x + 0)`.)

[19] `Sales` — количество проданных единиц товара (тыс.); `Urban` — индикаторная переменная, кодирующая местоположение магазина — в городе (`Yes`) или за его пределами (`No`); `US` — индикатор того, находится ли магазин в США (`Yes`) или нет (`No`). — *Прим. пер.*

(b) Теперь постройте простую линейную регрессию `x` по `y` без свободного члена, и приведите оценку регрессионного коэффициента, ее стандартную ошибку, а также t–критерий и p–значение, связанные с нулевой гипотезой $H_0 : \beta = 0$. Прокомментируйте эти результаты.

(c) Какова связь между результатами, полученными в (a) и (b)?

(d) В регрессии Y по X без свободного члена t–критерий для $H_0 : \beta = 0$ принимает форму $\hat{\beta}/\mathrm{SE}(\hat{\beta})$, где определение оценки $\hat{\beta}$ дано в (3.38), а

$$\mathrm{SE}(\hat{\beta}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2}{(n-1)\sum_{i'=1}^{n} x_{i'}^2}}.$$

(Эти формулы несколько отличаются от тех, которые были приведены в подразделах 3.1.1 и 3.1.2, поскольку здесь мы строим регрессию без свободного члена.) Покажите, используя необходимые алгебраические преобразования, а также количественные вычисления в `R`, что t–критерий можно записать в виде

$$\frac{(\sqrt{n-1})\sum_{i=1}^{n} x_i y_i}{\sqrt{(\sum_{i=1}^{n} x_i^2)(\sum_{i'=1}^{n} y_{i'}^2) - (\sum_{i'=1}^{n} x_{i'} y_{i'})^2}}.$$

(e) Используя результаты из (d), докажите, что t–критерий в регрессии `y` по `x` не отличается от t–критерия в регрессии `x` по `y`.

(f) Выполнив вычисления в `R`, покажите, что при подгонке модели, *включающей* свободный член, t–критерий для $H_0 : \beta_1 = 0$ в регрессии `y` по `x` не отличается от такового в регрессии `x` по `y`.

12. Эта задача касается простой линейной регрессии без свободного члена.

(a) Вспомните, что определение оценки коэффициента $\hat{\beta}$ для линейной регрессии Y по X без свободного члена дано в (3.38). При каком условии оценка коэффициента для регрессии X по Y не отличается от оценки коэффициента для регрессии Y по X?

(b) Используя `R`, создайте набор данных с $n = 100$ наблюдениями, для которого оценка коэффициента регрессии X по Y *отличается* от оценки коэффициента регрессии Y по X.

(c) Используя `R`, создайте набор данных с $n = 100$ наблюдениями, для которого оценка коэффициента регрессии X по Y *не отличается* от оценки коэффициента регрессии Y по X.

13. В этом упражнении вы создадите несколько искусственных наборов данных и подгоните к ним простые линейные регрессионные модели. Убедитесь, что для получения одинаковых результатов вы применили команду `set.seed(1)` до выполнения задания (a).

(a) Используя функцию `rnorm()`, создайте вектор `x` со 100 наблюдениями, извлеченными из нормального распределения $N(0, 1)$. Он будет соответствовать некоторому признаку X.

(b) Используя функцию `rnorm()`, создайте вектор `eps` с 100 наблюдениями, извлеченными из распределения $N(0, 0.25)$, т. е. из нормального распределения со средним значением 0 и дисперсией 0.25.

(c) Используя `x` и `eps`, создайте вектор `y` в соответствии с моделью

$$Y = -1 + 0.5X + \epsilon. \tag{3.39}$$

Какова длина вектора `y`? Чему равны значения β_0 и β_1 в этой линейной модели?

(d) Постройте диаграмму рассеяния, изображающую зависимость между `x` и `y`. Прокомментируйте, что вы видите.

(e) Постройте линейную модель по методу наименьших квадратов для предсказания `y` по `x`. Прокомментируйте полученный результат. Насколько $\hat{\beta}_0$ и $\hat{\beta}_1$ сравнимы с β_0 и β_1?

(f) Добавьте линию наименьших квадратов к диаграмме рассеяния, полученной в (d). Изобразите истинную регрессионную линию на графике каким–либо другим цветом. Воспользуйтесь командой `legend()` для создания подходящей легенды.

(g) Теперь постройте полиномиальную регрессионную модель, которая предсказывает `y` по `x` и $\texttt{x}^2$. Похоже ли, что квадратичный член улучшает модель? Объясните свой ответ.

(h) Повторите (a) – (f), изменив процесс генерации данных таким образом, чтобы в них было *меньше* шума. Модель (3.39) должна оставаться неизменной. Вы можете сделать это путем уменьшения дисперсии нормального распределения, используемого для создания остатков ϵ в (b). Опишите свои результаты.

(i) Повторите (a) – (f), изменив процесс генерации данных таким образом, чтобы в них было *больше* шума. Модель (3.39) должна оставаться неизменной. Вы можете сделать это путем увеличения дисперсии нормального распределения, из которого извлекаются остатки ϵ в (b). Опишите свои результаты.

(j) Каковы доверительные интервалы для β_0 и β_1, полученные на основе исходного набора данных, набора данных с большим шумом и набора данных с меньшим шумом? Прокомментируйте свои результаты.

14. Эта задача посвящена проблеме *коллинеарности*.

(a) Выполните следующие команды в `R`:

```
> set.seed(1)
> x1 = runif(100)
> x2 = 0.5 * x1 + rnorm(100)/10
> y = 2 + 2 * x1 + 0.3 * x2 + rnorm(100)
```

В последней строке создается линейная модель, в которой `y` является функцией от `x1` и `x2`. Запишите эту линейную модель в виде формулы. Чему равны регрессионные коэффициенты?

(b) Какова корреляция между `x1` и `x2`? Постройте диаграмму рассеяния, изображающую связь между этими переменными.

(c) Используя эти данные, постройте регрессионную модель по методу наименьших квадратов для предсказания `y` по `x1` и `x2`. Опишите полученные результаты. Чему равны $\hat{\beta}_0$, $\hat{\beta}_1$ и $\hat{\beta}_2$? Как эти коэффициенты соотносятся с истинными значениями β_0, β_1 и β_2? Можете ли вы отклонить нулевую гипотезу $H_0 : \beta_1 = 0$? Как насчет нулевой гипотезы $H_0 : \beta_2 = 0$?

(d) Теперь подгоните регрессию по методу наименьших квадратов для предсказания `y` только по `x1`. Прокомментируйте свои результаты. Можете ли вы отклонить нулевую гипотезу $H_0 : \beta_1 = 0$?

(e) Теперь подгоните регрессию по методу наименьших квадратов для предсказания `y` только по `x2`. Прокомментируйте свои результаты. Можете ли вы отклонить нулевую гипотезу $H_0 : \beta_1 = 0$?

(f) Противоречат ли результаты, полученные в (c) – (e), друг другу? Объясните свой ответ.

(g) Теперь допустим, что мы получили дополнительное наблюдение, которое, к сожалению, ранее было измерено неверно.

```
> x1 = c(x1, 0.1)
> x2 = c(x2, 0.8)
> y = c(y, 6)
```

Повторите подгонку моделей из (c) и (e), используя эти новые данные. Какой эффект оказывает это новое наблюдение на каждую из моделей? Является ли это наблюдение выбросом в каждой модели? Наблюдением с высокой разбалансировкой? И тем, и другим? Объясните свои ответы.

15. Эта задача касается набора данных `Boston`, который мы уже видели в лабораторной работе к этой главе. Теперь мы попытаемся предсказать уровень преступности на душу населения, используя другие переменные из этого набора данных. Другими словами, уровень преступности на душу населения является откликом, а все остальные переменные — предикторами.

(a) Постройте простую регрессионную модель для предсказания отклика на основе каждого предиктора. Опишите свои результаты. В какой из моделей имеет место статистически значимая связь между предиктором и откликом? Постройте несколько графиков в подтверждение своих выводов.

(b) Постройте множественную регрессионную модель для предсказания отклика на основе всех предикторов. Опишите свои результаты. Для каких предикторов мы можем отклонить нулевую гипотезу $H_0 : \beta_j = 0$?

(c) Как соотносятся результаты, полученные вами в заданиях (a) и (b)? Постройте график, на котором по оси x показаны регрессионные коэффициенты из простых моделей, полученных в (a), а по

оси y — коэффициенты из множественной регрессии, полученной в (b). Иными словами, каждому предиктору соответствует отдельная точка на графике. Коэффициент каждого предиктора из простой регрессии отложен по оси x, а соответствующий коэффициент из множественной регрессии — по оси y.

(d) Есть ли указания на нелинейную связь между каким–либо из предикторов и откликом? Для ответа на этот вопрос подгоните для каждого предиктора модель следующего вида:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon.$$

Глава 4

Классификация

Линейная регрессионная модель, которая обсуждалась в главе 3, подразумевает, что зависимая переменная Y является количественной. Однако часто отклик может быть представлен *качественной* переменной. Например, цвет глаз — это качественная переменная, принимающая значения «голубой», «коричневый» или «зеленый». Иногда качественные переменные называют *категориальными*; мы будем использовать эти термины попеременно. В этой главе мы рассмотрим методы, предназначенные для предсказания качественных откликов — процесса, известного как *классификация*. Предсказание качественного отклика для некоторого наблюдения можно назвать классификацией, поскольку это наблюдение относят к определенной категории, или классу. С другой стороны, нередко методы классификации в качестве основы для выполнения классификации сначала предсказывают вероятность каждой из категорий качественной переменной. В этом смысле они ведут себя подобно методам регрессии.

качественная переменная

классификация

Существует множество методов классификации, или *классификаторов*, которые можно использовать для предсказания качественного отклика. Мы затронули некоторые из них в подразделах 2.1.5 и 2.2.3. В этой главе мы обсуждаем три наиболее широко распространенных классификатора: *логистическую регрессию*, *линейный дискриминантный анализ* и *метод K ближайших соседей*. В более поздних главах мы обсуждаем методы, требующие большего объема вычислений, такие как обобщенные аддитивные модели (глава 7), деревья классификации, случайные леса и бустинг (глава 8), а также машины опорных векторов (глава 9).

классификатор

логистическая регрессия

линейный дискриминантный анализ

метод K ближайших соседей

4.1 Общее представление о классификации

Проблемы классификации встречаются часто, возможно даже чаще, чем проблемы регрессии. Вот некоторые примеры:

1. Человек поступает в отделение экстренной медицинской помощи с набором симптомов, которые могут быть связаны с одним из трех заболеваний. Каким из трех заболеваний болен этот человек?

2. Сервис банковского обслуживания через Интернет должен иметь возможность определить, является ли та или иная онлайн–транзакция мошеннической, учитывая IP–адрес пользователя, историю его предыдущих транзакций и т. д.

3. На основе данных по структуре ДНК у ряда пациентов, имеющих определенное заболевание, и у пациентов, свободных от этого заболевания, биолог хотел бы выяснить, какие мутации ДНК вредны (т. е. вызывают заболевание), а какие — нет.

Подобно тому, как это было с регрессией, у нас имеется набор обучающих наблюдений $(x_1, y_1), \ldots, (x_n, y_n)$, которые мы можем использовать для построения классификатора. Мы хотим, чтобы наш классификатор хорошо работал не только на обучающих данных, но и на контрольных наблюдениях, которые не были использованы для обучения классификатора.

В этой главе мы проиллюстрируем концепцию классификации на примере имитированных данных `Default`. Мы заинтересованы в предсказании того, погасит ли клиент свой долг по кредитной карте, учитывая размер годового дохода этого клиента и месячного баланса на его кредитной карте. Данные представлены на рис. 4.1. Мы изобразили годовой доход `income` и месячный баланс на кредитной карте `balance` для группы из 10 000 человек. Слева на рис. 4.1 клиенты, не выплатившие долг за текущий месяц, показаны оранжевым цветом, а выплатившие — голубым. (Общая доля неплательщиков составляет около 3%, и поэтому мы привели на графике только часть клиентов, погасивших долг). Похоже, что неплательщики в целом имеют более высокие значения долга по кредитной карте. На рис. 4.1 справа приведены две пары диаграмм размахов. Первая из них показывает распределение значений `balance`, разбитых в соответствии с уровнями бинарной переменной `default` (неплатеж), а вторая диаграмма показывает похожую информацию для `income`. В этой главе мы узнаем, как построить модель для предсказания `default` (Y) для любых заданных значений `balance` (X_1) и `income` (X_2). Поскольку Y не является количественной переменной, то простая линейная регрессионная модель из главы 3 здесь не подходит.

Следует отметить, что на рис. 4.1 хорошо видна тесная связь между предиктором `balance` и откликом `default`. В большинстве практических задач связь между предиктором и откликом будет далеко не такой выраженной. Однако в целях иллюстрации процедур классификации, обсуждаемых в этой главе, мы воспользуемся примером, в котором теснота связи между предиктором и откликом несколько преувеличена.

4.2 Почему не линейная регрессия?

Выше мы заявили, что метод линейной регрессии не подходит для работы с качественным откликом. Почему нет?

Представьте, что мы пытаемся предсказать состояние пациентки из отделения неотложной медицинской помощи по совокупности наблюдаемых у этой пациентки симптомов. В этом упрощенном примере возможны три диагноза: инсульт, передозировка наркотиками и эпилептический припадок. Мы могли бы закодировать эти значения в виде количественной переменной следующим образом:

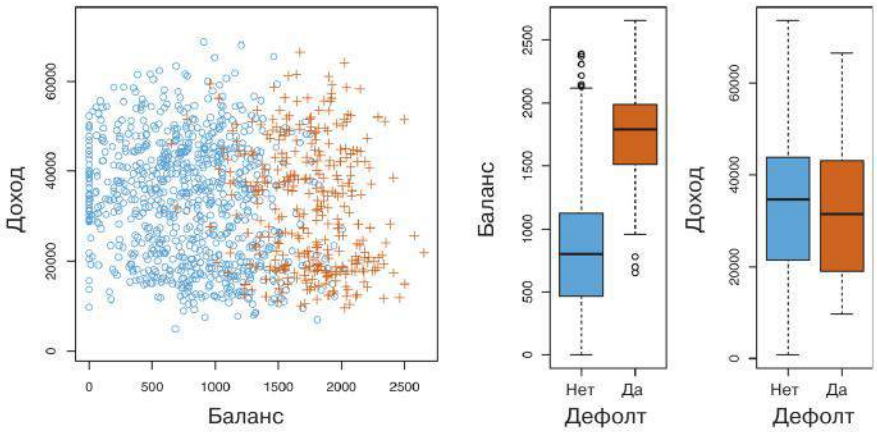

РИСУНОК 4.1. *Набор данных* `Default`. *Слева: значения годового дохода и уровня баланса на кредитной карте у нескольких клиентов банка. Клиенты, которые не погасили свой долг по кредитной карте, показаны оранжевым цветом, а те, которые погасили, — голубым.* В центре: *диаграмма размахов значений баланса на кредитной карте у клиентов с разным статусом дефолта.* Справа: *диаграмма размахов значений годового дохода у клиентов с разным статусом дефолта*

$$Y = \begin{cases} 1, & \text{если инсульт} \\ 2, & \text{если передозировка} \\ 3, & \text{если припадок.} \end{cases}$$

Применив такое кодирование, можно было бы воспользоваться методом наименьших квадратов и подогнать линейную регрессионную модель, предсказывающую Y на основе совокупности предикторов $X_1, \ldots, X_p$. К сожалению, такое кодирование подразумевает определенный порядок в расположении значений отклика: оно помещает передозировку наркотиками между инсультом и эпилептическим припадком и настаивает на том, что разница между инсультом и передозировкой наркотиками такая же, как и между передозировкой и эпилептическим припадком. На практике для этого нет никаких оснований. Например, одинаково приемлемым было бы следующее кодирование:

$$Y = \begin{cases} 1, & \text{если припадок} \\ 2, & \text{если инсульт} \\ 3, & \text{если передозировка,} \end{cases}$$

которое подразумевало бы совершенно другую связь между эти тремя медицинскими состояниями. Каждый из приведенных способов кодирования породил бы фундаментально разные линейные модели, которые в итоге привели бы к разным предсказаниям на контрольных данных.

Если бы значения зависимой переменной действительно можно было упорядочить естественным образом (например, *мягкий* $>$ *умеренный* $>$

суровый), и у нас были основания считать, что разница между «мягким» и «умеренным» похожа на разницу между «умеренным» и «суровым», то кодирование в виде 1, 2 и 3 было бы уместным. В целом, к сожалению, не существует естественного способа конвертировать качественную зависимую переменную с числом уровней больше двух в количественный отклик, пригодный для линейной регрессии.

бинарный

Для *бинарного* качественного отклика (т. е. переменной с двумя уровнями) ситуация лучше. Например, может оказаться так, что есть только два варианта состояния пациента: инсульт и передозировка наркотиками. В таком случае мы потенциально могли бы использовать метод индикаторных переменных (см. подраздел 3.3.1), чтобы закодировать отклик следующим образом:

$$Y = \begin{cases} 0, & \text{если инсульт} \\ 1, & \text{если передозировка.} \end{cases}$$

Далее мы могли бы построить линейную регрессию для этого бинарного отклика и предсказывать передозировку наркотиками при $\hat{Y} > 0.5$ и инсульт в остальных случаях. Для бинарного отклика несложно показать, что даже если мы поменяем приведенные выше коды состояний местами, то линейная регрессия в итоге приведет к таким же предсказаниям.

Для бинарного отклика, закодированного при помощи 1 и 0, как в вышеприведенном примере, регрессия по методу наименьших квадратов действительно имеет смысл: можно показать, что в этом специальном случае величина $X\hat{\beta}$, полученная при помощи линейной регрессии, есть не что иное, как оценка $\Pr(\text{передозировка наркотиками}|X)$. Однако при использовании линейной регрессии некоторые из наших оценок могут оказаться за пределами интервала $[0, 1]$ (см. рис. 4.2), что затруднит их интерпретацию в качестве вероятностей! Тем не менее такие предсказания можно упорядочить и рассматривать их как грубые оценки вероятности. Интересно, что результаты классификации, которые мы получаем для бинарного отклика при помощи линейной регрессии, будут идентичны результатам, которые дает линейный дискриминантный анализ (LDA) (раздел 4.4).

Однако метод индикаторных переменных нелегко приспособить для качественных откликов с числом уровней больше двух. По этим причинам предпочтительно использовать методы классификации (например, рассмотренные в этой книге), которые специально предназначены для качественных откликов.

4.3 Логистическая регрессия

Вернемся снова к набору данных `Default`, в котором зависимая переменная `default` представлена двумя категориями — `Yes` и `No`. Вместо того чтобы непосредственно моделировать отклик Y, логистическая регрессия моделирует *вероятность* принадлежности Y к той или иной категории.

Для набора данных `Default` логистическая регрессия моделирует вероятность неуплаты долга (дефолта). Например, вероятность дефолта для некоторого значения `balance` можно записать как

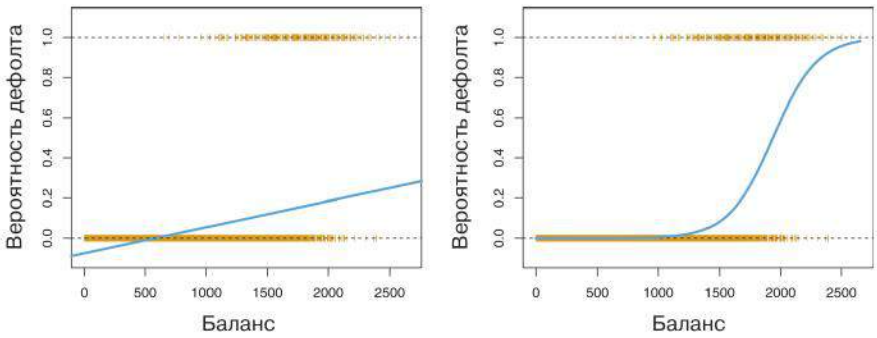

РИСУНОК 4.2. *Классификация с использованием набора данных* `Default`. Слева: *вероятности дефолта, оцененные при помощи линейной регрессии. Некоторые вероятности оказались отрицательными! Оранжевые вертикальные отрезки показывают значения 0 и 1, кодирующие категории дефолта* (`No` *и* `Yes`). Справа: *вероятности дефолта, оцененные при помощи логистической регрессии. Все вероятности лежат в пределах интервала от 0 до 1*

$$\Pr(\texttt{default} = \texttt{Yes}|\texttt{balance}).$$

Значения $\Pr(\texttt{default} = \texttt{Yes}|\texttt{balance})$, которые мы сокращенно обозначаем как $p(\texttt{balance})$, будут изменяться от 0 до 1. Далее для заданного значения `balance` можно предсказать вероятный статус дефолта. Например, для клиента с $p(\texttt{balance} > 0.5)$ можно было бы предсказать $\texttt{default} = \texttt{Yes}$. Если компания желает быть консервативной в предсказании того, какие клиенты не вернут свой долг, то в качестве альтернативы она может использовать более низкое пороговое значение вероятности, например $p(\texttt{balance} > 0.1)$.

4.3.1 Логистическая модель

Как же нам смоделировать связь между $p(X) = \Pr(Y = 1|X)$ и X? (Для удобства мы применяем универсальное кодирование отклика в виде 0/1). В разделе 4.2 мы говорили о том, что для представления этих вероятностей можно воспользоваться линейной регрессионной моделью:

$$p(X) = \beta_0 + \beta_1 X. \tag{4.1}$$

Если мы применим этот подход для предсказания $\texttt{default} = \texttt{Yes}$ на основе `balance`, то получим модель, показанную слева на рис. 4.2. Хорошо видна проблема с этим подходом: для значений баланса на кредитной карте, близких к нулю, мы предсказываем отрицательную вероятность дефолта; если бы мы хотели выполнить предсказания для очень больших величин баланса, то получили бы значения, превышающие 1. Эти предсказания не имеют смысла, поскольку истинная вероятность дефолта, вне зависимости от баланса на кредитной карте, должна изменяться от 0 до 1. Эта проблема не является уникальной для данных по кредитным неплатежам.

Каждый раз, когда к бинарному отклику, закодированному при помощи 0 и 1, подгоняется прямая линия, мы практически всегда можем получить $p(X) < 0$ для одних значений X и $p(X) > 1$ — для других (если только размах значений X не ограничен).

Во избежание этой проблемы мы должны моделировать $p(X)$ на основе функции, которая возвращает значения из интервала от 0 до 1 для всех X. Таким свойством обладают многие функции. В логистической регрессии мы применяем логистическую функцию:

логистическая функция

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}. \tag{4.2}$$

максимальное правдоподобие

Для подгонки модели (4.2) мы применяем так называемый *метод максимального правдоподобия*, который обсуждается в следующем разделе. На рис. 4.2 справа показана логистическая регрессионная модель, подогнанная к данным из таблицы `Default`. Заметьте, что теперь для низких значений кредитного баланса мы предсказываем близкую к нулю, но всегда положительную вероятность дефолта. Аналогично для высоких значений баланса мы предсказываем вероятность дефолта, близкую к единице, но никогда не превышающую ее. Логистическая функция всегда порождает такую *S–образную* кривую, и поэтому вне зависимости от значений X мы будем получать разумный прогноз. Мы видим также, что логистическая модель отражает размах вероятностей лучше, чем линейная регрессионная модель, представленная на графике слева. Средняя расчетная вероятность в обоих случаях составляет 0.0333 (в среднем по обучающим данным) и равна общей доли неплательщиков в этом наборе данных.

После небольших преобразований (4.2) мы обнаружим, что

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X}. \tag{4.3}$$

Величина $p(X)/\left[1 - p(X)\right]$ называется *риск события*[1] и может принимать любое значение от 0 до ∞. Значения риска, близкие к 0 и ∞, указывают на очень низкие и очень высокие вероятности дефолта соответственно. Например, в среднем 1 из 5 человек с риском $1/4$ станет неплательщиком, поскольку $p(X) = 0.2$ предполагает отношение $\frac{0.2}{1-0.2} = 1/4$. Аналогично неплательщиками в среднем станут девять из десяти человек с риском, равным 9, поскольку $p(X) = 0.9$ предполагает отношение $\frac{0.9}{1-0.9} = 9$. Концепция «риска» традиционно использовалась вместо вероятностей в конных скачках, т. к. она более естественно подходит для описания стратегии заключения пари.

Если мы прологарифмируем обе части (4.3), то получим

$$\log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X. \tag{4.4}$$

логит

Левая часть в этом уравнении называется *логарифмом риска*, или *логитом*. Как видим, логит регрессионной модели (4.2) является линейным по X.

[1] В оригинале используется термин «odds». Иногда этот термин переводится также как «шанс». — *Прим. пер.*

Вспомните из главы 3, что в линейной регрессионной модели β_1 отражает среднее изменение Y, вызванное увеличением X на 1. В то же время в логистической регрессионной модели увеличение X на одну единицу изменяет на β_1 логарифм риска (см. уравнение 4.4), или, что то же самое, умножает риск на e^{β_1} (см. уравнение 4.3). Однако в связи с тем, что связь между $p(X)$ и X в (4.2) не выглядит как прямая линия, β_1 *не отражает* изменение $p(X)$, вызванное увеличением X на 1. Величина, на которую $p(X)$ изменяется при изменении X на одну единицу, будет зависеть от текущего значения X. Однако вне зависимости от значения X при положительном β_1 рост X будет сопровождаться увеличением $p(X)$, а при отрицательном β_1 рост X вызовет снижение $p(X)$. Факт нелинейной связи между $p(X)$ и X, а также тот факт, что скорость изменения $p(X)$ при изменении X на одну единицу зависит от текущего значения X, можно увидеть на графике, представленном справа на рис. 4.2.

4.3.2 Оценивание регрессионных коэффициентов

Коэффициенты β_0 и β_1 в (4.2) неизвестны и подлежат оцениванию по доступным обучающим данным. В главе 3 для нахождения неизвестных коэффициентов линейной регрессии мы использовали метод наименьших квадратов. Хотя мы могли бы применить (нелинейный) метод наименьших квадратов для подгонки модели (4.4), предпочтительным является более общий метод максимального правдоподобия. Суть использования максимального правдоподобия для подгонки логистической регрессии заключается в следующем: используя (4.2), мы ищем такие оценки β_0 и β_1, чтобы предсказанная вероятность дефолта $\hat{p}(x_i)$ для каждого клиента максимально близко соответствовала его наблюдаемому статусу. Другими словами, мы пытаемся найти такие оценки $\hat{\beta}_0$ и $\hat{\beta}_1$, которые при подстановке в модель (4.2) для $p(X)$ давали бы число, близкое к единице для всех неплательщиков, и близкое к нулю для всех клиентов с высокой кредитоспособностью. Это простое объяснение можно формализовать в виде математического уравнения, известного как *функция правдоподобия*:

функция правдоподобия

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})). \tag{4.5}$$

Оценки $\hat{\beta}_0$ и $\hat{\beta}_1$ выбирают таким образом, чтобы *максимизировать* функцию правдоподобия.

Метод максимального правдоподобия является очень общим и используется для подгонки многих нелинейных моделей, рассматриваемых в этой книге. Оказывается, что в линейной регрессии метод наименьших квадратов является частным случаем метода максимального правдоподобия. Математические детали максимального правдоподобия находятся за рамками данной книги. Однако в целом логистическую регрессию и другие модели можно легко подогнать при помощи такой статистической программы, как `R`, и поэтому нам нет необходимости углубляться в детали процедуры нахождения максимального правдоподобия.

В табл. 4.1 показаны оценки коэффициентов и сопутствующая им информация, полученные в результате подгонки логистической регрессионной модели по данным `Default` с целью предсказания вероятности `default`

ТАБЛИЦА 4.1. *Коэффициенты логистической регрессионной модели для данных* `Default`*, которая предсказывает вероятность дефолта на основе размера долга по кредитной карте. Увеличение долга на одну единицу связано с увеличением логарифма риска дефолта на 0.0055 единицы*

	Коэффициент	Ст. ошибка	Z–критерий	p
Свободный член	−10.6513	0.3612	−29.5	< 0.0001
`balance`	0.0055	0.0002	24.9	< 0.0001

`= Yes` по `balance`. Мы видим, что $\hat{\beta}_1 = 0.0055$, т. е. увеличение `balance` связано с ростом вероятности дефолта. Если быть точнее, то увеличение баланса на кредитной карте на одну единицу приводит к увеличению логарифма риска дефолта на 0.0055 единицы.

Многие аспекты результатов расчета логистической регрессии, показанных в табл. 4.1, похожи на результаты для линейной регрессии из главы 3. Например, мы можем выразить точность оценок коэффициентов путем вычисления их стандартных ошибок. z–критерий в табл. 4.1 играет ту же роль, что и t–критерий в результатах для линейной регрессии (см., например, табл. 3.1 на стр. 80). Так, z–критерий коэффициента β_1 равен $\hat{\beta}_1/\mathrm{SE}(\hat{\beta}_1)$, и поэтому высокое (абсолютное) значение этого z–критерия является свидетельством против нулевой гипотезы $H_0 : \beta_1 = 0$. Эта нулевая гипотеза предполагает, что $p(X) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$, т. е. вероятность дефолта не зависит от баланса на кредитной карте. Поскольку приведенное в табл. 4.1 p–значение для `balance` является очень небольшим, мы можем отклонить H_0. Другими словами, мы заключаем, что между `balance` и `default` действительно имеется связь. Приведенная в табл. 4.1 оценка свободного члена уравнения обычно не представляет интереса; ее основное назначение заключается в коррекции предсказанных средних значений вероятности с учетом доли неплательщиков в данных.

4.3.3 Предсказания

Имея оценки коэффициентов, вычислить вероятность дефолта для любого значения баланса на кредитной карте очень просто. Например, используя оценки коэффициентов из табл. 4.1, мы предсказываем, что вероятность дефолта для клиента с долгом в 1000\$ составляет

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1000}}{1+e^{-10.6513+0.0055\times 1000}} = 0.00576,$$

что меньше 1%. В то же время предсказанная вероятность дефолта для клиента с долгом в 2000\$ намного выше и составляет 0.586, или 58.6%.

В логистическую регрессию можно включать качественные предикторы, используя метод индикаторных переменных (см. подраздел 3.3.1). Например, набор данных `Default` содержит качественную переменную `student`. Для построения модели мы просто создаем индикаторную переменную, которая принимает значение 1 для студентов и 0 для не студентов. Логистическая модель для предсказания вероятности дефолта на основе

ТАБЛИЦА 4.2. *Коэффициенты логистической регрессионной модели для данных* `Default`*, которая предсказывает вероятность дефолта на основе переменной* `student`*. Этот предиктор закодирован в виде индикаторной переменной (показана в таблице как* `student[Yes]`*) со значением 1 для студентов и значением 0 для не студентов*

	Коэффициент	Ст. ошибка	Z–критерий	p
Свободный член	−3.5041	0.0707	−49.55	< 0.0001
`student[Yes]`	0.4049	0.1150	3.52	0.0004

переменной `student` описана в табл. 4.2. Коэффициент индикаторной переменной положителен, а соответствующее p–значение статистически значимо. Это указывает на то, что студенты в целом имеют более высокие риски неуплаты долга, чем не студенты:

$$\widehat{\Pr}(\texttt{default = Yes} \mid \texttt{student = Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431,$$

$$\widehat{\Pr}(\texttt{default = Yes} \mid \texttt{student = No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292.$$

4.3.4 Множественная логистическая модель

Теперь мы рассмотрим проблему предсказания бинарного отклика на основе нескольких предикторов. Подобно тому, как простая линейная регрессия в главе 3 была расширена до множественной регрессионной модели, мы можем обобщить (4.4) следующим образом:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \tag{4.6}$$

где $X = (X_1, \ldots, X_p)$ — это набор из p предикторов. Уравнение (4.6) можно переписать как

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}. \tag{4.7}$$

Как и в подразделе 4.3.2, для нахождения оценок $\beta_0, \beta_1, \ldots, \beta_p$ мы применяем метод максимального правдоподобия.

В табл. 4.3 показаны оценки коэффициентов для логистической регрессионной модели, которая предсказывет дефолт по переменным `balance`, `income` (тыс. долларов) и `student`. Какого–либо сюрприза в полученном результате нет. P–значения у переменных `balance` и `student` очень низкие, что указывает на их тесную связь с вероятностью дефолта. Однако коэффициент у индикаторной переменной `student` отрицательный, указывая на то, что студенты имеют меньшую вероятность не выплатить долг в сравнении с не студентами. В то же время коэффициент у этой

ТАБЛИЦА 4.3. *Коэффициенты логистической регрессионной модели для данных* `Default`*, которая предсказывает вероятность дефолта на основе переменных* `balance`, `income` *и* `student`*. Предиктор* `student` *закодирован в виде индикаторной переменной* `student[Yes]` *со значением 1 для студентов и значением 0 для не студентов. При подгонке этой модели переменная* `income` *выражалась в тысячах долларов*

	Коэффициент	Ст. ошибка	Z–критерий	p
Свободный член	−10.8690	0.4923	−22.08	< 0.0001
`balance`	0.0057	0.0002	24.74	< 0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	−0.6468	0.2362	−2.74	0.0062

индикаторной переменной в табл. 4.2 является положительным. Но как это возможно, чтобы эта переменная была связана с *увеличением* вероятности дефолта в табл. 4.2 и со *снижением* вероятности дефолта в табл. 4.3? График, приведенный слева на рис. 4.3, иллюстрирует этот кажущийся парадокс. Оранжевая и голубая сплошные линии показывают средние уровни встречаемости неплательщиков среди студентов и не студентов соответственно в зависимости от долга по кредитной карте. Отрицательный коэффициент у `student` во множественной логистической регрессии показывает, что *для некоторых фиксированных значений* `balance` *и* `income` студенты имеют меньшую вероятность стать неплательщиками по сравнению с не студентами. И действительно — слева на рис. 4.3 мы видим, что встречаемость дефолта у студентов равна или ниже таковой у не студентов, вне зависимости от величины `balance`. Однако горизонтальные прерывистые линии в нижней части графика, которые показывают встречаемость неплательщиков среди студентов и не студентов в среднем по всем значениям `balance` и `income`, предполагают обратный эффект: в целом частота встречаемости неплательщиков у студентов выше, чем у не студентов. Как результат коэффициент для `student` в простой логистической регрессии имеет положительный знак (табл. 4.2).

График, приведенный на рис. 4.3 справа, объясняет это расхождение. Переменные `student` и `balance` коррелируют. Студенты в целом имеют более высокие долги, что, в свою очередь, связано с большей вероятностью дефолта. Другими словами, студенты чаще имеют высокие долги по кредитной карте, которые, как мы знаем по графику, показанному на рис. 4.3 слева, обычно сопровождаются большей частотой встречаемости дефолта. Поэтому даже если некоторый студент с определенным балансом на кредитной карте окажется неплательщиком с меньшей вероятностью, чем не студент с таким же балансом на кредитной карте, в целом студенты имеют более высокие долги, и поэтому становятся неплательщиками чаще, чем не студенты. Это важное различие для компании–эмитента кредитных карт, которая пытается определить, кому следует предложить свои услуги. Студент является более рискованным клиентом, чем не студент, при условии, что информация о долге по кредитной карте для этого студента недоступна. Однако студент является менее рискованным клиентом, чем не студент *с таким же балансом на кредитной карте*!

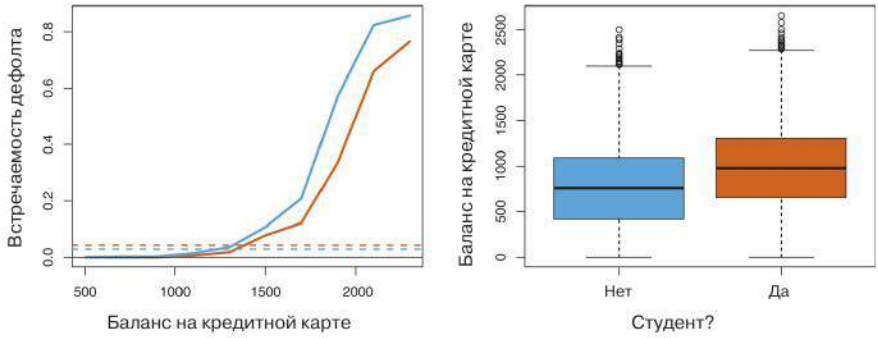

РИСУНОК 4.3. *Смешивание эффектов предикторов в данных* `Default`. *Слева: показана встречаемость дефолта среди студентов (оранжевый цвет) и не студентов (голубой цвет). Сплошные линии соответствуют встречаемости дефолта в зависимости от* `balance`, *тогда как горизонтальные прерывистые линии показывают его общую встречаемость. Справа: диаграмма размахов для* `Default` *у студентов (оранжевый цвет) и не студентов (голубой цвет)*

Этот простой пример иллюстрирует опасности и тонкости, связанные с построением регрессионных моделей с единственным предиктором в ситуациях, когда важными могут быть и другие предикторы. Как и в случае линейной регрессии, результаты, получаемые на основе одного предиктора, могут довольно сильно отличаться от результатов, полученных с использованием нескольких предикторов, особенно когда имеет место корреляция между предикторами. Обычно феномен, представленный на рис. 4.3, называют «*смешиванием эффектов*»[2].

смешивание эффектов

Подставляя оценки регрессионных коэффициентов из табл. 4.3 в (4.7), мы можем делать предсказания. Например, оценка вероятности дефолта у студента с долгом по кредитной карте в 1500\$ и доходом в 40 000\$ составляет

$$\hat{p}(X) = \frac{e^{-10.869+0.00574\times1500+0.003\times40-0.6468\times1}}{1+e^{-10.869+0.00574\times1500+0.003\times40-0.6468\times1}} = 0.058. \tag{4.8}$$

Оценка вероятности дефолта у не студента с такими же долгом и доходом составляет

$$\hat{p}(X) = \frac{e^{-10.869+0.00574\times1500+0.003\times40-0.6468\times0}}{1+e^{-10.869+0.00574\times1500+0.003\times40-0.6468\times0}} = 0.105. \tag{4.9}$$

(Здесь мы умножаем оценку коэффициента для `income` на 40, а не 40 000, поскольку в описанной в табл. 4.3 модели эта переменная выражается в тысячах долларов.)

[2] В оригинале используется термин «confounding». — *Прим. пер.*

4.3.5 Логистическая регрессия для зависимых переменных с числом классов > 2

Иногда мы хотим выполнить классификацию для зависимой переменной, у которой имеется больше двух классов. Например, в разделе 4.2 в примере с отделением экстренной медицинской помощи у нас было три типа состояния пациентов: инсульт, передозировка наркотиками и эпилептический припадок. При таком сценарии мы хотим смоделировать как $\Pr(Y = \text{инсульт}|X)$, так и $\Pr(Y = \text{передозировка}|X)$, при том что $\Pr(Y = \text{эпилепсия}|X) = 1 - \Pr(Y = \text{инсульт}|X) - \Pr(Y = \text{передозировка}|X)$. Логистические модели с двумя классами, описанные в предыдущих разделах, имеют расширенные формы для случаев с несколькими классами, однако на практике они применяются довольно редко. Одна из причин заключается в том, что метод, который мы обсуждаем в следующем разделе (дискриминантный анализ), очень популярен для решения проблем классификации с несколькими классами. Поэтому здесь мы не углубляемся в детали логистической регрессии для нескольких классов — отметим просто, что такой подход возможен и программное обеспечение для него доступно в R.

4.4 Дискриминантный анализ

В логистической регрессии $\Pr(Y = k|X = x)$ моделируется непосредственно на основе логистической функции, представленной в (4.7) для случая с двумя классами. Пользуясь статистическим жаргоном, можно сказать, что мы моделируем условное распределение отклика Y с учетом предиктора(–ов) X. Теперь мы рассмотрим альтернативный и прямой подход для оценивания этих вероятностей. В этом альтернативном подходе мы моделируем распределение предикторов X отдельно для каждого класса зависимой переменной (т. е. для конкретного уровня Y), а затем применяем теорему Баейса в обратном порядке для получения оценок $\Pr(Y = k|X = x)$. Если допустить, что эти распределения являются нормальными, то окажется, что эта модель очень похожа по форме на логистическую регрессию.

Зачем нам еще один метод, если у нас уже есть логистическая регрессия? На это есть несколько причин:

- Когда классы хорошо разделены, оценки параметров логистической регрессии на удивление очень нестабильны. Линейный дискриминантный анализ не страдает от этой проблемы.

- При малом n и примерно нормально распределенных предикторах во всех классах линейный дискриминантный анализ, опять–таки, более устойчив, чем логистическая регрессионная модель.

- Как было отмечено в подразделе 4.3.5, линейный дискриминантный анализ популярен в ситуациях, когда зависимая переменная имеет больше двух классов.

4.4.1 Использование теоремы Байеса для классификации

Предположим, что мы хотим отнести некоторое наблюдение к одному из K классов, где $K \geq 2$. Иными словами, качественная зависимая переменная Y может принимать K четко различающихся и неупорядоченных значений. Пусть π_k обозначает общую *априорную* вероятность того, что случайно выбранное наблюдение происходит из k–го класса, т. е. это вероятность того, что некоторое наблюдение ассоциировано с k–й категорией отклика Y. Пусть $f_k(X) \equiv Pr(X = x|Y = k)$ обозначает *функцию плотности вероятности* X для некоторого наблюдения, происходящего из k–го класса. Другими словами, $f_k(x)$ принимает относительно высокое значение, когда имеет место высокая вероятность пронаблюдать $X \approx x$ у некоторого наблюдения из класса k; в свою очередь, $f_k(x)$ принимает низкое значение при низкой вероятности того, что у некоторого наблюдения из k–го класса $X \approx x$. Тогда *теорема Байеса* утверждает, что

априорная вероятность

функция плотности вероятности

теорема Байеса

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}. \tag{4.10}$$

В соответствии в введенными ранее обозначениями мы будем использовать сокращение $p_k(X) = \Pr(Y = k|X)$. Это предполагает, что вместо непосредственного вычисления величины $p_k(X)$, подобно тому, как это было сделано в подразделе 4.3.1, мы можем просто подставить оценки π_k и $f_k(X)$ в (4.10). Как правило, получение оценки π_k не составляет труда, если у нас есть случайная выборка значений Y из генеральной совокупности: мы просто вычисляем долю обучающих наблюдений, принадлежащих к классу k. Однако нахождение $f_k(X)$ обычно является более сложной задачей, если только мы не допустим, что эти функции плотности имеют определенные простые формы. Мы называем $p_k(x)$ *апостериорной* вероятностью принадлежности некоторого наблюдения $X = x$ к классу k. Другими словами, это вероятность принадлежности данного наблюдения к классу k, *исходя из значения предиктора* у этого наблюдения.

апостериорная вероятность

Как мы знаем из главы 2, среди всех классификаторов баейсовский классификатор, который относит то или иное наблюдение к классу с наибольшей вероятностью $p_k(X)$, обладает минимальной возможной частотой ошибок. (Конечно, это справедливо только при условии, что параметры уравнения 4.10 заданы правильно.) Следовательно, если мы сумеем найти способ оценить $f_k(X)$, то сможем разработать классификатор, который аппроксимирует байесовский классификатор. Этому способу посвящены следующие разделы.

4.4.2 Линейный дискриминантный анализ для $p = 1$

Представим пока, что $p = 1$, т. е. у нас есть только один предиктор. Мы хотели бы получить оценку $f_k(x)$, которую можно подставить в (4.10) для нахождения $p_k(x)$. Далее мы будем относить то или иное наблюдение к классу, для которого вероятность $p_k(x)$ максимальна. Перед тем как оценить функцию $f_k(x)$, мы сделаем некоторые допущения в отношении ее формы.

нормальное, гауссово распределение

Пусть $f_k(x)$ представлена *нормальным*, или *гауссовым*, распределением. В одномерном случае нормальная функция плотности принимает форму

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right), \tag{4.11}$$

где μ_k и σ_k^2 — это среднее значение и дисперсия для k–го класса. Допустим пока, что $\sigma_1^2 = \cdots = \sigma_K^2$, т. е. все классы имеют общую дисперсию, которую мы обозначим как σ^2. Подставив (4.11) в (4.10), мы выясним, что

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}. \tag{4.12}$$

(Заметьте, что π_k в (4.12) обозначает априорную вероятность принадлежности некоторого наблюдения к классу k — ее не следует путать с математической константой $\pi \approx 3.14159$.) Байесовский классификатор относит то или иное наблюдение $X = x$ к классу, у которого вероятность (4.12) максимальна. Прологарифмировав (4.12) и выполнив некоторые перестановки, нетрудно будет показать, что это аналогично отнесению данного наблюдения к классу, у которого максимальна величина

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k). \tag{4.13}$$

Например, если $K = 2$ и $\pi_1 = \pi_2$, то байесовский классификатор относит наблюдение к классу 1 при $2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$ и к классу 2 в противном случае. Здесь байесовская решающая граница соответствует точке, где

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 - \mu_2}{2}. \tag{4.14}$$

Слева на рис. 4.4 приведен пример. Показаны две нормальные функции плотности — $f_1(x)$ и $f_2(x)$, соответствующие двум отдельным классам. Средние значения и дисперсии этих двух функций составляют $\mu_1 = -1.25$, $\mu_2 = 1.25$ и $\sigma_1^2 = \sigma_2^2 = 1$. Функции перекрываются, и поэтому для заданного $X = x$ имеет место некоторая неопределенность относительно класса, к которому принадлежит это наблюдение. Если предположить, что некоторое наблюдение может с одинаковой вероятностью принадлежать к любому классу, т. е. $\pi_1 = \pi_2 = 0.5$, то из (4.14) мы увидим, что байесовский классификатор относит наблюдение к классу 1 при $x < 0$ и к классу 2 в противном случае. Заметьте, что в этом примере мы имеем возможность вычислить байесовский классификатор благодаря нашей осведомленности о гауссовом распределении X в пределах каждого класса, а также о всех соответствующих параметрах распределения. В реальной ситуации вычислить байесовский классификатор мы не можем.

линейный дискриминантный анализ

На практике, даже если мы вполне уверены в нашем предположении о гауссовом распределении X в пределах каждого класса, нам все еще необходимо оценить параметры $\mu_1, \ldots, \mu_K$, $\pi_1, \ldots, \pi_K$ и σ^2. Метод *линейного дискриминантного анализа* (LDA)[3] аппроксимирует байесовский класси-

[3] Аббревиатура от «linear disriminant analysis». — *Прим. пер.*

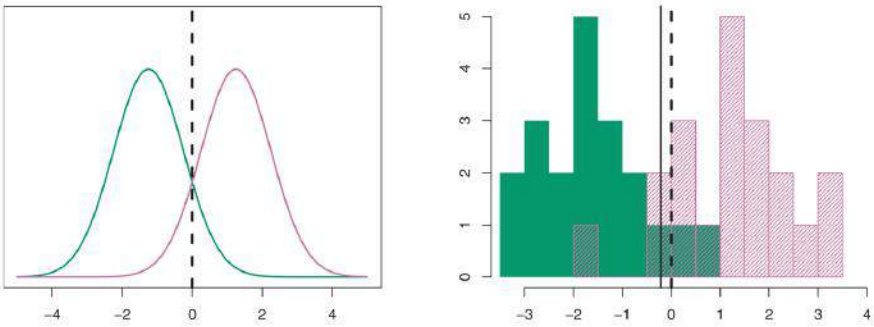

РИСУНОК 4.4. Слева: *показаны две одномерные нормальные функции плотности вероятностей. Вертикальная прерывистая линия соответствует байесовской решающей границе.* Справа: *гистограмма 20 наблюдений, случайным образом извлеченных из каждого класса. Байесовская решающая граница снова показана в виде вертикальной прерывистой линии. Сплошная вертикальная линии соответствует решающей границе LDA, оцененной по обучающим данным*

фикатор, подставляя оценки π_k, μ_k и σ^2 в (4.13). В частности, используются следующие оценки:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2, \tag{4.15}$$

где n — это общее число обучающих наблюдений, а n_k — число обучающих наблюдений в k–м классе. Оценка μ_k представляет собой просто среднее значение по всем обучающим наблюдениям из класса k, тогда как $\hat{\sigma}^2$ можно рассматривать как взвешенное среднее из выборочных дисперсий каждого из K классов. Иногда нам известны вероятности принадлежности к каждому классу $\pi_1, \ldots, \pi_K$, которые можно сразу использовать в вычислениях. В отсутствие какой–либо дополнительной информации LDA оценивает π_k по доле обучающих наблюдений, принадлежащих к классу k. Другими словами,

$$\hat{\pi}_k = n_k/n. \tag{4.16}$$

LDA–классификатор подставляет оценки из (4.15) и (4.16) в (4.13) и относит наблюдение $X = x$ к классу, у которого величина

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k) \tag{4.17}$$

максимальна. Прилагательное «*линейный*» в названии классификатора обусловлено тем фактом, что *дискриминантная функция* $\hat{\delta}_k(x)$ в (4.17)

дискриминантная функция

является линейной функцией от x (в противоположность какой–либо более сложной функции от x).

На рис. 4.4 справа показана гистограмма для 20 наблюдений, случайно отобранных из каждого класса. Для выполнения LDA мы сначала нашли оценки π_k, μ_k и σ^2, используя (4.15) и (4.16). Далее мы вычислили решающую границу (показана в виде черной сплошной линии) путем отнесения каждого наблюдения к классу с наибольшей величиной (4.17). Все точки слева от этой линии будут отнесены к «зеленому классу», а точки справа от этой линии — к «фиолетовому классу». Поскольку в данном случае $n_1 = n_2 = 20$, мы имеем $\hat{\pi}_1 = \hat{\pi}_2$. Как следствие решающая граница соответствует середине между выборочными средними двух классов: $(\hat{\mu}_1 + \hat{\mu}_2)/2$. На рисунке видно, что решающая граница LDA проходит несколько левее оптимальной байесовской решающей границы, равной $(\mu_1 + \mu_2)/2 = 0$. Каково качество этого LDA–классификатора в отношении рассматриваемых данных? Поскольку это имитированные данные, мы можем сгенерировать большой набор контрольных наблюдений для расчета байесовской частоты ошибок и частоты ошибок LDA на контрольной выборке. Эти ошибки составляют 10.6% и 11.1% соответственно. Другими словами, частота ошибок LDA–классификатора только на 0.5% выше минимально возможной ошибки! Таким образом, LDA очень хорошо работает на этом наборе данных.

Повторим еще раз: LDA–классификатор получают, делая допущение о том, что наблюдения в каждом классе имеют нормальное распределение со специфичным для конкретного класса вектором математического ожидания и общей дисперсией σ^2, и подставляя оценки этих параметров в байесовский классификатор. В подразделе 4.4.4 мы рассмотрим набор менее строгих допущений, которые позволяют наблюдениям из класса k иметь свою собственную дисперсию σ_k^2.

4.4.3 Линейный дискриминантный анализ для $p > 1$

Теперь мы расширим LDA–классификатор на случай с несколькими предикторами. Для этого мы предположим, что вектор $X = (X_1, X_2, \ldots, X_p)$ извлечен из *многомерного гауссова*, или многомерного нормального, распределения со специфичным для каждого класса вектором математического ожидания и общей ковариационной матрицей. Начнем с краткого описания такого распределения.

многомерное гауссово распределение

Многомерное гауссово распределение подразумевает, что каждый из предикторов имеет свое одномерное нормальное распределение (как в (4.11)) и в определенной степени коррелирует с другими предикторами. Два примера многомерных гауссовых распределений с $p = 2$ приведены на рис. 4.5. Высота поверхности в любой заданной точке соответствует вероятности того, что X_1 и X_2 находятся в пределах некоторой малой области около этой точки. Если на любом из приведенных графиков рассечь поверхность вдоль оси X_1 или оси X_2, то полученный поперечный разрез будет иметь форму одномерного нормального распределения. Слева на рис. 4.5 приведен пример, в котором $\mathrm{Var}(X_1) = \mathrm{Var}(X_2)$, а $\mathrm{Cor}(X_1, X_2) = 0$; эта поверхность имеет характерную *колоколообразную форму*. Однако, как показано на том же рисунке справа, эта колоколообразная форма будет искажена, если предикторы коррелируют друг с другом или имеют неоди-

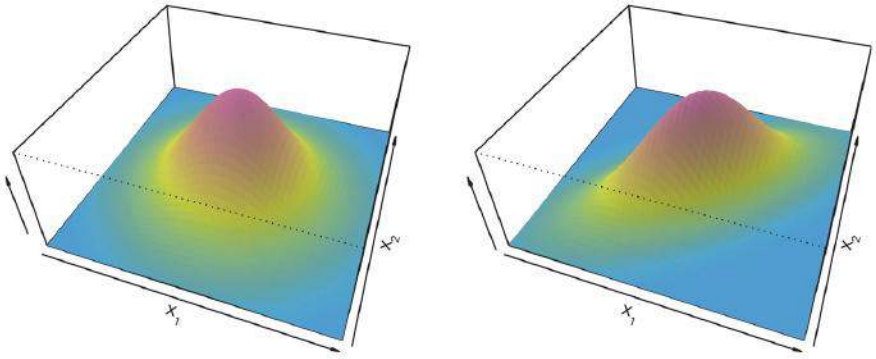

РИСУНОК 4.5. *Показаны две многомерные гауссовы функции плотности с $p = 2$. Слева: два предиктора не коррелируют друг с другом. Справа: корреляция между предикторами составляет 0.7*

наковые дисперсии. В такой ситуации основа колокола будет иметь форму эллипса, а не круга. Чтобы показать, что p–мерная случайная переменная X имеет многомерное гауссово распределение, мы используем нотацию $X \sim N(\mu, \mathbf{\Sigma})$. Здесь $E(X) = \mu$ — это среднее значение X (вектора с p элементами), а $\mathrm{Cov}(X) = \mathbf{\Sigma}$ — ковариационная матрица вектора X размером $p \times p$. Формально плотность вероятности многомерного гауссова распределения выражается следующим образом:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\mathbf{\Sigma}^{-1}(x-\mu)\right). \tag{4.18}$$

При $p > 1$ LDA–классификатор предполагает, что наблюдения в k–м классе имеют многомерное гауссово распределение $N(\mu_k, \mathbf{\Sigma})$, где μ_k — это специфичный для данного класса вектор математического ожидания, а $\mathbf{\Sigma}$ — общая для всех K классов ковариационная матрица. Подстановка функции плотности вероятности для k–го класса $f_k(X = x)$ в (4.10) и выполнение некоторых алгебраических преобразований покажут, что байесовский классификатор относит наблюдение $X = x$ к классу, у которого величина

$$\delta_k(x) = x^T\mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}^{-1}\mu_k + \log\pi_k \tag{4.19}$$

максимальна. Это — векторно–матричная форма записи уравнения (4.13).

Слева на рис. 4.6 приведен пример. Изображены три нормально распределенных класса одинакового размера, каждый со своим специфичным вектором математического ожидания и общей ковариационной матрицей. Три эллипса охватывают области, соответствующие 95%–ной вероятности каждого из этих классов. Прерывистые линии представляют собой байесовские решающие границы. Иными словами — это совокупности значений x, для которых $\delta_k(x) = \delta_l(x)$, т. е.

$$x^T\mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}^{-1}\mu_k = x^T\mathbf{\Sigma}^{-1}\mu_l - \frac{1}{2}\mu_l^T\mathbf{\Sigma}^{-1}\mu_l \tag{4.20}$$

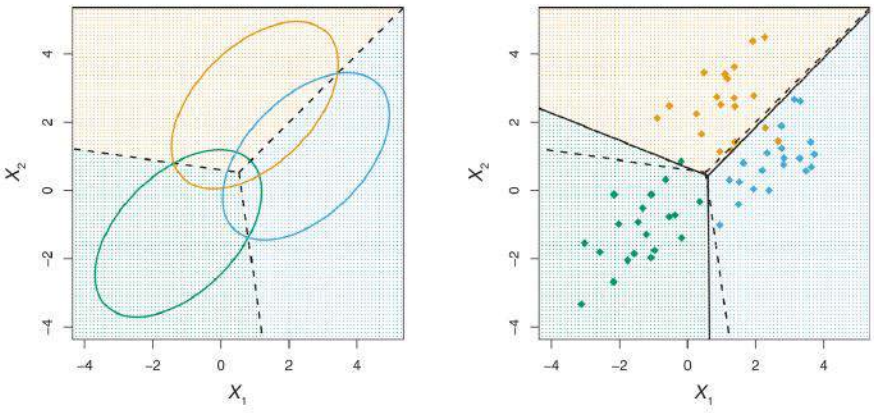

РИСУНОК 4.6. *Показаны две многомерные гауссовы функции плотности с $p = 2$. Слева: два предиктора не коррелируют друг с другом.* Справа: *корреляция между предикторами составляет 0.7*

для $k \neq l$. ($\log \pi_k$ из (4.19) исчез, поскольку все три класса имеют одинаковое число обучающих наблюдений, т. е. π_k одинаковы во всех классах.) Заметьте, что имеют место три линии, соответствующие байесовским решающим границам, поскольку имеются три *пары классов*. Другими словами, одна байесовская решающая граница отделяет класс 1 от класса 2, одна отделяет класс 1 от класса 3, и еще одна отделяет класс 2 от класса 3. Эти три байесовские решающие границы разделяют пространство предикторов на три области. Байесовский классификатор будет классифицировать то или иное наблюдение в соответствии с областью, в которой оно находится.

Напомним еще раз, что нам необходимо оценить неизвестные параметры $\mu_1, \ldots, \mu_k$, $\pi_1, \ldots, \pi_K$ и $\mathbf{\Sigma}$; формулы для них похожи на соответствующие формулы для одномерного случая, приведенные в (4.15). Для классификации нового наблюдения $X = x$ LDA подставляет эти оценки в (4.19) и относит это наблюдение к классу с наиболее высоким значением $\hat{\delta}_k(x)$. Заметьте, что $\delta_k(x)$ в (4.19) является линейной функцией от x, т. е. решающее правило LDA определяется только линейной комбинацией элементов x. Это также является причиной использования прилагательного «*линейный*» в названии LDA.

Справа на рис. 4.6 показаны 20 наблюдений, случайным образом извлеченных из каждого из трех классов, а также соответствующие решающие границы LDA (сплошные черные линии). В целом решающие границы LDA довольно близки к байесовским решающим границам, показанным в виде прерывистых линий. Частоты ошибок на контрольной выборке у байесовского и LDA–классификаторов составляют 0.0746 и 0.0770 соответственно. Таким образом, LDA хорошо работает на этих данных.

Мы можем выполнить линейный дискриминантный анализ для данных `Default` для предсказания вероятности неуплаты долга по кредитной карте, исходя из размера этого долга, а также информации о том, является ли клиент банка студентом. Модель LDA, подогнанная по 10 000 обуча-

ющих наблюдений, приводит к частоте ошибок на *обучающих данных* в 2.27%. Это довольно низкая ошибка, однако следует сделать следующие два предостережения:

- Во–первых, частоты ошибок на обучающих данных обычно будут ниже частот ошибок на контрольных данных, которые в действительности интересуют нас больше всего. Другими словами, мы можем ожидать, что этот классификатор сработает хуже, когда мы применим его для выполнения предсказаний для новой группы клиентов. Это объясняется тем, что мы намеренно подгоняем параметры нашей модели таким образом, чтобы она хорошо работала на обучающих данных. Чем выше отношение числа параметров p к объему наблюдений n, тем важнее будет роль подобного *переобучения*. Для рассматриваемых данных это вряд ли будет проблемой, поскольку $p = 2$, а $n = 10\,000$.

переобучение

- Во–вторых, поскольку 3.33% всех клиентов в обучающей выборке не уплатили свой долг, то простой, но при этом бесполезный классификатор, который всегда предсказывает, что каждый клиент выплатит свой долг вне зависимости от размера долга и социального статуса клиента (студент или нет), даст частоту ошибки в 3.33%. Иными словами, тривиальный *нулевой* классификатор обеспечит частоту ошибок, которая лишь незначительно выше частоты ошибок LDA.

нулевой классификатор

На практике подобный бинарный классификатор может делать два типа ошибок: он может неверно отнести того или иного неплательщика к категории *плательщиков* или неверно отнести плательщика к категории *неплательщиков*. Часто интерес представляет выяснение того, к какому именно из этих двух типов относятся сделанные ошибки. *Матрица неточностей*[4], приведенная для данных `Default` в табл. 4.4, является удобным способом представления такой информации. Как видно из этой табл., LDA–классификатор предсказал, что всего неплательщиками станут 104 человека. Среди них 81 клиент действительно стал неплательщиком, а 23 клиента — нет. Следовательно, неверно были классифицированы только 23 из 9667 человек, которые уплатили свой долг. Это выглядит как довольно низкая частота ошибок! Однако 252 из 333 (75.7%) неплательщиков были пропущены LDA–классификатором. Таким образом, несмотря на то что общая частота ошибок незначительна, частота ошибок среди неплательщиков является очень высокой. С точки зрения компании–эмитента кредитных карт, которая пытается определить рискованных клиентов, частота ошибки в $252/333 = 75.7\%$ среди неплательщиков может быть весьма неприемлемой.

матрица неточностей

Качество модели в отношении конкретного класса является важным также в медицине и биологии, где употребляются термины *чувствительность* и *специфичность*. В рассматриваемом случае чувствительность — это доля истинных неплательщиков, обнаруженных классификатором (здесь она довольно низка — 24.3%). Специфичность — это доля правильно идентифицированных клиентов, уплативших свой долг (здесь она составляет $(1 - 23/9667) = 99.8\%$).

чувствительность и специфичность

[4] В оригинале используется термин «confusion matrix». — *Прим. пер.*

ТАБЛИЦА 4.4. *Матрица неточностей сравнивает предсказания LDA-классификатора с истинным статусом дефолта для 10 000 обучающих наблюдений из набора данных* `Default`. *Числа, расположенные на диагонали этой матрицы, соответствуют клиентам с правильно предсказанным статусом дефолта, а числа вне этой диагонали соответствуют неправильно классифицированным клиентам. LDA сделал неверные предсказания для 23 клиентов, уплативших свой долг, а также для 252 клиентов-неплательщиков.*

		Истинный статус дефолта		
		Нет	Да	Всего
Предсказанный статус дефолта	Нет	9644	252	9896
	Да	23	81	104
	Всего	9667	333	10 000

Почему LDA так плохо распознал неплательщиков? Иными словами, почему у него такая низкая чувствительность? Как мы уже видели, LDA пытается аппроксимировать байесовский классификатор, который имеет самую низкую *общую* частоту ошибок из всех возможных классификаторов (при условии, что гауссова модель верна). Другими словами, байесовский классификатор даст наименьшее возможное количество неправильно классифицированных наблюдений, вне зависимости от того, в каком классе допущены ошибки. Это значит, что некоторые неверно классифицированные наблюдения будут результатом ошибочного отнесения плательщика к неплательщикам, а другие такие наблюдения — результатом ошибочного отнесения неплательщика к категории плательщиков. Однако кредитная компания может быть особенно заинтересована в том, чтобы избегать ошибочной классификации неплательщиков, тогда как неверная классификация плательщиков, хотя и нежелательна, все же является менее проблематичной. Как мы сейчас увидим, LDA можно модифицировать таким образом, чтобы он лучше соответствовал нуждам кредитной компании.

Байесовский классификатор относит то или иное наблюдение к классу с наибольшей апостериорной вероятностью $p_k(X)$. В случае с двумя классами это равносильно отнесению клиента к группе неплательщиков, если

$$\Pr(\texttt{default = Yes}|X = x) > 0.5. \tag{4.21}$$

Таким образом, для отнесения того или иного клиента к классу *неплательщиков* байесовский классификатор и его расширенная версия — LDA — используют пороговое значение апостериорной вероятности дефолта, равное 50%. Однако если мы озабочены неверным предсказанием статуса дефолта для клиентов, которые действительно становятся неплательщиками, тогда мы должны понизить этот порог. Например, мы можем отнести к классу *неплательщиков* любого клиента, у которого апостериорная вероятность дефолта превышает 20%. Другими словами, вместо отнесения некоторого наблюдения к классу *неплательщиков* в случае выполнения условия (4.21) мы могли бы отнести его к этому классу в случае, если

$$\Pr(\texttt{default = Yes}|X = x) > 0.2. \tag{4.22}$$

Частоты ошибок, получаемые при использовании этого подхода, показаны в табл. 4.5. Теперь LDA–классификатор предсказывает, что неплательщиками станут 430 клиентов. Среди 333 клиентов, которые действительно стали неплательщиками, LDA делает неверное предсказание для 138 человек (41.4%). Это существенное улучшение, по сравнению с частотой ошибок 75.7%, которая имела место при использовании порогового значения в 50%. Однако это улучшение имеет свою цену: теперь 235 плательщиков классифицированы неверно. Как следствие общая частота ошибок несколько возросла — до 3.73%. Однако кредитная компания может рассматривать этот незначительный рост общей частоты ошибок как небольшую цену за более точное выявление неплательщиков.

ТАБЛИЦА 4.5. *Матрица неточностей сравнивает предсказания LDA–классификатора с истинным статусом дефолта для 10 000 обучающих наблюдений из набора данных* `Default` *на основе модифицированного порогового значения, согласно которому к неплательщикам относятся все клиенты с предсказанной вероятностью дефолта* $> 20\%$.

		Истинный статус дефолта		
		Нет	Да	Всего
Предсказанный статус дефолта	Нет	9432	138	9570
	Да	235	195	430
	Всего	9667	333	10 000

Рисунок 4.7 иллюстрирует компромисс, возникающий в результате изменения порогового значения апостериорной вероятности дефолта. Показаны разные уровни частоты ошибок в зависимости от этого порогового значения. Применение порогового значения, равного 0.5 (как в 4.21), минимизирует общую частоту ошибок (черная сплошная линия). Это ожидаемо, т. к. байесовский классификатор использует пороговое значение 0.5 и, как известно, обладает наименьшей общей частотой ошибок. Однако при использовании этого порога (0.5) частота ошибок среди неплательщиков довольно высока (голубая прерывистая линия). По мере уменьшения порогового значения частота ошибок среди неплательщиков постепенно снижается, но частота ошибок среди плательщиков возрастает. Как же нам выбрать оптимальное пороговое значение? Подобное решение должно основываться на *знании предметной области* — например, на основе детальной информации о финансовых потерях, связанных с невозможностью возврата долгов.

Популярным графическом способом одновременного описания двух типов ошибок для всех возможных пороговых значений вероятности является *ROC–кривая*. «ROC» — это исторический термин, возникший в теории связи. Данная аббревиатура обозначает «receiver operating characteristic»[5]. На рис. 4.8 показана ROC–кривая для LDA–классификатора на обучающей выборке. Общее качество классификатора, суммированное по всем возможным пороговым значениям вероятности,

ROC–кривая

[5] «Радиочастностная характеристика приемника». — *Прим. пер.*

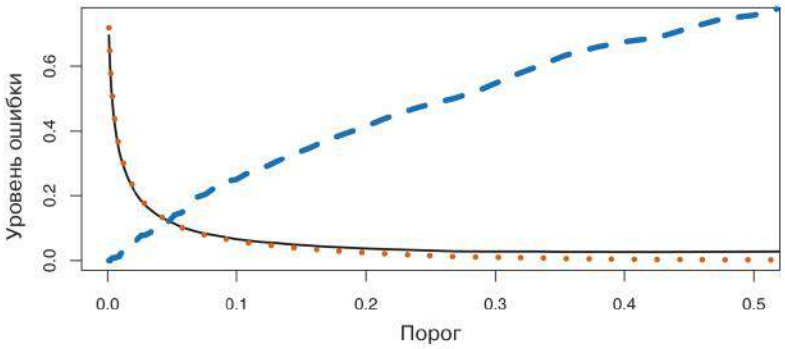

РИСУНОК 4.7. *На примере набора данных* `Default` *показана зависимость частоты ошибок от порогового значения апостериорной вероятности, используемой для классификации. Черная сплошная линия соответствует общей частоте ошибок. Голубая прерывистая показывает долю неверно классифицированных неплательщиков, а оранжевая прерывистая линия — долю неверно классифицированных плательщиков*

площадь под ROC–кривой

выражают в виде *площади под ROC–кривой* — AUC[6]. Идеальная ROC–кривая будет касаться левого верхнего угла графика, и поэтому чем выше AUC, тем выше качество классификатора. Мы ожидаем, что у классификатора, чье качество не отличается от простого случайного угадывания, AUC составит 0.5 (при проверке на независимом контрольном наборе данных, которые не были использованы в обучении модели). ROC–кривые полезны для сравнения разных классификаторов, поскольку они принимают во внимание все возможные пороговые значения вероятности. Оказывается, что ROC–кривая логистической регрессионной модели из подраздела 4.3.4, подогнанной к этим же данным, практически неотличима от кривой LDA–модели, и поэтому мы ее здесь не приводим.

чувствительность

специфичность

Как мы видели выше, изменение порогового значения вероятности у классификатора изменяет частоту истинно положительных и ложноположительных случаев. Эти частоты известны также как *чувствительность* и (1 − *специфичность*) нашего классификатора. Поскольку разнообразие терминов, используемых в этом контексте, часто приводит к замешательству, мы приведем здесь их небольшой обзор. В табл. 4.6 показаны возможные результаты применения классификатора (или диагностического теста) к некоторой совокупности объектов. Чтобы провести параллель с эпидемиологической литературой, можно думать о «+» как о наличии «болезни», которую мы пытаемся обнаружить, а о «−» — как об отсутствии этой «болезни». Для сравнения с классической литературой по проверке статистических гипотез можно думать о «−» как о нулевой гипотезе, а о «+» — как об альтернативной гипотезе. Для набора данных `Default` «+» обозначает клиента–неплательщика, а «−» — клиента, который таковым не является.

[6] Широко используемое сокращение «AUC» происходит от «area under the curve» («площадь под кривой»). — *Прим. пер.*

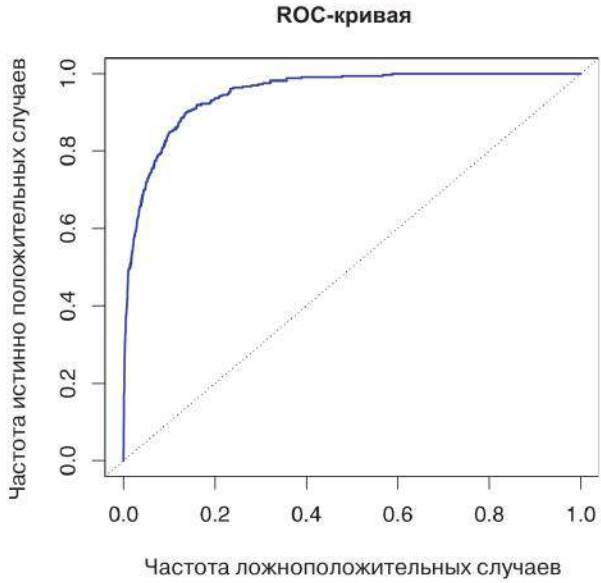

РИСУНОК 4.8. *ROC–кривая LDA–классификатора для данных* `Default`. *Она изображает изменение двух типов ошибок по мере изменения порогового значения апостериорной вероятности дефолта. Сами эти значения не показаны. Частота истинно положительных случаев — это чувствительность, т. е. доля правильно классифицированных неплательщиков при заданном пороговом значении. Частота ложноположительных случаев — это (*1*− специфичность), т. е. доля неверно классифицированных плательщиков при том же пороговом значении. Идеальная ROC–кривая касается верхнего левого угла, указывая на высокую частоту истинно положительных случаев и низкую частоту ложноположительных случаев. Прерывистая линия соответствует классификатору, «не несущему никакой информации», — это то, что мы ожидали бы в случае, когда социальный статус клиента и величина долга по кредитной карте не были связаны с вероятностью дефолта*

ТАБЛИЦА 4.6. *Возможные результаты применения классификатора или диагностического теста к некоторой совокупности объектов*[7]

		Предсказанный класс		
		$-(H_0)$	$+(H_a)$	Всего
Истинный класс	$-(H_0)$	TN	FP	N
	$+(H_a)$	FN	TP	P
	Всего	N*	P*	

[7]Условные обозначения, использованные в этой таблице: H_0 и H_a — нулевая и альтернативная гипотезы соответственно; TN — истинно отрицательные случаи; FP — ложноположительные случаи; FN — ложноотрицательные случаи; TP — истинно положительные случаи; N и P — общее число наблюденных отрицательных и положитель-

В табл. 4.7 перечислены многие из популярных показателей качества, применяемых в этом контексте. Знаменатели частот ложноположительных и истинно положительных случаев — это общие количества случаев, действительно наблюденных в соответствующих классах. В то же время знаменатель в предсказательной ценности положительного и предсказательной ценности отрицательного тестов — это общее число наблюдений, предсказанных для соответствующего класса.

ТАБЛИЦА 4.7. *Важные параметры классификаторов и диагностических тестов, полученные из величин, представленных в табл. 4.6*

Название	Определение	Синонимы
Частота ложноположительных случаев	FP/N	Ошибка I типа; 1 – специфичность
Частота истинно положительных случаев	TP/P	1 – ошибка II типа; мощность; чувствительность; полнота
Предсказательная ценность положительного теста	TP/P*	Точность; 1 – доля ложных срабатываний
Предсказательная ценность отрицательного теста	TN/N*	

4.4.4 Квадратичный дискриминантный анализ

Как обсуждалось ранее, LDA предполагает, что наблюдения в каждом классе происходят из многомерного гауссова распределения со специфичными для каждого класса векторами математических ожиданий и общей для всех K классов ковариационной матрицей. Квадратичный дискриминантный анализ (QDA[8]) обеспечивает альтернативный подход. Подобно LDA, QDA–классификатор получают, делая предположение о том, что наблюдения в каждом классе имеют гауссово распределение, и подставляя оценки соответствующих параметров в уравнение теоремы Байеса для вычисления предсказаний. Однако, в отличие от LDA, QDA предполагает, что каждый класс имеет свою собственную ковариационную матрицу. Другими словами, этот метод предполагает, что наблюдение из k–го класса имеет форму $X \sim N(\mu_k, \mathbf{\Sigma}_k)$, где $\mathbf{\Sigma}_k$ — это ковариационная матрица k–го класса. Исходя из этого предположения, байесовский классификатор относит наблюдение $X = x$ к классу, для которого величина

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\mathbf{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\mathbf{\Sigma}_k| + \log\pi_k = \\ &= -\frac{1}{2}x^T\mathbf{\Sigma}_k^{-1}x + x^T\mathbf{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\log|\mathbf{\Sigma}_k| + \log\pi_k\end{aligned} \tag{4.23}$$

максимальна. Таким образом, QDA–классификатор подразумевает подстановку оценок $\mathbf{\Sigma}_k$, μ_k и π_k в (4.23) и последующее отнесение наблюдения $X = x$ к классу, для которого полученная величина является наибольшей.

ных случаев соответственно; N* и P* — общее число предсказанных отрицательных и положительных случаев соответственно. — *Прим. пер.*

[8] Аббревиатура от «quadratic discriminant analysis». — *Прим. пер.*

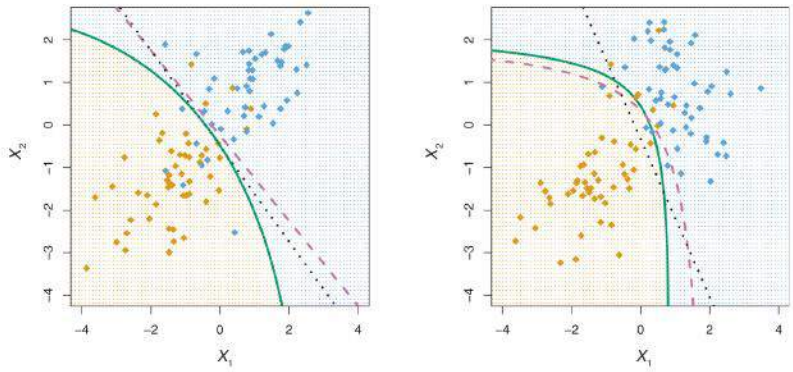

РИСУНОК 4.9. Слева: *решающие границы байесовского классификатора (фиолетовая прерывистая линия), LDA (черная прерывистая линия) и QDA (зеленая прерывистая линия) для проблемы с двумя классами, где* $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$. *Заштрихованные области соответствуют правилу принятия решений QDA. Поскольку байесовская решающая граница линейна, она лучше аппроксимируется при помощи LDA, нежели QDA.* Справа: *ситуация та же, что и на графике слева, но* $\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2$. *Поскольку байесовская решающая граница нелинейна, она лучше аппроксимируется при помощи QDA, нежели LDA*

В отличие от (4.19), в (4.23) член x представлен в виде *квадратичной* функции. Отсюда и происходит название QDA.

Почему так важно предположение о том, что K классов имеют (или нет) общую ковариационную матрицу? Иными словами, почему предпочтительным мог бы быть метод QDA, а не LDA (и наоборот)? Ответ скрывается в компромиссе между смещением и дисперсией. При наличии p предикторов нахождение ковариационной матрицы требует оценивания $p(p+1)/2$ параметров. QDA оценивает отдельную ковариационную матрицу для каждого класса, что в целом дает $Kp(p+1)/2$ параметров. При наличии 50 предикторов результат будет кратным 1275, а это очень много параметров. Если же мы предположим, что K классов имеют общую ковариационную матрицу, то модель LDA становится линейной по x, а это означает, что нахождению подлежат Kp линейных коэффициентов. Следовательно, LDA является намного менее гибким классификатором, чем QDA, и поэтому обладает намного меньшей дисперсией. Потенциально это может привести к более высокому качеству предсказаний. Однако имеет место компромисс: если допущение LDA о том, что K классов обладают общей ковариационной матрицей, далеко от действительности, то LDA может демонстрировать очень высокое смещение. В целом LDA обычно будет более подходящим методом, чем QDA, если в наличии имеется относительно небольшое число наблюдений и, как следствие, критичным является снижение дисперсии. В то же время QDA рекомендуется применять, когда обучающая выборка велика и дисперсия классификатора не вызывает большого беспокойства, или когда допущение об общей ковариационной матрице у K классов явно несостоятельно.

Рисунок 4.9 иллюстрирует качество LDA– и QDA–классификаторов в двух сценариях. На диаграмме, приведенной слева, корреляция между X_1 и X_2 у двух нормально распределенных классов составляет 0.7. Как следствие байесовская решающая граница является линейной и точно аппроксимируется решающей границей LDA. Решающая граница QDA обладаем более низким качеством, поскольку она страдает от высокой дисперсии, которая не компенсируется соответствующим снижением уровня смещения. В то же время справа на рис. 4.9 показан сценарий, в котором корреляция между переменными в «оранжевом» классе составляет 0.7, а в «голубом» классе -0.7. Теперь байесовская решающая граница является квадратичной, и поэтому QDA аппроксимирует эту границу более точно, чем LDA.

4.5 Сравнение методов классификации

В этой главе мы рассмотрели три разных метода классификации: логистическую регрессию, LDA и QDA. В главе 2 мы также обсудили метод K ближайших соседей (KNN). Теперь мы рассмотрим ситуации, в которых один из этих методов мог бы сработать лучше других.

Несмотря на их разные предпосылки, логистическая регрессия и метод LDA тесно связаны. Пусть у нас есть два класса и $p = 1$ предиктор, и пусть $p_1(x)$ и $p_2(x) = 1-p_1(x)$ обозначают вероятности того, что наблюдение $X = x$ принадлежит к классу 1 и классу 2 соответственно. Применив немного алгебры к (4.12) и (4.13), мы можем увидеть, что в рамках линейного дискриминантного анализа логарифм риска принимает следующий вид:

$$\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = \log\left(\frac{p_1(x)}{p_2(x)}\right) = c_0 + c_1 x, \tag{4.24}$$

где c_0 и c_1 являются функциями от μ_1, μ_2 и σ^2. Из (4.4) мы знаем, что в логистической регрессии

$$\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = \beta_0 + \beta_1 x. \tag{4.25}$$

Оба выражения — (4.24) и (4.25) — являются линейными функциями от x. Следовательно, как логистическая регрессия, так и LDA порождают линейные решающие границы. Единственное различие между этими двумя методами заключается в том, что β_0 и β_1 оцениваются при помощи метода максимального правдоподобия, а c_0 и c_1 вычисляются на основе оценок среднего значения и дисперсии нормального распределения. Аналогичная связь между LDA и логистической регрессией имеет место и в случае многомерных данных с $p > 1$.

Поскольку модели логистической регрессии и LDA различаются только способом подгонки, можно было бы ожидать, что эти два метода будут давать похожие результаты. Часто, но всегда, это действительно так. LDA подразумевает, что наблюдения происходят из гауссова распределения с общей для всех классов ковариационной матрицей, и поэтому может дать более качественные решения, по сравнению с логистической регрессией,

когда это предположение примерно соответствует действительности. Однако логистическая регрессия может превзойти LDA, когда эти допущения в отношении гауссова распределения не выполняются.

Вспомните из главы 2, что KNN использует совершенно иной подход, по сравнению с классификаторами, рассмотренными в этой главе. Чтобы получить предсказание для некоторого наблюдения $X = x$ находят K ближайших к x обучающих наблюдений. После этого X относят к классу, к которому принадлежит большинство этих наблюдений. Следовательно, KNN является полностью непараметрическим методом: он не делает никаких предположений в отношении формы решающей границы. Поэтому мы можем ожидать, что этот метод будет превосходить LDA и логистическую регрессию в случаях, когда решающая граница в значительной мере нелинейна. С другой стороны, KNN не говорит нам, какие из предикторов являются важными, — мы не получаем таблицу с коэффициентами, подобную табл. 4.3.

Наконец, QDA служит своего рода компромиссом между непараметрическим методом KNN и линейными методами логистической регрессии и LDA. Поскольку QDA предполагает наличие квадратичной решающей границы, то в сравнении с линейными методами он способен точно моделировать более широкий круг проблем. Обладая меньшей гибкостью в сравнении с KNN, QDA все же может давать более точные предсказания при небольшом числе обучающих наблюдений, поскольку он делает определенные предположения в отношении формы решающей границы.

Для сравнения качества предсказаний этих четырех методов классификации мы искусственно создали данные для шести различных сценариев. В трех из этих сценариев байесовская решающая граница является линейной, а в остальных сценариях — нелинейной. Для каждого сценария мы создали 100 случайных обучающих выборок. Мы подогнали к этим обучающим данным модели каждого типа и рассчитали итоговые частоты ошибок на контрольной выборке большого объема. Результаты для линейных сценариев показаны на рис. 4.10, а результаты для нелинейных сценариев — на рис. 4.11. Метод KNN требует выбора значения K — числа ближайших соседей. Мы выполнили подгонку KNN–моделей для $K = 1$, а также для значения, выбранного автоматически с использованием т. н. метода *перекрестной проверки*, который мы будем обсуждать в главе 5.

В каждом из шести сценариев было $p = 2$ предиктора. Эти сценарии заключались в следующем:

Сценарий 1: В каждом из двух классов было по 20 обучающих наблюдений. В каждом классе наблюдения были представлены независимыми друг от друга случайными нормально распределенными переменными с разными средними значениями. Слева на рис. 4.10 видно, что LDA в этой ситуации сработал ожидаемо хорошо, поскольку это как раз та модель, которую предполагает данный метод. KNN сработал плохо из–за высокой дисперсии, которая не была компенсирована снижением уровня смещения. QDA также сработал хуже LDA, т. к. он породил более гибкий классификатор, чем требовалось. Поскольку логистическая регрессия предполагает линейную решающую границу, ее результаты лишь незначительно уступили результатам LDA.

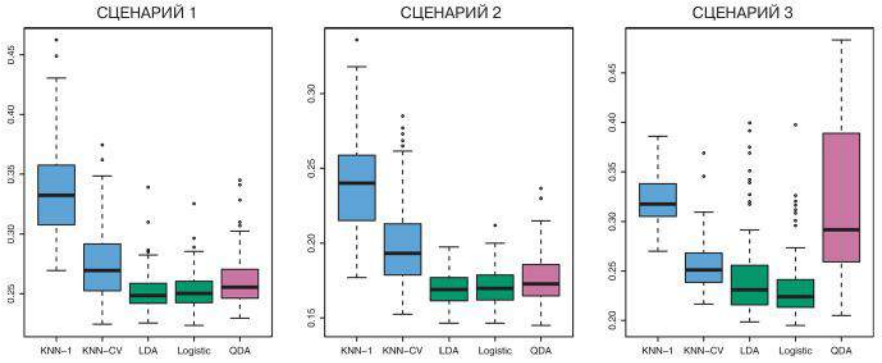

РИСУНОК 4.10. *Диаграммы размахов для частот ошибок на контрольных данных в каждом из линейных сценариев, описанных в тексте. Обозначения: KNN–1 — KNN с $K = 1$; KNN–CV — KNN с автоматическим выбором K; LDA — линейный дискриминантный анализ; Logistic — логистическая регрессия; QDA — квадратичный дискриминантный анализ*

Сценарий 2: Детали те же, что и в сценарии 1, однако предикторы в каждом классе коррелировали на уровне -0.5. Центральная диаграмма на рис. 4.10 указывает на незначительные изменения относительного качества предсказаний исследованных методов, по сравнению с предыдущим сценарием.

t–распределение

Сценарий 3: Мы создали X_1 и X_2 на основе *t–распределения*, с 50 наблюдениями в каждом классе. t–распределение по форме похоже на нормальное распределение, однако обычно оно генерирует более экстремальные наблюдения, т. е. большее количество точек, находящихся далеко от среднего значения. В этой ситуации решающая граница все еще была линейной и поэтому соответствовала ожиданиям логистической регрессии. Однако этот сценарий нарушил допущения метода LDA, поскольку наблюдения не были извлечены из нормального распределения. Диаграмма, приведенная на рис. 4.10 справа, показывает, что логистическая регрессия превзошла LDA, хотя в целом эти два метода сработали лучше других методов. В частности, результаты QDA значительно ухудшились из–за ненормального распределения наблюдений.

Сценарий 4: Данные были созданы на основе нормального распределения, причем корреляция между предикторами составила 0.5 в первом классе и -0.5 во втором классе. Эта ситуация соответствовала допущениям QDA и привела к квадратичной решающей границе. Диаграмма, представленная на рис. 4.11 слева, показывает, что QDA превзошел все другие методы.

Сценарий 5: В каждом классе наблюдения были сгенерированы на основе нормального распределения с независимыми предикторами. Однако значения отклика были созданы на основе логистической

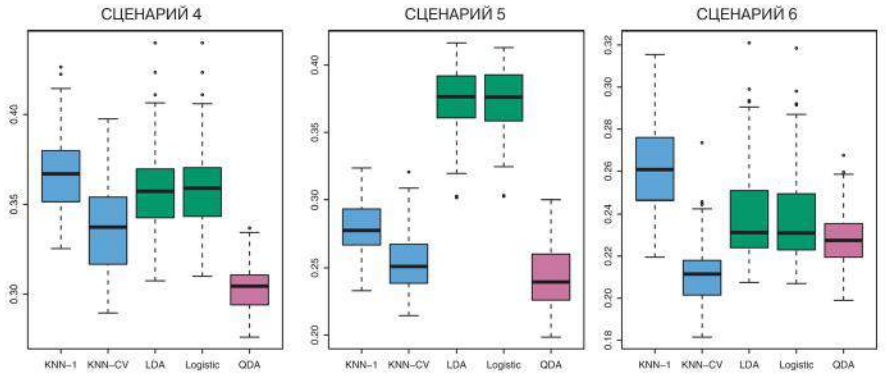

РИСУНОК 4.11. *Диаграммы размахов для частот ошибок на контрольных данных в каждом из нелинейных сценариев, описанных в тексте. Обозначения те же, что и на рис. 4.10*

функции с использованием X_1^2, X_2^2 и $X_1 \times X_2$ в качестве предикторов. Следовательно, имеет место квадратичная решающая граница. В центре на рис. 4.11 приведена диаграмма, из которой следует, что лучше всего снова сработал QDA, а на втором месте после него оказалась модель KNN с автоматически выбранным значением K. Качество предсказаний у линейных методов оказалось очень низким.

Сценарий 6: Основные условия те же, что в предыдущем сценарии, однако значения отклика были сгенерированы на основе более сложной нелинейной функции. В результате даже квадратичные решающие границы QDA не смогли адекватно смоделировать эти данные. Как следует из диаграммы, представленной на рис. 4.11 справа, QDA сработал немного лучше линейных методов, тогда как существенно более гибкий метод KNN с автоматическим выбором значения K показал наиболее высокие результаты. В то же время KNN с $K = 1$ показал наиболее низкие результаты из всех методов. Это подчеркивает тот факт, что даже когда данные проявляют сложную нелинейную зависимость, непараметрический метод вроде KNN все же может показать очень плохие результаты, если уровень сглаживания подобран неправильно.

Эти шесть примеров показывают, что не существует метода, который будет превосходить другие методы в любой ситуации. Когда истинные решающие границы являются линейными, LDA и логистическая регрессия в целом будут работать хорошо. Когда эти границы умеренно нелинейны, QDA может дать более качественные решения. Наконец, для существенно более сложных границ наилучшим может оказаться такой непараметрический метод, как KNN. Однако следует очень тщательно выбирать уровень сглаживания для непараметрического метода. В следующей главе мы рассмотрим несколько подходов для выбора корректного уровня сглаживания, а также для выбора оптимального метода в целом.

В заключение вспомним из главы 3, что в ходе выполнения регресси-

онного анализа мы можем подогнать нелинейную зависимость между предикторами и откликом, используя преобразованные предикторы. Похожий подход можно было бы применить и для классификации. Например, мы могли бы создать более гибкую версию логистической регрессии, включив в нее в качестве предикторов X^2, X^3 и даже X^4. Если увеличение дисперсии за счет добавленной гибкости компенсируется достаточно большим снижением уровня смещения, это может улучшить качество предсказаний логистической регрессии. Мы могли бы сделать то же самое и для LDA. Если бы мы добавили все возможные квадратичные члены и взаимодействия предикторов в LDA, то модель приобрела бы форму QDA–модели, хотя оценки параметров оказались бы другими. Такой подход позволяет нам построить нечто среднее между LDA– и QDA–моделями.

4.6 Лабораторная работа: логистическая регрессия, LDA, QDA и KNN

4.6.1 Данные по цене акций

Мы начнем с рассмотрения некоторых количественных и графических сводок по данным `Smarket`, которые входят в состав библиотеки `ISLR`. Этот набор данных включает процентные изменения доходности биржевого индекса S&P 500 для 1250 дней — с начала 2001 г. до конца 2005 г. Для каждой даты у нас есть значения процентного изменения доходности для каждого из пяти предыдущих дней — с `Lag1` по `Lag5`. У нас есть также переменные `Volume` (количество акций (в миллиардах), проданных в предыдущий день), `Today` (процентное изменение доходности за конкретный день) и `Direction` (направление движения рынка — вверх (`Up`) или вниз (`Down`) — в этот день).

```
> library(ISLR)
> names(Smarket)
[1] "Year"     "Lag1"     "Lag2"     "Lag3"     "Lag4"
[6] "Lag5"     "Volume"   "Today"    "Direction"

> dim(Smarket)
[1] 1250 9

> summary(Smarket)
      Year              Lag1                Lag2
Min.    :2001     Min.    :-4.92200     Min.    :-4.92200
1st Qu. :2002     1st Qu. :-0.63950     1st Qu. :-0.63950
Median  :2003     Median  : 0.03900     Median  : 0.03900
Mean    :2003     Mean    : 0.00383     Mean    : 0.00392
3rd Qu. :2004     3rd Qu. : 0.59675     3rd Qu. : 0.59675
Max.    :2005     Max.    : 5.73300     Max.    : 5.73300

      Lag3                Lag4                Lag5
Min.    :-4.92200     Min.    :-4.92200     Min.    :-4.92200
1st Qu. :-0.64000     1st Qu. :-0.64000     1st Qu. :-0.64000
Median  : 0.03850     Median  : 0.03850     Median  : 0.03850
```

```
Mean    : 0.00172     Mean    : 0.00164     Mean    : 0.00561
3rd Qu. : 0.59675     3rd Qu. : 0.59675     3rd Qu. : 0.59700
Max.    : 5.73300     Max.    : 5.73300     Max.    : 5.73300

   Volume                  Today              Direction
Min.    :0.356          Min.    :-4.92200     Down :602
1st Qu. :1.257          1st Qu. :-0.63950       Up :648
Median  :1.423          Median  : 0.03850
Mean    :1.478          Mean    : 0.00314
3rd Qu. :1.642          3rd Qu. : 0.59675
Max.    :3.152          Max.    : 5.73300
> pairs(Smarket)
```

Функция `cor()` создает матрицу со значениями всех парных корреляций между предикторами из некоторого набора данных. Первая из приведенных ниже команд выдает сообщение об ошибке, поскольку переменная `Direction` является качественной.

```
> cor(Smarket)
Error in cor(Smarket): 'x' must be numeric
> cor(Smarket[, -9])
            Year       Lag1       Lag2       Lag3       Lag4       Lag5
Year    1.00000    0.02970    0.03060    0.03319    0.03569    0.02979
Lag1    0.02970    1.00000   -0.02629   -0.01080   -0.00299   -0.00567
Lag2    0.03060   -0.02629    1.00000   -0.02590   -0.01085   -0.00356
Lag3    0.03319   -0.01080   -0.02590    1.00000   -0.02405   -0.01881
Lag4    0.03569   -0.00299   -0.01085   -0.02405    1.00000   -0.02708
Lag5    0.02979   -0.00567   -0.00356   -0.01881   -0.02708    1.00000
Volume  0.53900    0.04091   -0.04338   -0.04182   -0.04841   -0.02200
Today   0.03010   -0.02616   -0.01025   -0.00245   -0.00690   -0.03486
           Volume      Today
Year    0.53900    0.03010
Lag1    0.04091   -0.02616
Lag2   -0.04338   -0.01025
Lag3   -0.04182   -0.00245
Lag4   -0.04841   -0.00690
Lag5   -0.02200   -0.03486
Volume  1.00000    0.01459
Today   0.01459    1.00000
```

Коэффициенты корреляции между лаг–переменными и доходностью за «сегодня» вполне ожидаемо близки к нулю. Другими словами, похоже, что доходность «сегодня» очень слабо связана с доходностью в предыдущие дни. Единственная сильная корреляция имеется между `Year` и `Volume`. Изобразив данные графически, мы увидим, что `Volume` увеличивается во времени. Иными словами, в период с 2001 по 2005 г. среднее количество продаваемых ежедневно акций возросло.

```
> attach(Smarket)
> plot(Volume)
```

4.6.2 Логистическая регрессия

glm()

обобщенные линейные модели

Далее мы построим логистическую модель с целью предсказания `Direction` на основе `Lag1` – `Lag5` и `Volume`. Функция `glm()` подгоняет *обобщенные линейные модели* — класс моделей, который включает и логистическую регрессию. Синтаксис функции `glm()` похож на синтаксис `lm()`, за тем исключением, что мы должны подать аргумент `family = binomial`, который скажет `R` построить логистическую, а не какую–либо другую обобщенную линейную модель.

```
> glm.fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
                Volume, data = Smarket, family = binomial)
> summary(glm.fit)
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Smarket)

Deviance  Residuals:
   Min       1Q  Median      3Q     Max
 -1.45    -1.20    1.07    1.15    1.33

Coefficients:
             Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)  -0.12600     0.24074    -0.52      0.60
Lag1         -0.07307     0.05017    -1.46      0.15
Lag2         -0.04230     0.05009    -0.84      0.40
Lag3          0.01109     0.04994     0.22      0.82
Lag4          0.00936     0.04997     0.19      0.85
Lag5          0.01031     0.04951     0.21      0.83
Volume        0.13544     0.15836     0.86      0.39

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1742
```

Наименьшее p–значение здесь у переменной `Lag1`. Отрицательный коэффициент у этого предиктора предполагает, что если вчера наблюдалась положительная доходность, то сегодня она вряд ли пойдет вверх. Однако p–значение, равное 0.15, все–таки достаточно велико, и поэтому четкого свидетельства в пользу зависимости между `Lag1` и `Direction` нет.

Для получения доступа к коэффициентам этой подогнанной модели мы воспользуемся функцией `coef()`. Для доступа к детальному описанию свойств подогнанной модели, таких как p–значения коэффициентов, мы также можем применить функцию `summary()`.

```
> coef(glm.fit)
(Intercept)        Lag1        Lag2        Lag3        Lag4
   -0.1260    -0.07307    -0.04230     0.01109     0.00936
       Lag5      Volume
    0.01031     0.13544
```

```
> summary(glm.fit)$coef
             Estimate  Std. Error    z value  Pr(>|z|)
(Intercept)  -0.12600      0.2407     -0.523     0.601
Lag1         -0.07307      0.0502     -1.457     0.145
Lag2         -0.04230      0.0501     -0.845     0.398
Lag3          0.01109      0.0499      0.222     0.824
Lag4          0.00936      0.0500      0.187     0.851
Lag5          0.01031      0.0495      0.208     0.835
Volume        0.13544      0.1584      0.855     0.392
> summary(glm.fit)$coef[, 4]
(Intercept)        Lag1        Lag2        Lag3        Lag4
      0.601       0.145       0.398       0.824       0.851
       Lag5      Volume
      0.835       0.392
```

Имея те или иные значения предикторов, можно воспользоваться функцией `predict()` и предсказать вероятность того, что доходность индекса пойдет вверх. Опция `type = "response"` сообщает R, что необходимо выводить вероятности в виде $P(Y = 1|X)$, а не какую–то другую информацию вроде логита. Если на функцию `predict()` не подан какой–то определенный набор данных, то вероятности рассчитываются для обучающих данных, которые были использованы для подгонки логистической модели. Ниже мы вывели только первые десять значений вероятности. Мы знаем, что эти значения соответствуют вероятности роста, а не падения доходности акций, поскольку функция `contrasts()` показывает, что программа создала индикаторную переменную, у которой 1 соответствует значению `Up`.

```
> glm.probs = predict(glm.fit, type = "response")
> glm.probs[1:10]
    1     2     3     4     5     6     7     8     9    10
0.507 0.481 0.481 0.515 0.511 0.507 0.493 0.509 0.518 0.489
> contrasts(Direction)
     Up
Down  0
Up    1
```

Чтобы предсказать направление роста доходности для конкретного дня, мы должны конвертировать эти вероятности в метки классов — `Up` или `Down`. Приведенные ниже команды создают вектор с метками классов, присвоенными в зависимости от того, превышает ли предсказанная вероятность роста доходности 0.5.

```
> glm.pred = rep("Down", 1250)
> glm.pred[glm.probs > .5] = "Up"
```

Первая команда создает вектор с 1250 значениями `Down`. Вторая команда превращает в `Up` все элементы вектора, у которых предсказанная вероятность роста доходности превышает 0.5. Имея эти предсказания, можно воспользоваться функцией `table()`, чтобы получить матрицу неточностей и определить, сколько наблюдений были классифицированы неверно.

`table()`

```
> table(glm.pred, Direction)
```

```
          Direction
glm.pred  Down   Up
    Down   145  141
    Up     457  507
> (507 + 145)/1250
[1] 0.5216
> mean(glm.pred == Direction)
[1] 0.5216
```

Диагональные элементы матрицы неточностей соответствуют верным предсказаниям, а элементы, расположенные вне диагонали, — ошибочным предсказаниям. Таким образом, наша модель для 507 дней правильно предсказала, что доходность возрастет, и для 145 дней, что доходность снизится, — в целом это $507+145 = 652$ правильно предсказанных случаев. Функцию `mean()` можно применить для вычисления доли правильно предсказанных случаев. В этом примере логистическая регрессия правильно предсказала направление движения рынка в 52.2% случаев.

На первый взгляд может показаться, что логистическая регрессионная модель работает немного лучше, чем случайное угадывание. Однако этот результат вводит в заблуждение, поскольку мы обучили и проверили модель на тех же 1250 наблюдениях. Другими словами, $100 - 52.2 = 47.8\%$ — это частота ошибок на *обучающих* данных. Как мы уже видели ранее, частота ошибок на обучающих данных часто слишком оптимистична — она имеет тенденцию недооценивать частоту ошибок на контрольной выборке. Чтобы лучше оценить верность предсказаний логистической регрессионной модели при таком сценарии, мы можем подогнать модель на основе определенной части данных, а затем исследовать качество ее предсказаний на *неиспользованных* для обучения данных. Это даст более реалистичную частоту ошибок, в том смысле, что на практике нам будет интересно качество предсказаний нашей модели не для тех данных, которые мы использовали для ее подгонки, а для будущих дней, для которых направление движения рынка неизвестно.

Для реализации этой стратегии мы сначала создадим вектор, соответствующий наблюдениями за период с 2001 по 2004 г. Затем мы используем этот вектор для создания контрольной выборки, содержащей наблюдения за 2005 г.

```
> train = (Year < 2005)
> Smarket.2005 = Smarket[!train, ]
> dim(Smarket.2005)
[1] 252  9
> Direction.2005 = Direction[!train]
```

Объект `train` представляет собой вектор из 1250 элементов, соответствующих наблюдениям из нашего набора данных. Элементам вектора, которые соответствуют наблюдениям, полученным до 2004 г., присвоено значение `TRUE`, а элементам, которые соответствуют наблюдениям за 2005 г., присвоено значение `FALSE`. Объект `train` — это логический вектор, поскольку он содержит значения `TRUE` и `FALSE`. Логические векторы можно применять для получения определенных подмножеств строк или столбцов матрицы. Например, команда `Smarket[train, ]` отобрала бы ту часть матрицы с данными по доходности акций, которая соответствует только датам до

логический вектор

2005 г., поскольку это те данные, для которых элементы вектора train равны TRUE. Символ ! позволяет обратить все элементы логического вектора. Другими словами, вектор !train похож на train, за исключением того, что элементы со значением TRUE в train заменяются на FALSE в !train, элементы со значением FALSE в train заменяются на TRUE в !train. Следовательно, Smarket[!train,] возвращает часть матрицы с данными по доходности акций, содержащую только те наблюдения, для которых значения train равны FALSE, т. е. наблюдения с датами за 2005 г.

Теперь, применяя аргумент subset, мы подгоняем логистическую регрессионную модель на основе только тех наблюдений, которые соответствуют датам до 2005 г. Далее мы получаем предсказанные вероятности роста доходности для каждого дня из контрольной выборки, т. е. для дней 2005 г.

```
> glm.fit = glm(Direction ~ Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
                data = Smarket, family = binomial, subset=train)
> glm.probs = predict(glm.fit, Smarket.2005, type = "response")
```

Заметьте, что мы обучили и проверили нашу модель на двух совершенно отдельных наборах данных: обучение было выполнено с использованием дат только до 2005 г., а контроль качества был выполнен лишь с использованием дат 2005 г. Наконец, мы вычисляем предсказания для 2005 г. и сравниваем их с направлениями движения рынка, действительно имевшими место в том году.

```
> glm.pred = rep("Down", 252)
> glm.pred[glm.probs > .5] = "Up"
> table(glm.pred, Direction.2005)
         Direction.2005
glm.pred  Down  Up
    Down    77  97
    Up      34  44
> mean(glm.pred == Direction.2005)
[1] 0.48
> mean(glm.pred != Direction.2005)
[1] 0.52
```

Выражение != обозначает «*не равно*», и поэтому последняя команда вычисляет частоту ошибок на контрольных данных. Результаты довольно неутешительны: частота ошибок на контрольных данных составляет 52%, что хуже, чем случайное угадывание. Конечно, этот результат не является большим сюрпризом, поскольку обычно невозможно предсказать будущее развитие рынка на основе доходности за предыдущий день. (В конце концов, если бы такое было возможно, то авторы этой книги уже сколотили бы себе состояние, а не писали учебники по статистике.)

Напомним, что p–значения у всех предикторов в логистической регрессионной модели были совсем не впечатляющими, а самое низкое p–значение (хотя и не очень низкое) соответствовало переменной Lag1. Возможно, удалив переменные, выглядящие бесполезными для предсказания Default, мы сможем получить более эффективную модель. В конце концов, использование предикторов, не связанных с откликом, обычно увеличивает частоту ошибок на контрольной выборке (поскольку они вызывают рост дисперсии без соответствующего снижения смещения), и поэтому

удаление таких предикторов может привести к определенному улучшению. Ниже мы повторно подогнали логистическую регрессию, используя только Lag1 и Lag2, которые в исходной модели выглядели как переменные с самым высоким предсказательным потенциалом.

```
> glm.fit = glm(Direction ~ Lag1 + Lag2,
              data = Smarket, family = binomial, subset = train)
> glm.probs = predict(glm.fit, Smarket.2005, type = "response")
> glm.pred = rep("Down", 252)
> glm.pred[glm.probs > .5] = "Up"
> table(glm.pred, Direction.2005)
         Direction.2005
glm.pred  Down  Up
    Down    35  35
    Up      76 106

> mean(glm.pred == Direction.2005)
[1] 0.56
> 106/(106 + 76)
[1] 0.582
```

Теперь результаты выглядят немного лучше: 56% дневных изменений были предсказаны верно. Стоит отметить, что в данном примере намного более простая стратегия предсказания ежедневного роста доходности также будет верна в 56% случаев! Следовательно, по общей частоте ошибок метод логистической регрессии ничем не лучше наивного подхода. Однако матрица неточностей показывает, что в случаях, когда логистическая регрессия предсказывает рост доходности, частота верных предсказаний составляет 58%. Это предполагает возможную компромиссную стратегию: покупку активов, когда модель предсказывает рост рынка, и избегание торгов в дни, для которых предсказывается снижение рынка. Конечно, нужно было бы изучить более тщательно, является ли это улучшение реальным или просто было вызвано случайными факторами.

Представьте, что мы хотим предсказать доходность, связанную с конкретными значениями Lag1 и Lag2. В частности, мы хотим предсказать Default для дня, когда Lag1 и Lag2 составляют 1.2 и 1.1 соответственно, а также когда они равны 1.5 и -0.8. Мы делаем это при помощи функции predict().

```
> predict(glm.fit, newdata = data.frame(Lag1 = c(1.2, 1.5),
          Lag2 = c(1.1, -0.8)), type = "response")
     1          2
0.4791     0.4961
```

4.6.3 Линейный дискриминантный анализ

lda()

Теперь мы выполним LDA на данных Smarket. В R мы подгоняем LDA–модель при помощи функции lda(), которая является частью библиотеки MASS. Заметьте, что синтаксис функции lda() идентичен синтаксису lm() и glm(), за тем исключением, что опция family отсутствует. Мы подгоняем модель на основе наблюдений, полученных ранее 2005 г.

```
> library(MASS)
> lda.fit = lda(Direction ~ Lag1 + Lag2,
                data = Smarket, subset = train)
> lda.fit
Call:
lda (Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
 Down     Up
0.492  0.508

Group means:
        Lag1    Lag2
Down  0.0428  0.0339
Up   -0.0395 -0.0313

Coefficients of linear discriminants:
         LD1
Lag1  -0.642
Lag2  -0.514

> plot(lda.fit)
```

Результаты LDA показывают, что $\hat{\pi}_1 = 0.492$, $\hat{\pi}_2 = 0.508$, т. е. 49.2% обучающих наблюдений соответствуют дням, когда рынок снижался. В результатах представлены также групповые средние — это средние значения каждого предиктора в каждом классе, которые используются LDA в качестве оценок μ_k. Эти значения указывают на то, что за два дня до роста рынка доходность акций обычно отрицательна, а за день до снижения рынка доходность обычно положительна. В части результатов, озаглавленной как «*Coefficients of linear disriminants*» (коэффициенты линейных дискриминантов), представлена линейная комбинация переменных `Lag1` и `Lag2`, которая используется для формирования правила принятия решений LDA. Другими словами, это коэффициенты элементов $X = x$ в уравнении (4.19). Если величина $-0.642 \times$ `Lag1` $- 0.514 \times$ `Lag2` велика, то LDA–классификатор предскажет рост доходности, а если она мала, то классификатор предскажет снижение доходности. Функция `plot()` строит графики *линейных дискриминантов*, полученных путем вычисления $-0.642 \times$ `Lag1` $- 0.514 \times$ `Lag2` для каждого обучающего наблюдения.

Функция `predict()` возвращает список с тремя элементами. Первый элемент — `class` — содержит предсказания LDA касательно направления движения рынка. Второй элемент — `posterior` — представляет собой матрицу, в которой k–й столбец содержит вычисленную по (4.1) апостериорную вероятность принадлежности соответствующего наблюдения к классу k. Наконец, `x` содержит описанные ранее линейные дискриминанты.

```
> lda.pred = predict(lda.fit, Smarket.2005)
> names(lda.pred)
[1] "class"   "posterior"   "x"
```

Как было отмечено в разделе 4.5, предсказания логистической регрессии и LDA почти идентичны.

```
> lda.class = lda.pred$class

> table(lda.class, Direction.2005)
        Direction.2005
lda.pred  Down  Up
    Down    35  35
    Up      76 106
> mean(lda.class == Direction.2005)
[1] 0.56
```

Применение 50%–го порогового значения к апостериорным вероятностям позволяет нам восстановить предсказания, содержащиеся в `lda.pred$class`.

```
> sum(lda.pred$posterior[, 1] >= .5)
[1] 70
> sum(lda.pred$posterior[, 1] < .5)
[1] 182
```

Обратите внимание на то, что выданная моделью апостериорная вероятность соответствует вероятности снижения доходности:

```
> lda.pred$posterior[1:20, 1]
> lda.class[1:20]
```

Если бы мы хотели при расчете предсказаний использовать пороговое значение апостериорной вероятности, отличное от 50%, то легко могли бы это сделать. Допустим, например, что мы хотим предсказывать снижение доходности только, если мы убеждены, что это действительно произойдет в соответствующий день (скажем, если апостериорная вероятность составляет не менее 90%).

```
> sum(lda.pred$posterior[, 1] > .9)
[1] 0
```

Ни один из дней 2005 г. не соответствует этому пороговому значению! Оказывается, что максимальная апостериорная вероятность снижения для всего 2005 г. составила 52.02%.

4.6.4 Квадратичный дискриминантный анализ

`qda()`

Теперь мы подгоним QDA–модель к данным `Smarket`. Метод QDA реализуется в `R` при помощи функции `qda()`. Синтаксис идентичен таковому у `lda()`.

```
> qda.fit = qda(Direction ~ Lag1 + Lag2,
                data = Smarket, subset = train)
> qda.fit

Call:
qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
 Down     Up
```

```
0.492  0.508

Group means:
        Lag1     Lag2
Down  0.0428   0.0339
Up   -0.0395  -0.0313
```

Результаты содержат групповые средние. Однако они не включают коэффициенты линейных дискриминантов, поскольку QDA основан на квадратичной, а не линейной функции предикторов. Функция `predict()` работает точно таким же образом, как и в случае с LDA.

```
> qda.class = predict(qda.fit, Smarket.2005)$class
> table(qda.class, Direction.2005)
Direction.2005
qda.class  Down Up
    Down    30  20
      Up    81 121
> mean(qda.class == Direction.2005)
[1] 0.599
```

Интересно, что предсказания QDA верны примерно в 60% случаев, хотя данные за 2005 г. не были использованы при подгонке модели. Подобный уровень точности довольно внушителен для данных по рынку акций, который, как известно, трудно поддается точному моделированию. Это указывает на то, что лежащая в основе QDA квадратичная форма может отражать истинную зависимость корректнее линейных форм, предполагаемых LDA и логистической регрессией. Однако прежде чем заключать пари на то, что этот метод обставит рынок, мы советуем оценить качество его предсказаний на большей контрольной выборке!

4.6.5 Метод K ближайших соседей

Теперь мы выполним построение KNN–модели при помощи функции `knn()` из библиотеки `class`. Эта функция довольно сильно отличается от других подгоняющих модели функций, с которыми мы уже познакомились. Вместо двухэтапного подхода, когда мы сначала подгоняем модель, а затем используем эту модель для получения предсказаний, `knn()` формирует предсказания при помощи одной команды. Эта функция требует четыре входных параметра: `knn()`

1. Матрица с предикторами из обучающего набора данных, обозначаемая ниже как `train.X`.

2. Матрица с предикторами из набора данных, для которого мы хотим сделать предсказания (обозначена ниже как `test.X`).

3. Вектор с метками классов обучающих наблюдений (обозначен ниже как `train.Direction`).

4. Значение K — используемое классификатором число ближайших соседей.

cbind()

Мы применяем функцию cbind()[9] для объединения переменных Lag1 и Lag2 в две матрицы — для обучающего и контрольного наборов данных.

```
> library(class)
> train.X = cbind(Lag1, Lag2)[train, ]
> test.X = cbind(Lag1, Lag2)[!train, ]
> train.Direction = Direction[train]
```

Теперь можно применить функцию knn() для предсказания направления движения рынка для дат 2005 г. Перед вызовом knn() мы задаем значение «зерна» генератора случайных чисел, поскольку в ситуациях, когда несколько ближайших соседей оказываются равноудаленными от интересующего нас наблюдения, R случайным образом выберет некоторые из них. Поэтому «зерно» необходимо задать для получения воспроизводимых результатов.

```
> set.seed(1)
> knn.pred = knn(train.X, test.X, train.Direction, k = 1)
> table(knn.pred, Direction.2005)
         Direction.2005
knn.pred  Down  Up
    Down    43  58
    Up      68  83
> (83 + 43)/252
[1] 0.5
```

Результаты, полученные с использованием $K = 1$, не очень хороши, поскольку только 50% наблюдений предсказаны верно. Конечно, может оказаться так, что $K = 1$ приводит к чрезмерно гибкой модели. Ниже мы повторяем анализ, используя $K = 3$.

```
> knn.pred = knn(train.X, test.X, train.Direction, k = 3)
> table(knn.pred, Direction.2005)
         Direction.2005
knn.pred  Down  Up
    Down    48  54
    Up      63  87
> mean(knn.pred == Direction.2005)
[1] 0.536
```

Качество модели несколько повысилось. Однако дальнейшее увеличение K не приводит к соответствующим улучшениям. Похоже, что из всех рассмотренных нами на данный момент методов QDA обеспечивает наилучшие результаты для этого набора данных.

4.6.6 Применение к данным по жилым прицепам

Наконец, мы применим метод KNN к данным Caravan, который является частью библиотеки ISLR. Этот набор данных включает 85 предикторов, описывающих демографические характеристики 5822 человек. Зависимой является переменная Purchase, которая показывает, покупает ли тот или

[9] Сокращение от «column» (столбец) и «bind» (связывать). — *Прим. пер.*

иной человек страховой полис для своего жилого прицепа. В этом наборе данных страховку для своих прицепов купили только 6% людей.

```
> dim(Caravan)
[1] 5822 86
> attach(Caravan)
> summary(Purchase)
  No  Yes
5474  348
> 348/5822
[1] 0.0598
```

Поскольку классификатор KNN предсказывает класс некоторого контрольного наблюдения путем определения ближайших к нему наблюдений, имеет значение шкала измерения переменных. Любые переменные, которые выражаются большими числами, будут иметь гораздо больший эффект на *расстояние* между наблюдениями и, следовательно, на KNN–классификатор, чем переменные, которые выражаются при помощи малых чисел. Например, представьте себе набор данных с двумя переменными — *«заработная плата»* и *«возраст»*, измеренными в долларах и в годах соответственно. С «точки зрения» KNN, разница в заработной плате, составляющая 1000$, огромна, по сравнению с разницей в возрасте, составляющей 50 лет. Это противоречит нашему восприятию: разница в 1000$ воспринимается нами как довольно небольшая, по сравнению с разницей в возрасте, составляющей 50 лет. Более того, важность шкалы измерения для KNN приводит к другой проблеме: если бы мы измерили заработную плату в японских иенах или выразили возраст в минутах, то результаты классификации довольно существенно отличались бы от того, что мы получаем при выражении этих переменных в долларах и в годах.

Хороший способ решения этой проблемы заключается в *стандартизации* данных таким образом, чтобы все переменные имели среднее значение, равное 0, и стандартное отклонение, равное 1. В таком случае все переменные будут представлены на одинаковой шкале. Функция `scale()` как раз и выполняет такую процедуру. При стандартизации данных мы исключаем столбец 86, поскольку он представляет собой качественную переменную `Purchase`.

стандартизация

`scale()`

```
> standardized.X = scale(Caravan[, -86])
> var(Caravan[, 1])
[1] 165
> var(Caravan[, 2])
[1] 0.165
> var(standardized.X[, 1])
[1] 1
> var(standardized.X[, 2])
[1] 1
```

Теперь каждый столбец таблицы `standardized.X` имеет стандартное отклонение, равное 1, и среднее значение, равное 0.

Далее мы разбиваем данные на обучающую выборку, содержащую 1000 первых наблюдений, и контрольную выборку, содержащую все остальные наблюдения. Мы строим KNN–модель по обучающим данным с использо-

ванием $K = 1$ и оцениваем качество предсказаний на контрольных данных.

```
> test = 1:1000
> train.X = standardized.X[-test, ]
> test.X = standardized.X[test, ]
> train.Y = Purchase[-test]
> test.Y = Purchase[test]
> set.seed(1)
> knn.pred = knn(train.X, test.X, train.Y, k = 1)
> mean(test.Y != knn.pred)
[1] 0.118
> mean(test.Y != "No")
[1] 0.059
```

Вектор `test` является числовым и содержит числа от 1 до 1000. Команда `standardized.X[test, ]` возвращает часть данных с наблюдениями, чьи индексные номера изменяются от 1 до 1000, тогда как команда `standardized.X[-test, ]` возвращает часть данных с наблюдениями, чьи индексные номера *не* лежат в диапазоне от 1 до 1000. Частота ошибки KNN на 1000 контрольных наблюдений лишь немногим меньше 12%. На первый взгляд это значение может показаться довольно хорошим. Но поскольку страховку купили только 6% клиентов, мы всегда могли бы свести частоту ошибок до 6%, предсказывая `No` (страховка не куплена) вне зависимости от значений предикторов!

Предположим, что попытка продать страховку определенному клиенту связана с существенными издержками. Возможно, например, что продавец должен посетить каждого потенциального клиента. Если компания пытается продать страховку случайно выбранной группе клиентов, то успех будет достигнут только в 6% случаев, что может оказаться слишком низкой величиной, по сравнению с уровнем издержек. Вместо этого компания хотела бы продавать страховку только тем клиентам, которые купят ее с высокой вероятностью. В связи с этим общая частота ошибок не представляет интереса. Вместо этого интерес представляет доля клиентов, для которых покупка страховки предсказана верно.

Оказывается, что в случае с клиентами, для которых классификатор предсказывает покупку страховки, KNN с $K = 1$ работает гораздо лучше случайного угадывания. Среди 77 таких клиентов 9 (11.7%) действительно покупают страховку. Это в два раза лучше результата, который был бы получен при простом угадывании.

```
> table(knn.pred, test.Y)
         test.Y
knn.pred  No  Yes
     No  873   50
     Yes  68    9
> 9/(68 + 9)
[1] 0.117
```

При использовании $K = 3$ частота правильных предсказаний повышается до 19%, а при $K = 5$ она составляет 26.7%. Это более чем в четыре раза превышает частоту правильных предсказаний при случайном угады-

вании. Похоже, что KNN находит какие–то реальные закономерности в довольно сложном наборе данных!

```
> knn.pred = knn(train.X, test.X, train.Y, k = 3)
> table(knn.pred, test.Y)
          test.Y
knn.pred  No  Yes
     No  920   54
     Yes  21    5
> 5/26
[1] 0.192
> knn.pred = knn(train.X, test.X, train.Y, k = 5)
> table(knn.pred, test.Y)
          test.Y
knn.pred  No  Yes
     No  930   55
     Yes  11    4
> 4/15
[1] 0.267
```

Для сравнения мы можем также подогнать к этим данным логистическую регрессионную модель. Если мы применим 0.5 в качестве порогового значения для предсказанной классификатором вероятности, то столкнемся с проблемой: покупка страховки будет предсказана только для семи контрольных наблюдений. Хуже того, мы ошибемся во всех этих случаях! Однако нам нет необходимости использовать пороговое значение 0.5. Если мы вместо этого будем предсказывать покупку страховки каждый раз, когда предсказанная вероятность превышает 0.25, то результаты окажутся намного лучше: мы предскажем покупку страховки для 33 человек и будем правы в 33% случаев. Это более чем в пять раз лучше случайного угадывания!

```
> glm.fit = glm(Purchase ~ ., data = Caravan,
                family = binomial, subset = -test)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
glm.fit: fitted probabilities numerically 0 or 1 occurred

> glm.probs = predict(glm.fit, Caravan[test, ],
                      type ="response")
> glm.pred = rep("No", 1000)
> glm.pred[glm.probs > .5] = "Yes"
> table(glm.pred, test.Y)
          test.Y
glm.pred  No Yes
      No 934  59
      Yes  7   0
> glm.pred = rep("No", 1000)
> glm.pred[glm.probs > .25] = "Yes"
> table(glm.pred, test.Y)
          test.Y
glm.pred  No Yes
```

```
      No 919  48
      Yes 22  11
> 11/(22 + 11)
[1] 0.333
```

4.7 Упражнения

Теоретические

1. Применив немного алгебры, докажите, что выражение (4.2) эквивалентно (4.3). Другими словами, покажите, что представление логистической регрессионной модели в виде логистической функции эквивалентно логит–представлению.

2. В тексте было сделано утверждение о том, что отнесение некоторого наблюдения к классу с максимальной величиной (4.2) эквивалентно отнесению этого наблюдения к классу с максимальной величиной (4.13). Докажите, что это действительно так. Иными словами, покажите, что если некоторое наблюдение из класса k происходит из нормального распределения $N(\mu_k, \sigma^2)$, то байесовский классификатор относит это наблюдение к классу, у которого дискриминтантная функция максимальна.

3. Эта задача касается QDA–модели, в которой наблюдения в каждом классе имеют нормальное распределение со специфичным для конкретного класса вектором математического ожидания и специфичной ковариационной матрицей. Мы рассмотрим простой случай с $p = 1$, т. е. только с одной независимой переменной. Предположим, что у нас есть K классов и что если некоторое наблюдение принадлежит к классу k, то X имеет одномерное нормальное распределение: $X \sim N(\mu_k, \sigma_k^2)$. Напомним, что функция плотности одномерного нормального распределения приведена в (4.11). Докажите, что в данном случае байесовский классификатор *не* является линейным. Покажите, что в действительности он является квадратичным.

   Подсказка: для решения этой задачи вам следует воспользоваться аргументами, изложенными в подразделе 4.4.2, но не делать предположения о том, что $\sigma_1^2 = \cdots = \sigma_K^2$.

4. При большом количестве предикторов p обычно наблюдается снижение качества KNN и других *локальных* методов, которые делают предсказания на основе только нескольких наблюдений, находящихся *вблизи* от интересующего нас контрольного наблюдения. Это явление известно как *проклятие размерности* и объясняет тот факт, что при большом p непараметрические методы часто работают неудовлетворительно. Сейчас мы познакомимся с этим проклятием подробнее.

проклятие размерности

   (a) Пусть у нас есть набор наблюдений, для каждого из которых измерен $p = 1$ признак X. Мы предполагаем, что признак X равномерно распределен на отрезке $[0, 1]$. С каждым наблюдением связано некоторое значение отклика. Предположим, что мы хотим

предсказать отклик для некоторого контрольного наблюдения на основе только тех близлежащих точек, которые находятся в пределах 10% от значения X у этого наблюдения. Например, чтобы предсказать отклик для контрольного наблюдения с $X = 0.6$ мы будем использовать наблюдения из диапазона $[0.55, 0.65]$. Какую долю имеющихся наблюдений (в среднем) мы будем использовать для получения предсказания?

(b) Теперь допустим, что у нас есть некоторый набор наблюдений, для каждого из которых измерены $p = 2$ признака — X_1 и X_2. Мы предполагаем, что наблюдения (X_1, X_2) равномерно распределены в диапазоне $[0, 1] \times [0, 1]$. Мы хотим предсказать отклик для некоторого контрольного наблюдения, используя только те близлежащие точки, которые находятся в пределах 10% от значения X_1 *и* в пределах 10% от значения X_2 у этого наблюдения. Например, чтобы предсказать отклик для контрольного наблюдения с $X_1 = 0.6$ и $X_2 = 0.35$ мы будем использовать наблюдения со значениями X_1 из диапазона $[0.55, 0.65]$ и значениями X_2 из диапазона $[0.3, 0.4]$. Какую долю имеющихся наблюдений (в среднем) мы будем использовать для получения предсказания?

(c) Теперь представьте, что у нас есть набор наблюдений с $p = 100$ признаками. Как и раньше, наблюдения равномерно распределены по каждому признаку и каждый признак изменяется от 0 до 1. Мы хотим предсказать отклик для контрольного наблюдения, используя те ближайшие к нему точки, которые находятся в пределах 10% от значений каждого признака у этого наблюдения. Какую долю имеющихся наблюдений мы будем использовать для получения предсказания?

(d) На основании своих ответов для (a)–(c) покажите, что недостаток KNN при большом p состоит в очень малом количестве обучающих наблюдений, находящихся «рядом» с любым контрольным наблюдением.

(e) Теперь представьте, что мы хотим сделать предсказание для некоторого контрольного наблюдения путем создания p–мерного гиперкуба, центрированного по этому наблюдению и содержащего в среднем 10% обучающих наблюдений. Какова длина каждого ребра такого гиперкуба при $p = 1, 2$ и 100? Прокомментируйте свой ответ.

Примечание: гиперкуб — это обобщение куба на пространство произвольной размерности. При $p = 1$ гиперкуб является просто отрезком, при $p = 2$ — это квадрат, а при $p = 100$ — это p–мерный куб.

5. Теперь мы рассмотрим различия между LDA и QDA.

(a) Если байесовская решающая граница является линейной, у какого метода мы ожидаем более высокое качество предсказаний на обучающих данных — LDA или QDA? А на контрольных данных?

(b) Если байесовская решающая граница является нелинейной, у какого метода мы ожидаем более высокое качество предсказаний на обучающих данных — LDA или QDA? А на контрольных данных?

(c) Ожидаем ли мы, что в целом по мере увеличения объема выборки точность предсказаний QDA в сравнении с LDA возрастет? А может, она снизится или останется неизменной? Почему?

(d) Верно ли следующее утверждение: даже если байесовская решающая граница для некоторой задачи является линейной, при использовании QDA мы, возможно, получим более точные предсказания на контрольных данных, чем при использовании LDA, поскольку QDA является достаточно гибким методом для моделирования линейной решающей границы. Обоснуйте свой ответ.

6. Представьте, что мы собрали данные для группы студентов, посещающих курс по статистике, с переменными $X_1 =$ количество часов, посвященных изучению предмета, $X_2 =$ средний балл в школе и $Y =$ индикатор получения самой высокой отметки за экзамен. Мы подогнали логистическую регрессию и получили оценки коэффициентов $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$.

   (a) Рассчитайте вероятность того, что студент, затративший на обучение 40 часов и имеющий средний школьный балл 3.5, покажет наилучший результат на экзамене.

   (b) Сколько часов студенту из (a) необходимо затратить на обучение, чтобы с 50%–ной вероятностью получить наивысшую отметку за экзамен?

7. Представим, что мы хотим предсказать выплату дивидендов по некоторым акциям в этом году («да» или «нет») на основе X — прибыли, выраженной в процентах от объема продаж. Мы изучаем большое количество компаний и выясняем, что среднее значение X для компаний, выплативших дивиденды, составило $\bar{X} = 10$, тогда как для компаний, которые дивиденды не выплатили, оно составило $\bar{X} = 0$. Кроме того, дисперсия X у этих двух групп компаний составила $\hat{\sigma}^2 = 36$. Наконец, дивиденды были выплачены 80% компаний. Допустив, что X имеет нормальное распределение, рассчитайте вероятность выплаты дивидендов некоторой компанией в этом году при условии, что в прошлом году прибыль, выраженная в процентах от объема продаж, составила $X = 4$.

   Подсказка: вспомните, что функция плотности вероятности случайной нормально распределенной переменной выражается как $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$. *Вам придется воспользоваться теоремой Байеса.*

8. Представьте, что мы берем некоторый набор данных, разбиваем его на обучающую и контрольную выборки одинакового размера и затем проверяем две разные процедуры классификации. Сначала мы используем логистическую регрессию и получаем частоту ошибок 20%

на обучающих данных и 30% на контрольных данных. Далее мы применяем метод K ближайших соседей с $K = 1$ и получаем среднюю частоту ошибок 18% (усредненную по обучающим и контрольным данным). Судя по этим результатам, какой метод нам следует предпочесть для классификации новых наблюдений? Почему?

9. Эта задача касается *риска* наступления события.

   (a) В среднем какая доля людей с риском неуплаты долга по кредитной карте, равным 0.37, в действительности станут неплательщиками?

   (b) Представьте, что у некоторого клиента вероятность неуплаты долга по кредитной карте составляет 16%. Чему у этого клиента равен риск неуплаты долга?

Практические

10. На этот вопрос необходимо ответить, используя набор данных `Weekly` из пакета `ISLR`. По своей сути этот набор данных похож на данные `Smarket` из лабораторной работы для предыдущей главы, за тем исключением, что он содержит 1089 значений недельной доходности акций за 21 год — с начала 1990 г. до конца 2010 г.

    (a) Рассчитайте описательные статистики и постройте графики для переменных из таблицы `Weekly`. Видны ли какие–либо закономерности?

    (b) Используйте весь набор данных для подгонки логистической регрессии с `Direction` в качестве отклика и пятью лаг–переменными и `Volume` в качестве предикторов. Примените функцию `summary()` для вывода результатов. Можно ли сделать заключение, что некоторые из предикторов являются статистически значимыми? Если да, то какие?

    (c) Рассчитайте матрицу неточностей и общую долю правильно классифицированных случаев. Объясните, что говорит матрица неточностей о разных типах ошибок, сделанных логистической регрессией.

    (d) Теперь постройте логистическую регрессию на основе обучающих данных, охватывающих период с 1990 по 2008 гг., с `Lag2` в качестве единственного предиктора. Рассчитайте матрицу неточностей и общую долю правильно классифицированных случаев для контрольной выборки (т. е. данных за 2009–2010 г.).

    (e) Повторите (d) с использованием LDA.

    (f) Повторите (d) с использованием QDA.

    (g) Повторите (d) с использованием KNN с $K = 1$.

    (h) Какой из методов обеспечивает наиболее точные предсказания для этих данных?

(i) Поэкспериментируйте с разными комбинациями предикторов (включая возможные трансформации и взаимодействия) для каждого метода. Составьте отчет с указанием предикторов, метода и матрицы неточностей, которые соответствуют наилучшим результатам на контрольной выборке. Обратите внимание, что вам также следует поэкспериментировать со значениями K для KNN–классификатора.

11. В этой задаче вы разработаете модель на основе данных `Auto` для предсказания того, является ли расход топлива у автомобиля высоким или низким.

    (a) Создайте бинарную переменную `mpg01`, которая равна 1, если переменная `mpg` превышает собственную медиану, и 0, если `mpg` меньше собственной медианы. Вы можете вычислить медиану при помощи функции `median()`. Обратите внимание: полезной может оказаться функция `data.frame()`, которая поможет создать одну таблицу данных, включающую как `mpg01`, так и другие переменные из таблицы `Auto`.

    (b) Исследуйте эти данные графически для установления связи между `mpg01` и другими переменными. Какие из этих переменных выглядят наиболее подходящими для предсказания `mpg01`? Для ответа на этот вопрос полезными могут оказаться диаграммы рассеяния и диаграммы размахов. Опишите свои находки.

    (c) Разбейте данные на обучающую и контрольные выборки.

    (d) Постройте LDA–модель по обучающим данным для предсказания `mpg01` на основе переменных, которые выглядели наиболее тесно связанными с `mpg01` в (b). Чему равна частота ошибок на контрольной выборке?

    (e) Постройте QDA–модель по обучающим данным для предсказания `mpg01` на основе переменных, которые выглядели наиболее тесно связанными с `mpg01` в (b). Чему равна частота ошибок на контрольной выборке?

    (f) Постройте логистическую регрессию по обучающим данным для предсказания `mpg01` на основе переменных, которые выглядели наиболее тесно связанными с `mpg01` в (b). Чему равна частота ошибок на контрольной выборке?

    (g) Постройте KNN–модель по обучающим данным для предсказания `mpg01` (с несколькими различающимися значениями K). Используйте только те переменные, которые выглядели наиболее тесно связанными с `mpg01` в (b). Чему равна частота ошибок на контрольной выборке? Какое значение K выглядит наиболее подходящим для этих данных?

12. Эта задача предполагает написание функций.

    (a) Напишите функцию `Power()`, которая выводит результат возведения 2 в 3–ю степень. Другими словами, ваша функция должна вычислять 2^3 и выводить результат на экран.

Подсказка: вспомните, что `x^a` *возводит* `x` *в степень* `a`*. Для вывода результата используйте функцию* `print()`.

(b) Создайте новую функцию — `Power2()`, которая позволяет подать на нее *любые* два числа — `x` и `a` — и вывести значение `x^a`. Вы можете сделать это, начав свою функцию со строки

```
> Power2 = function(x, a) {
```

Вы должны иметь возможность вызвать свою функцию путем набора в командной строке, например следующего:

```
> Power2(3, 8)
```

Эта команда должна вывести значение 3^8, т. е. 6561.

(c) Используя написанную вами только что функцию `Power2()`, вычислите 10^3, 8^{17} и 131^3.

(d) Теперь создайте функцию `Power3()`, которая возвращает результат `x^2` в виде объекта `R`, а не просто выводит результат вычислений на экран. Другими словами, если вы сохраните значение `x^2` в объекте под названием `result`, то вы сможете получить этот объект при помощи функции `return()`, используя следующую строку:

```
> return(result)
```

Эта строка должна быть последней в вашей функции и располагаться перед символом `}`.

(e) Теперь, используя функцию `Power3()`, создайте график $f(x) = x^2$. Ось x должна показывать диапазон целых чисел от 1 до 10, а ось y — x^2. Дайте подходящие названия осям и подберите соответствующий заголовок для графика. Подумайте о том, чтобы представить ось x или ось y на логарифмической шкале. Вы можете сделать это при помощи аргументов `log = "x"`, `log = "y"` или `log = "xy"` функции `plot()`.

(f) Создайте функцию `PlotPower()`, которая позволяет вам построить график зависимости `x^a` от `x` для некоторого фиксированного `a` и определенного диапазона значений `x`. Например, если вы выполните

```
> PlotPower(1:10, 3)
```

то должен появиться график, у которого по оси x представлены значения $1, 2, \ldots, 10$, а по оси y — $1^3, 2^3, \ldots, 10^3$.

13. Используя набор данных `Boston`, постройте классификаторы для предсказания того, превышает ли уровень преступности в некотором пригороде медианное значение этой переменной. Попробуйте модели логистической регрессии, LDA и KNN с разными наборами предикторов. Опишите свои результаты.

Глава 5

Методы создания повторных выборок

Методы создания повторных выборок[1] являются незаменимым инструментом в современной статистике. Они подразумевают многократное извлечение выборок из обучающих данных и подгонку некоторой интересующей нас модели по каждой из этих выборок для получения дополнительной информации об этой модели. Например, для оценивания дисперсии линейной модели мы можем извлечь несколько различных выборок из обучающих данных, подогнать линейную регрессию по каждой новой выборке, а затем исследовать степень различий получающихся моделей. Такой подход может дать нам информацию, которая была бы недоступной при подгонке только одной модели на основе исходной обучающей выборки.

Методы создания повторных выборок могут потребовать больших вычислительных ресурсов, поскольку они включают многократную подгонку одной и той же статистической модели к разным подмножествам обучающих данных. Однако благодаря недавним достижениям в области вычислительной техники ресурсоемкость этих методов обычно не является запретительной. В этой главе мы обсуждаем два наиболее часто используемых метода создания повторных выборок — *перекрестную проверку* и *бутстреп*. Оба метода являются важными инструментами в практическом применении многих методов статистического обучения. Например, перекрестную проверку можно использовать для оценивания ошибки на контрольной выборке, чтобы оценить качество того или иного метода статистического обучения или найти подходящий для этого метода уровень гибкости. Процесс определения качества предсказаний модели известен как *оценка модели*, тогда как процесс нахождения подходящего уровня гибкости известен как *отбор модели*[2]. Бутстреп применяется в разных контекстах, но чаще всего — для получения меры точности оценки некоторого параметра или метода статистического обучения.

оценка модели

отбор модели

[1] В оригинале используется термин «resampling methods». — *Прим. пер.*

[2] В оригинале используются термины «model assessment» и «model selection» соответственно. — *Прим. пер.*

5.1 Перекрестная проверка

В главе 2 мы обсуждали различие между *частотой ошибок на контрольной выборке* и *частотой ошибок на обучающей выборке*. Частота ошибок на контрольной выборке представляет собой среднюю ошибку, которая возникает в результате применения некоторого метода статистического обучения для предсказания отклика по новому наблюдению, т. е. измерению, которое не было использовано при обучении модели. Применение того или иного метода статистического обучения для определенного набора данных является оправданным, если этот метод обеспечивает низкую ошибку на контрольной выборке. Ошибку на контрольной выборке можно легко вычислить при наличии предназначенного для этого контрольного набора данных. К сожалению, обычно такой набор данных отсутствует. В то же время ошибку на обучающей выборке можно легко рассчитать путем применения некоторой модели непосредственно к наблюдениям, по которым она была обучена. Однако, как мы видели в главе 2, частота ошибок на обучающей выборке часто значительно отличается от ошибки на контрольной выборке; в частности, частота ошибок на обучающих данных может существенно недооценивать частоту ошибок на контрольных данных.

В отсутствие очень большой и специально созданной контрольной выборки, непосредственно подходящей для оценивания частоты ошибок, можно применить ряд методов для нахождения подобной оценки по имеющимся обучающим данным. Некоторые из этих методов оценивают частоту ошибок на контрольной выборке за счет определенной математической поправки, применяемой к частоте ошибок на обучающей выборке. Такие подходы обсуждаются в главе 6. В этом же разделе мы рассматриваем класс методов, которые оценивают частоту ошибки на контрольной выборке путем *исключения* части обучающих наблюдений из процесса подгонки модели и последующего применения этой модели к исключенным наблюдениям.

В подразделах 5.1.1–5.1.4 мы для простоты предполагаем, что интерес представляет построение регрессионной модели с количественным откликом. В подразделе 5.1.5 мы рассматриваем случай классификации с качественным откликом. Как мы вскоре увидим, ключевая идея остается неизменной вне зависимости от того, является отклик количественными или качественным.

5.1.1 Метод проверочной выборки

Предположим, что мы хотели бы оценить ошибку на контрольной выборке для некоторой статистической модели, подогнанной по некоторому набору наблюдений. Показанный на рис. 5.1 *метод проверочной выборки*[3] представляет собой очень простую стратегию для решения этой задачи. Он заключается в случайном разбиении имеющегося набора наблюдений на две части — *обучающую выборку* и *проверочную*, или *удержанную*, *выборку*. Модель строится по обучающей выборке, а затем применяется для предсказания отклика у наблюдений из проверочной выборки. Получаемая в итоге ошибка на проверочной выборке (обычно для количественного

метод проверочной выборки

проверочная выборка

[3] В оригинале используется термин «the validation set approach». — *Прим. пер.*

отклика она выражается при помощи MSE) дает оценку ошибки на независимых данных.

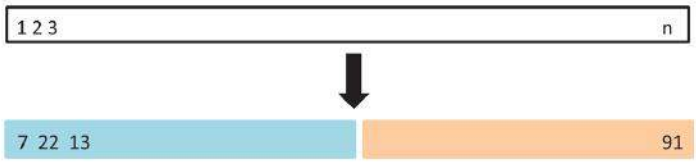

РИСУНОК 5.1. *Схематичная иллюстрация метода проверочной выборки. Набор из n наблюдений случайным образом разбивается на обучающую выборку (показана голубым цветом и содержит наблюдения 7, 22 и 13) и проверочную выборку (показана бежевым цветом и, помимо других наблюдений, содержит наблюдение 91). Статистическая модель подгоняется на обучающей выборке, а ее качество оценивается на проверочной выборке*

Мы продемонстрируем метод проверочной выборки на примере набора данных `Auto`. Вспомните из главы 3, что `mpg` и `horsepower` связаны нелинейной зависимостью и что модель, предсказывающая `mpg` на основе `horsepower` и $\texttt{horsepower}^2$, обладает более высоким качеством, чем модель, содержащая только линейный член. Естественным является вопрос о том, мог ли полином третьей или более высокой степени привести к дальнейшему улучшению качества модели. В главе 3 мы отвечаем на этот вопрос, исследуя p–значения, связанные в линейной регрессии с полиномиальными членами 3–й и более высокой степени. Однако мы могли бы ответить на этот вопрос и при помощи метода проверочной выборки. Для этого мы случайным образом разбиваем 392 наблюдения на две части — обучающую выборку, содержащую 196 наблюдений, и проверочную выборку, содержащую остальные 196 наблюдений. Среднеквадратичные ошибки (MSE), полученные в результате подгонки разных регрессионных моделей по обучающей выборке и последующего оценивания качества этих моделей на проверочной выборке, показаны слева на рис. 5.2. У квадратичной модели MSE на проверочной выборке значительно ниже, чем у линейной модели. Однако у кубической модели MSE на проверочной выборке несколько выше, чем у квадратичной модели. Таким образом, добавление кубического члена в регрессионную модель не приводит к улучшению качества предсказаний, по сравнению с использованием только квадратичного члена.

Вспомните, что для создания графика, приведенного на рис. 5.2 слева, мы случайным образом разбили исходный набор данных на две части — обучающую и проверочную. Если мы повторим процесс случайного разбиения данных на две части, то получим несколько отличающуюся оценку MSE на контрольной выборке. В качестве иллюстрации справа на рис. 5.2 для набора данных `Auto` показаны десять разных кривых MSE на проверочной выборке, полученных с использованием десяти различных разбиений на обучающую и проверочную выборки. Все десять кривых указывают на то, что модель с квадратичным членом имеет существенно меньшую MSE на проверочной выборке, чем модель, содержащая только линейный

член. Более того, все десять кривых указывают на то, что добавление в модель кубического члена или членов более высокой степени особой пользы не приносит. Однако стоит отметить, что каждая из этих десяти кривых дает свою оценку MSE на контрольной выборке для каждой из десяти рассмотренных регрессионных моделей. При этом между кривыми нет согласия в том, какая модель приводит к наименьшей MSE на проверочной выборке. Учитывая эти различия между кривыми, мы можем быть уверены лишь в том, что линейная модель для этих данных не подходит.

Метод проверочной выборки концептуально прост, и его несложно реализовать. Однако у него есть два потенциальных недостатка:

1. Как показано справа на рис. 5.2, оценка ошибки на проверочной выборке может оказаться очень вариабельной, что определяется тем, какие наблюдения попадают в обучающую выборку, а какие — в проверочную.

2. В ходе реализации метода проверочной выборки в подгонке модели участвует только часть наблюдений — те наблюдения, которые попадают в обучающую выборку. Поскольку при обучении на небольших выборках качество получаемых статистических моделей обычно невелико, это предполагает, что ошибка на проверочной выборке может *переоценивать* истинную ошибку модели, обученной по всему набору данных.

В следующих подразделах мы опишем *перекрестную проверку* — улучшенную версию метода проверочной выборки, которая решает эти две проблемы.

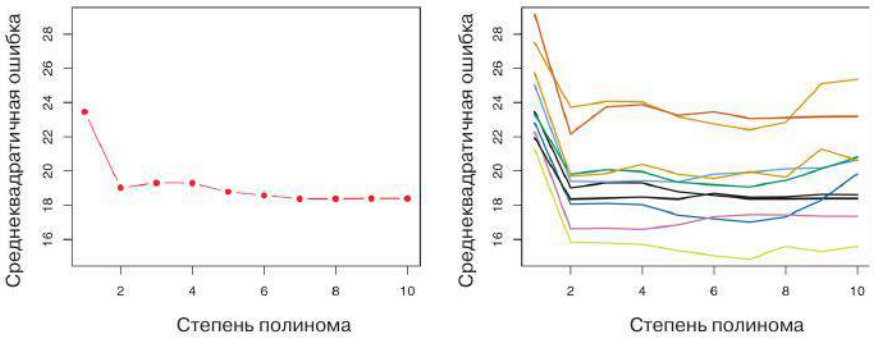

РИСУНОК 5.2. *Метод проверочной выборки был применен к набору данных* `Auto` *для оценивания ошибки прогнозирования* `mpg` *на основе полиномиальных функций* `horsepower`. Слева: *ошибки на проверочной выборке при однократном разбиении на обучающую и проверочную выборки.* Справа: *метод проверочной выборки был применен десять раз, каждый раз с использованием нового разбиения наблюдений на обучающую и проверочную выборки. Этот график иллюстрирует вариабельность оценок MSE на контрольной выборке, полученных при помощи данного метода*

5.1.2 Перекрестная проверка по отдельным наблюдениям

LOOCV

Перекрестная проверка по отдельным наблюдениям (LOOCV)[4] тесно связана с описанным в подразделе 5.1.1 методом проверочной выборки, однако пытается преодолеть недостатки последнего.

Подобно методу проверочной выборки, в ходе LOOCV исходные наблюдения разбиваются на две части. Однако вместо создания двух подмножеств сравнимого размера одно наблюдение — (x_1, y_1) — используется в качестве контрольного, а остальные наблюдения — $\{(x_2, y_2), \ldots, (x_n, y_n)\}$ — составляют обучающую выборку. Статистическая модель подгоняется по $n-1$ обучающим наблюдениям, а предсказание $\hat{y}_1$ производится для исключенного наблюдения по его значению x_1. Поскольку наблюдение (x_1, y_1) не было использовано в процессе обучения, то $\mathrm{MSE}_1 = (y_1 - \hat{y}_1)^2$ обеспечивает приблизительно несмещенную оценку ошибки на контрольной выборке. Однако, несмотря на то что MSE_1 является несмещенной оценкой истинной ошибки на контрольной выборке, она служит плохой оценкой, поскольку, будучи рассчитанной по единственному наблюдению (x_1, y_1), она обладает высокой дисперсией.

Мы можем продолжить эту процедуру, выбрав (x_2, y_2) в качестве контрольного наблюдения, обучив статистическую модель по $n-1$ наблюдениям $\{(x_1, y_1), (x_3, y_3), \ldots, (x_n, y_n)\}$ и рассчитав $\mathrm{MSE}_2 = (y_2 - \hat{y}_2)^2$. Повторение этой процедуры n раз приводит к получению n квадратов остатков: $\mathrm{MSE}_1, \ldots, \mathrm{MSE}_n$. LOOCV–оценка MSE есть среднее значение из этих n оценок ошибки:

$$\mathrm{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{MSE}_i. \tag{5.1}$$

Схематичное изображение метода LOOCV приведено на рис. 5.3.

Метод LOOCV обладает двумя существенными преимуществами, по сравнению с методом проверочной выборки. Во–первых, он обладает меньшим смещением. При реализации LOOCV мы многократно подгоняем статистическую модель с использованием $n-1$ обучающих наблюдений — это почти столько же, сколько содержится в составе всего набора данных. В этом состоит отличие от метода проверочной выборки, где размер обучающей выборки обычно составляет примерно половину от размера исходного набора данных. Следовательно, метод LOOCV обычно не переоценивает частоту ошибок на контрольной выборке в той мере, как это делает метод проверочной выборки. Во–вторых, в отличие от многократного применения метода проверочной выборки, которое дает разные результаты из–за случайного разбиения на обучающую и проверочную выборки, многократное выполнение LOOCV всегда даст одинаковые результаты: случайность при разбиении на обучающую и проверочную выборки попросту отсутствует.

Мы применили LOOCV к данным `Auto` с целью нахождения MSE на контрольной выборке для линейной регрессионной модели, которая пред-

[4] Аббревиатура от «leave–one–out cross–validation». В русскоязычных источниках используется также термин «контроль по отдельным объектам». — *Прим. пер.*

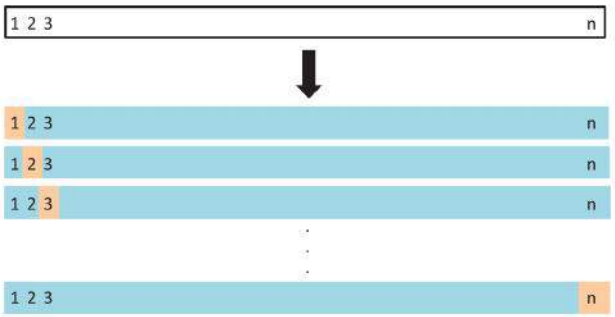

РИСУНОК 5.3. *Схематичное изображение метода LOOCV. Набор из n наблюдений многократно разбивается на обучающую выборку (показана голубым цветом), содержащую все наблюдения, кроме одного, и проверочную выборку, которая содержит только одно это наблюдение (показана бежевым цветом). Ошибка на контрольном множестве затем оценивается путем усреднения n получающихся MSE. Первое обучающее подмножество содержит все наблюдения, за исключением первого, второе обучающее подмножество содержит все наблюдения, за исключением второго, и т. д.*

сказывает `mpg` на основе полиномиальных функций `horsepower`. Результаты этого анализа показаны слева на рис. 5.4.

Реализация LOOCV потенциально может оказаться ресурсоемкой, поскольку модель необходимо подогнать n раз. При большом n, а также при длительной подгонке каждой отдельной модели это может занять очень много времени. Для линейной или полиномиальной регрессии, рассчитываемой по методу наименьших квадратов, имеется удивительный экономный способ, который позволяет сравнять затраты ресурсов на реализацию LOOCV с затратами на подгонку единственной модели. Применяется следующая формула:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2, \tag{5.2}$$

где $\hat{y}_i$ — это i–е модельное значение отклика, а h_i — показатель разбалансировки, определение которого дано в (3.37) на стр. 112. Эта формула похожа на обычную формулу MSE, за тем исключением, что i–й остаток делится на $1 - h_i$. Показатель разбалансировки изменяется от $1/n$ до 1 и отражает степень влияния того или иного наблюдения на его модельное значение. Следовательно, для соблюдения приведенного равенства остатки наблюдений с высоким показателем разбалансировки увеличиваются пропорционально степени влияния этих наблюдений.

LOOCV — это очень общий метод, применимый для любых предсказательных моделей. Например, мы могли бы применить его для логистической регрессии или дискриминантного анализа, или любых других методов, описанных в предыдущих главах. Волшебная формула (5.2) не является универсальной — в случаях, когда она не работает, модель должна подгоняться повторно n раз.

5.1.3 k–кратная перекрестная проверка

k–кратная перекрестная проверка

Альтернативой LOOCV является *k–кратная перекрестная проверка*[5]. Этот метод подразумевает случайное разбиение совокупности наблюдений на k групп, или *блоков*, примерно одинакового размера. Первый блок служит в качестве проверочной выборки, а модель подгоняется по остальным $k-1$ блокам. Среднеквадратичная ошибка MSE_1 рассчитывается далее по наблюдениям из удержанного блока. Эта процедура повторяется k раз: каждый раз в качестве проверочной выборки используется другой блок. В результате этого процесса получают k оценок ошибки на контрольном множестве — $\mathrm{MSE}_1, \mathrm{MSE}_2, \ldots, \mathrm{MSE}_k$. Итоговая оценка рассчитывается путем усреднения этих значений:

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i. \tag{5.3}$$

Иллюстрация метода k–кратной проверки приведена на рис. 5.5.

Нетрудно увидеть, что LOOCV является частным случаем k–кратной перекрестной проверки, где k равно n. На практике k–кратная перекрестная проверка обычно выполняется с использованием $k=5$ или $k=10$. Почему $k=5$ или $k=10$ лучше, чем $k=n$? Наиболее очевидное преимущество заключается в экономии вычислительных ресурсов. LOOCV требует подгонки статистической модели n раз. Это потенциально может оказаться очень ресурсоемким процессом (за исключением линейных моделей, подгоняемых по методу наименьших квадратов, для которых можно применить формулу (5.2)). Однако перекрестная проверка — очень общий подход, применимый почти для любого метода статистического обучения. Процедуры подгонки некоторых статистических моделей требуют больших объемов вычислений, в связи с чем LOOCV может вызвать вычислительные проблемы, особенно при очень больших n. В то же время для выполнения 10–кратной перекрестной проверки подгонку статистической модели необходимо выполнить только десять раз, что может оказаться намного более выполнимым. Как мы увидим в подразделе 5.1.4, могут быть и другие, не имеющие отношения к ресурсоемкости, преимущества использования 5– или 10–кратной перекрестной проверки — они имеют отношение к компромиссу между смещением и дисперсией.

Справа на рис. 5.4 показаны девять различных оценок среднеквадратичной ошибки для набора данных `Auto`, каждая из которых была получена в результате случайного разбиения исходных наблюдений на 10 частей. Как видно по этому рисунку, из–за случайного разбиения наблюдений на десять блоков имеет место некоторая вариабельность оценок. Однако эта вариабельность обычно намного ниже вариабельности оценок ошибки на контрольной выборке, получаемых при помощи метода проверочной выборки (см. график, представленный справа на рис. 5.2).

При работе с реальными данными *истинная* ошибка предсказаний нам не известна, что затрудняет определение точности оценки, получаемой при помощи перекрестной проверки. Однако, используя имитированные данные, мы можем вычислить истинную ошибку и таким образом оценить верность результатов перекрестной проверки. На рис. 5.6 мы изобразили

[5] В оригинале используется термин «k–fold cross–validation». — *Прим. пер.*

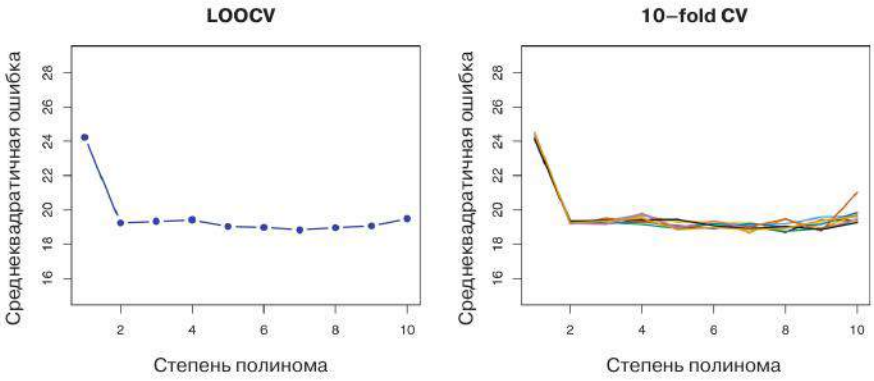

РИСУНОК 5.4. *Перекрестная проверка была применена к набору данных* `Auto` *для нахождения ошибки предсказания* `mpg` *на основе полиномиальных функций* `horsepower`. Слева: *кривая ошибок LOOCV*. Справа: *10-кратная перекрестная проверка была повторена девять раз, каждый раз с использованием нового разбиения наблюдений на десять частей. График показывает девять слабо различающихся кривых ошибок, полученных с помощью перекрестной проверки*

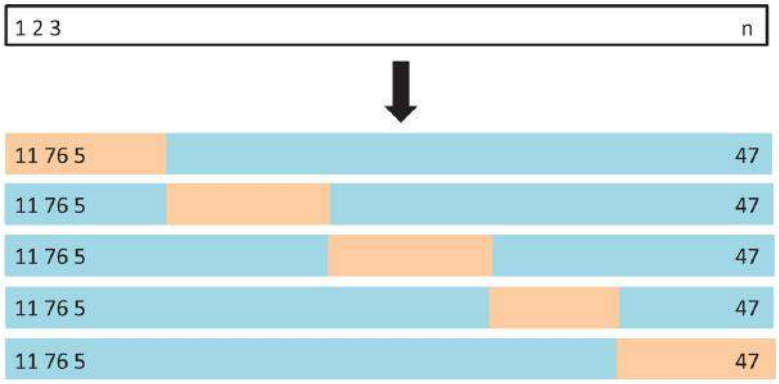

РИСУНОК 5.5. *Схематичное изображение 5-кратной перекрестной проверки. Набор из n наблюдений случайным образом разбивается на пять неперекрывающихся групп. Каждая из этих пяти групп последовательно выступает в качестве проверочной выборки (обозначены бежевым цветом), а остальные группы — в качестве обучающей выборки (показаны голубым цветом). Ошибка предсказаний оценивается путем усреднения пяти итоговых оценок MSE*

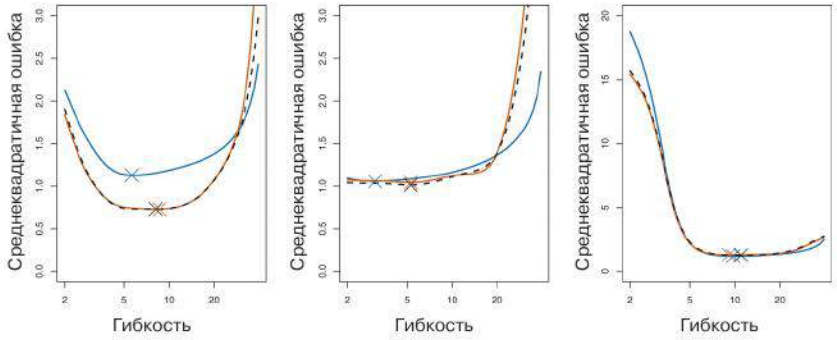

РИСУНОК 5.6. *Истинная и оцененные среднеквадратичные ошибки предсказаний для имитированных данных, показанных на рис. 2.9 (слева), 2.10 (в центре) и 2.11 (справа). Истинная MSE показана голубым цветом, LOOCV–оценки — в виде черной прерывистой линии, а оценки 10–кратной перекрестной проверки — в виде сплошной оранжевой линии. Крестиками показаны минимумы каждой кривой*

оценки MSE на контрольной выборке, полученные при помощи перекрестной проверки, и истинные ошибки, получаемые в результате применения сглаживающих сплайнов к имитированным данным из рис. 2.9–2.11 (глава 2). Истинная MSE на контрольной выборке показана голубым цветом. Черные прерывистые и сплошные оранжевые линии показывают оценки, полученные при помощи LOOCV и 10–кратной перекрестной проверки соответственно. На всех трех графиках оценки, полученные при помощи этих двух методов перекрестной проверки, очень похожи. Справа на рис. 5.6 истинная MSE и кривые, соответствующие этим двум методам перекрестной проверки, почти идентичны. В центре на рис. 5.6 все кривые очень похожи в области низких значений гибкости, однако при высоких уровнях гибкости методы перекрестной проверки переоценивают истинную ошибку предсказаний. Слева на рис. 5.6 форма кривых, соответствующих методам перекрестной проверки, в целом имеет правильную форму, однако оба метода недооценивают истинную ошибку на контрольной выборке.

Выполняя перекрестную проверку, мы стремимся определить качество предсказаний той или иной статистической модели на независимых данных; в этом случае нам интересна сама оценка MSE на контрольной выборке. Однако в других случаях нам будет интересно только *положение минимума на кривой ошибок*. Дело в том, что мы можем выполнять перекрестную проверку для нескольких методов статистического обучения или для одного метода с использованием разных уровней его гибкости с целью найти метод, обеспечивающий наименьшую ошибку на контрольной выборке. Здесь важно положение минимума на кривой оцененных ошибок, а не конкретное значение оценки MSE. Как видно на рис. 5.6, несмотря на то что методы перекрестной проверки иногда занижают истинную MSE на контрольных данных, их кривые довольно хорошо определяют правильный уровень гибкости, т. е. уровень, соответствующий наименьшей ошибке предсказаний.

5.1.4 Компромисс между смещением и дисперсией в контексте k–кратной перекрестной проверки

В подразделе 5.1.3 мы отметили, что k–кратная перекрестная проверка с $k < n$ имеет вычислительные преимущества, по сравнению с LOOCV. Однако помимо вычислительных аспектов, менее очевидным, но потенциально более важным преимуществом k–кратной перекрестной проверки является то, что она дает более точную оценку ошибки на контрольной выборке, по сравнению с LOOCV. Это обусловлено компромиссом между смещением и дисперсией.

Как было отмечено в подразделе 5.1.1, метод проверочной выборки может приводить к завышенным оценкам ошибки на контрольной выборке, поскольку при реализации этого метода обучающее множество, по которому подгоняется статистическая модель, содержит только половину всех наблюдений. Используя ту же логику, не составляет труда увидеть, что LOOCV будет давать примерно несмещенные оценки ошибки на контрольной выборке, поскольку каждое обучающее множество содержит $n - 1$ наблюдений, т. е. почти столько же, сколько есть наблюдений в исходном наборе данных. В свою очередь, выполнение k–кратной перекрестной проверки со значением k, равным, например, 5 или 10, будет давать промежуточный уровень смещения, поскольку каждое обучающее множество содержит $(k - 1)n/k$ наблюдений — меньше, чем в случае с LOOCV, но намного больше, чем в случае с методом проверочной выборки. Следовательно, с точки зрения снижения уровня смещения, LOOCV обладает явным преимуществом в сравнении с k–кратной перекрестной проверкой.

Тем не менее мы знаем, что смещение не является единственным источником беспокойства в отношении той или иной статистической процедуры — мы должны также учесть дисперсию этой процедуры. Оказывается, что LOOCV обладает более высокой дисперсией, чем k–кратная перекрестная проверка с $k < n$. Почему это так? Когда мы выполняем LOOCV, мы, по сути, усредняем результаты подгонки n моделей, каждая из которых обучена на почти одинаковом наборе наблюдений; как следствие эти результаты очень сильно (положительно) коррелируют друг с другом. В то же время при выполнении k–кратной перекрестной проверки с $k < n$ мы усредняем результаты подгонки k моделей, которые коррелируют друг с другом в меньшей степени из–за меньшего сходства между их обучающими выборками. Поскольку среднее значение большого числа коррелирующих величин имеет более высокую дисперсию, чем среднее значение большого числа слабо коррелирующих величин, то LOOCV–оценка ошибки предсказаний обычно обладает большей дисперсией, чем оценка ошибки, полученная при помощи k–кратной перекрестной проверки.

В качестве заключения отметим, что компромисс между смещением и дисперсией имеет место и при выборе k для k–кратной перекрестной проверки. Исходя из приведенных выше рассуждений, обычно выполняют 5– или 10–кратную перекрестную проверку, поскольку было установлено, что эти значения обеспечивают оценки ошибки предсказаний с умеренными уровнями смещения и дисперсии.

5.1.5 Перекрестная проверка при решении задач классификации

До сих пор в этой главе мы обсуждали перекрестную проверку в контексте регрессионных моделей, в которых отклик Y является количественным, и поэтому для количественного описания ошибки на контрольном множестве мы использовали MSE. Однако перекрестная проверка может быть также очень полезной для решения задач классификации, когда Y является качественной переменной. При таком сценарии механизм перекрестной проверки не отличается от описанного ранее в этой главе, за тем исключением, что вместо MSE мы выражаем ошибку на контрольной выборке в виде числа неверно классифицированных объектов. Например, для задач классификации частота ошибок при использовании LOOCV принимает форму

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i, \tag{5.4}$$

где $\mathrm{Err}_i = I(y_i \neq \hat{y}_i)$. Формулы частот ошибок для k–кратной перекрестной проверки и метода проверочной выборки имеют аналогичную форму.

В качестве примера мы построим несколько моделей логистической регрессии по двумерным данным, изображенным на рис. 2.13. Слева вверху на рис. 5.7 черная сплошная линия соответствует решающей границе, оцененной в результате подгонки к этим данным стандартной логистической регрессии. Поскольку это имитированные данные, мы можем вычислить истинную частоту ошибок на контрольной выборке — она равна 0.201 и существенно превышает байесовскую частоту ошибок, равную 0.133. Очевидно, что в этом случае логистическая регрессия не имеет достаточной гибкости для моделирования байесовской решающей границы. Подобно тому, как мы сделали это в подразделе 3.3.2 при рассмотрении линейных регрессионных моделей, для получения нелинейной решающей границы мы можем легко расширить логистическую регрессию при помощи полиномиальных функций предикторов. Например, мы можем подогнать следующую *квадратичную* логистическую модель:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X_2^2. \tag{5.5}$$

Справа вверху на рис. 5.7 показана решающая граница, которая теперь принимает изогнутую форму. Однако частота ошибок улучшилась незначительно — до 0.197. Несколько более выраженное улучшение видно на графике, представленном на рис. 5.7 слева внизу, где мы подогнали логистическую регрессию с кубическими полиномами предикторов. Теперь частота ошибок снизилась до 0.160. Использование полиномов четвертой степени (справа внизу) несколько увеличивает частоту ошибок.

На практике при работе с реальными данными байесовская решающая граница и истинная частота ошибочных предсказаний неизвестны. Какую же из четырех изображенных на рис. 5.7 логистических регрессионных моделей нам выбрать? Для принятия решения можем применить перекрестную проверку. Слева на рис. 5.8 черным цветом показаны частоты

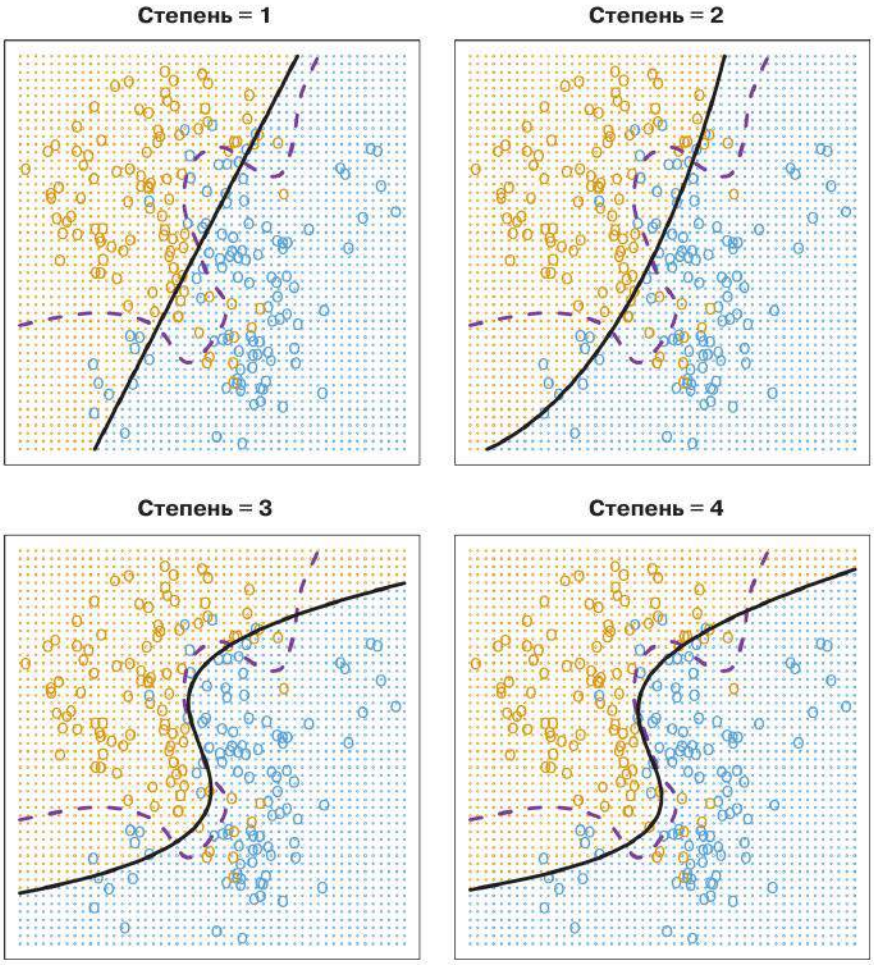

РИСУНОК 5.7. *Логистические регрессионные модели, подогнанные к данным из рис. 2.13. Байесовская решающая граница показана в виде фиолетовой прерывистой линии. Решающие границы, оцененные на основе полиномиальных функций первой, второй, третьей и четвертой степеней, показаны в виде черных линий. Частоты ошибок для этих четырех регрессионных моделей составляют 0.201, 0.197, 0.160 и 0.162 соответственно, тогда как байесовская частота ошибок равна 0.133*

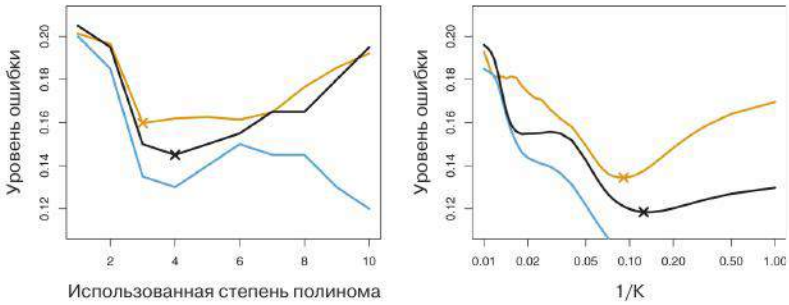

РИСУНОК 5.8. *Ошибка классификации объектов на контрольной выборке (коричневая кривая), на обучающей выборке (голубая кривая) и ошибка, полученная при помощи 10–кратной перекрестной проверки (черная кривая) для двумерных данных рис. 5.7.* Слева: *логистическая регрессия на основе полиномиальных функций предикторов. Степень использованных полиномов указана на оси X.* Справа: *KNN–классификатор с разными значениями K (числом ближайших соседей)*

ошибок, полученные при помощи 10–кратной перекрестной проверки логистических регрессионных моделей с полиномиальными функциями предикторов вплоть до 10–й степени. Истинные частоты ошибок показаны коричневым цветом, а частоты ошибок на обучающих выборках — голубым. Как мы уже видели ранее, при увеличении гибкости модели частота ошибок на обучающей выборке обычно снижается. (На рисунке видно, что хотя частота ошибок на обучающих данных и не снижается монотонно, в целом при увеличении сложности модели она все же имеет тенденцию на снижение.) В то же время частота ошибок на контрольной выборке демонстрирует характерную U–образную форму. Десятикратная перекрестная проверка обеспечивает довольно хорошую аппроксимацию истинной частоты ошибок на контрольной выборке. Несмотря на некоторое занижение частоты ошибок, кривая перекрестной проверки достигает минимума при использовании полиномов четвертой степени, что очень близко к минимуму кривой истинной частоты ошибок на контрольной выборке, который наблюдается при использовании полиномов третьей степени. Более того, применение полиномов четвертой степени могло бы привести к хорошему качеству предсказаний, поскольку истинная частота ошибок на контрольном множестве для полиномов третьей, четвертой, пятой и шестой степеней примерно одинакова.

Справа на рис. 5.8 показаны такие же три кривые для KNN–классификатора с разными значениями K (здесь K означает используемое классификатором количество ближайших соседей, а не количество блоков для перекрестной проверки). Частота ошибок на обучающей выборке снова снижается по мере возрастания гибкости метода, из чего следует, что мы не можем использовать эту частоту ошибок для выбора оптимального значения K. Хотя кривая перекрестной проверки несколько занижает частоту ошибок на контрольной выборке, ее минимум близок к оптимальному значению K.

5.2 Бутстреп

бутстреп

Бутстреп[6] — это широко применимый и чрезвычайно мощный статистический инструмент, который можно использовать для количественного описания неопределенности в отношении оценки некоторого параметра или метода статистического обучения. В качестве простого примера можно привести использование бутстрепа для оценивания стандартных ошибок коэффициентов некоторой линейной модели. В случае с линейной регрессией этот подход не очень полезен, поскольку мы видели в главе 3, что стандартные статистические программы (например, R) выдают такие стандартные ошибки автоматически. Однако сила бутстрепа заключается в том, что его можно легко применить для широкого круга методов статистического обучения, включая такие методы, для которых меру дисперсии сложно вычислить или получить автоматически при помощи статистических программ.

В этом разделе мы описываем бутстреп на гипотетическом примере, в котором мы хотим определить оптимальное размещение инвестиций на основе простой модели. В разделе 5.3 мы рассмотрим применение бутстрепа для описания вариабельности регрессионных коэффициентов линейной модели.

Представьте, что мы хотим инвестировать некоторую фиксированную сумму денег в два финансовых актива, которые обеспечивают доходность в размере X и Y соответственно, где X и Y — это случайные величины. Мы вложим некоторую долю наших денег α в X, а оставшуюся часть $(1-\alpha)$ — в Y. Поскольку имеет место вариабельность доходности по этим двум активам, мы хотим выбрать такую долю α, которая минимизирует общий риск, или вариабельность, нашей инвестиции. Иными словами, мы хотим минимизировать $\mathrm{Var}(\alpha X + (1-\alpha)Y)$. Можно показать, что минимизирующее этот риск значение α вычисляется как

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}, \tag{5.6}$$

где $\sigma_X^2 = \mathrm{Var}(X)$, $\sigma_Y^2 = \mathrm{Var}(Y)$, а $\sigma_{XY} = \mathrm{Cov}(X, Y)$.

В действительности величины σ_X^2, σ_Y^2 и σ_{XY} неизвестны. Мы можем вычислить оценки этих величин $\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$ и $\hat{\sigma}_{XY}$ по данным, которые содержат прошлые измерения X и Y. Затем мы можем оценить значение α, которое минимизирует дисперсию нашей инвестиции, следующим образом:

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}. \tag{5.7}$$

Рисунок 5.9 иллюстрирует этот подход для нахождения α на основе имитированных данных. На каждом графике мы сгенерировали 100 пар значений доходности для инвестиций X и Y. Эти значения доходности были использованы для оценивания σ_X^2, σ_Y^2 и σ_{XY}, которые мы подставили в (5.7) для нахождения α. Итоговое значение $\hat{\alpha}$, полученное на основе каждого имитированного набора данных, изменяется от 0.532 до 0.657.

[6] Устоявшегося перевода «bootstrap» на русский язык не существует. Здесь используется прямая транслитерация этого термина («бутстреп»), чаще всего встречающаяся

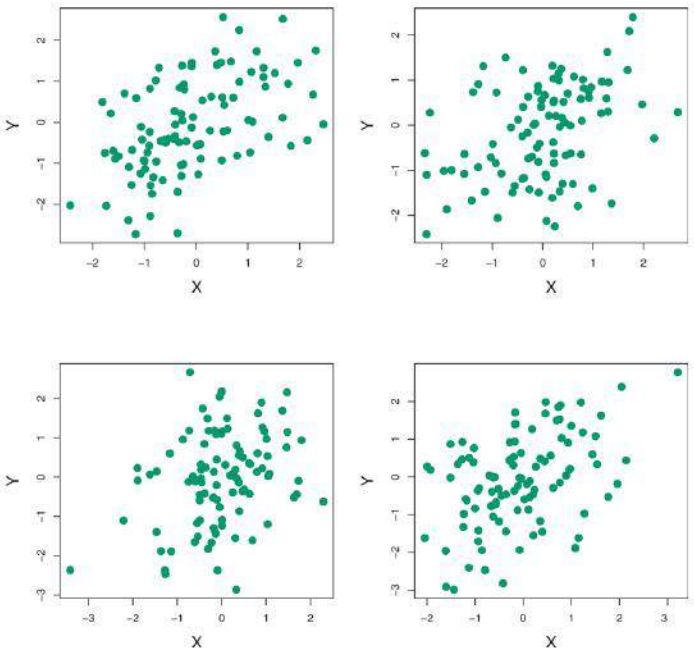

РИСУНОК 5.9. *На каждом графике изображены 100 имитированных значений доходности инвестиционных активов X и Y. Слева направо и сверху вниз: полученные оценки α составляют 0.576, 0.532, 0.657 и 0.651*

Естественным является желание дать количественную характеристику точности нашей оценки α. Для нахождения оценки стандартного отклонения $\hat{\alpha}$ мы повторили процесс имитации 100 пар наблюдений X и Y и расчета α по формуле (5.7) 1000 раз. Тем самым мы получили 1000 оценок α, которые можно обозначить как $\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{1000}$. На рис. 5.10 слева показана гистограмма полученных оценок. Для этих имитаций были использованы параметры $\sigma_X^2 = 1$, $\sigma_Y^2 = 1.25$ и $\sigma_{XY} = 0.5$, и поэтому мы знаем, что $\alpha = 0.6$. Мы отметили это значение при помощи сплошной вертикальной линии на гистограмме. Среднее значение из 1000 оценок α составляет

$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996,$$

что очень близко к $\alpha = 0.6$, а стандартное отклонение этой ошибки составляет

$$\sqrt{\frac{1}{1000-1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083.$$

в русскоязычных источниках по статистике и машинному обучению. — *Прим. пер.*

Это дает нам очень хорошее представление о точности $\hat{\alpha}$: $\mathrm{SE}(\hat{\alpha}) \approx 0.083$. Таким образом, для случайной выборки из генеральной совокупности мы могли бы ожидать отклонение $\hat{\alpha}$ от α в среднем примерно на 0.08.

Однако на практике описанная выше процедура нахождения $\mathrm{SE}(\hat{\alpha})$ неприменима, поскольку в реальности мы не можем взять новые выборки из исходной генеральной совокупности. Тем не менее метод бутстрепа позволяет нам использовать компьютер для подражания процессу создания новых выборок, что дает нам возможность оценить вариабельность $\hat{\alpha}$ без создания настоящих дополнительных выборок. Вместо многократного получения независимых наборов данных из генеральной совокупности мы получаем различные наборы данных путем многократного извлечения наблюдений *из уже имеющегося набора данных.*

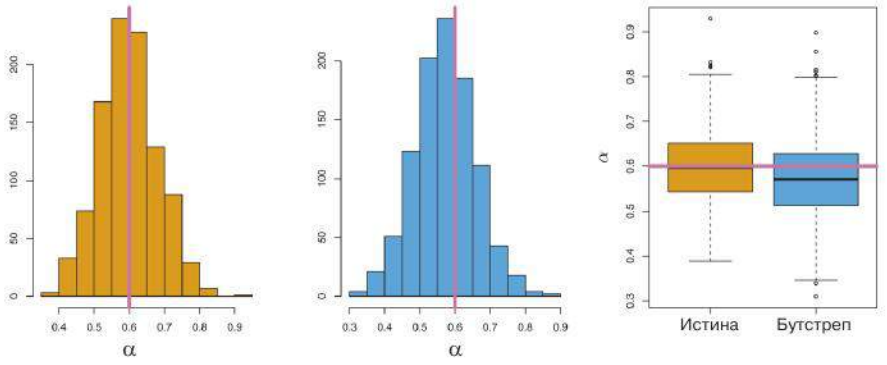

РИСУНОК 5.10. Слева: *гистограмма оценок α, полученных путем создания 1000 имитированных выборок из генеральной совокупности.* В центре: *гистограмма оценок α, полученных на основе 1000 бутстреп–выборок из одного набора данных.* Справа: *оценки α, приведенные на графиках слева и в центре, показаны в виде диаграммы размахов. На каждом графике розовая линия показывает истинное значение α*

Этот подход проиллюстрирован на рис. 5.11 на примере простого набора данных Z, который содержит только 3 наблюдения. Мы случайным образом отбираем n наблюдений из этого набора данных для создания бутстреп–выборки Z^{*1}. Отбор выполняется с *возвращением*[7] — это означает, что одно и то же наблюдение может встречаться в бутстреп–выборке более одного раза. В этом примере третье наблюдение встречается в Z^{*1} два раза, первое — один раз, а второе — ни одного раза. Заметьте, что если Z^{*1} содержит то или иное наблюдение, то включены как X–, так и Y–значения этого наблюдения. Мы можем использовать Z^{*1} для нахождения оценки α, которую мы обозначим как $\hat{\alpha}^{*1}$. Эта процедура повторяется определенное большое число раз B с целью получения B различных бутстреп–выборок $Z^{*1}, Z^{*2}, \ldots, Z^{*B}$ и B соответствующих оценок α — $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \ldots, \hat{\alpha}^{*B}$. Используя следующую формулу, мы можем вычислить стандартную ошибку этих бутстреп–оценок:

возвращение

[7] В оригинале для обозначения этой процедуры используется термин «sampling with replacement». — *Прим. пер.*

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2}. \qquad (5.8)$$

Эта величина служит оценкой стандартной ошибки $\hat{\alpha}$, полученной на основе исходного набора данных.

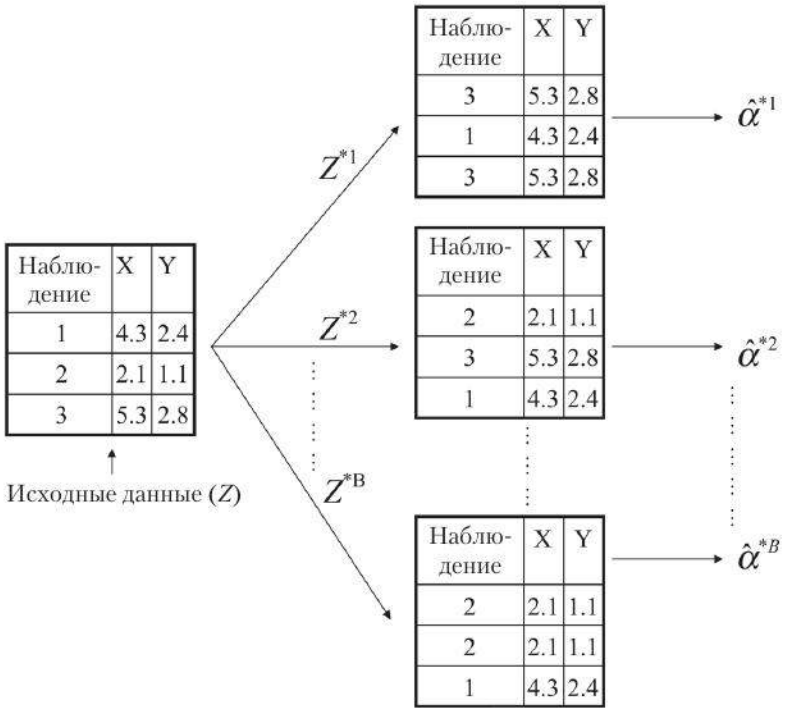

РИСУНОК 5.11. *Схематическое изображение принципа работы бутстрепа на примере небольшого набора данных с $n = 3$ наблюдениями. Каждая бутстреп–выборка содержит 3 наблюдения, отобранных с возвращением из исходного набора данных. Каждая из этих бутстреп–выборок используется для нахождения оценки α.*

Метод бутстрепа проиллюстрирован на рис. 5.10, в центре которого приведена гистограмма 1000 оценок α, каждая из которых была вычислена по отдельной бутстреп–выборке. Эта гистограмма была построена на основе единственного набора данных, из чего следует, что ее можно было бы создать и с использованием реальных данных. Заметьте, что эта гистограмма очень похожа на показанную слева идеализированную гистограмму оценок α, полученных путем создания 1000 имитированных выборок из генеральной совокупности. В частности, бутстреп–оценка $\mathrm{SE}(\hat{\alpha})$, рассчитанная по формуле (5.8), составляет 0.087, что очень близко к 0.083 — оценке, полученной на основе 1000 имитированных выборок. Справа показана та же информация, что и на первых двух графиках, но другим

образом — в виде диаграммы размахов оценок α, полученных путем создания 1000 имитированных выборок из генеральной совокупности и на основе бутстреп–выборок. Как и раньше, наблюдается близкое сходство этих двух множеств оценок, что указывает на возможность эффективного применения бутстрепа для нахождения оценки вариабельности $\hat{\alpha}$.

5.3 Лабораторная работа: перекрестная проверка и бутстреп

В этой лабораторной работе мы рассмотрим методы создания повторных выборок, описанные в данной главе. Выполнение некоторых команд из этой лабораторной работы на вашем компьютере может занять определенное время.

5.3.1 Метод проверочной выборки

Мы рассмотрим применение метода проверочной выборки для оценивания ошибок предсказаний разных моделей, построенных по данным из таблицы `Auto`.

Сначала мы воспользуемся функцией `set.seed()` для установления *«зерна»* генератора случайных чисел `R`, что позволит читателю этой книги получить результаты, идентичные приведенным ниже. Обычно при выполнении анализа, включающего элемент случайности (как, например, в случае с перекрестной проверкой), установление такого «зерна» является хорошей идеей, поскольку в будущем это позволит с точностью воспроизвести полученные результаты.

зерно

Мы начнем с вызова функции `sample()`, позволяющей разбить данные на две половины путем случайного извлечения 196 наблюдений из исходных 392 наблюдений. Мы будем называть эти наблюдения обучающей выборкой.

sample()

```
> library(ISLR)
> set.seed(1)
> train = sample(392, 196)
```

(Здесь мы используем сокращенную форму записи команды `sample()`; подробнее см. `?sample`.) Далее мы воспользуемся аргументом `subset` функции `lm()` для подгонки линейной модели на основе только тех наблюдений, которые входят в состав обучающей выборки.

```
> lm.fit = lm(mpg ~ horsepower, data = Auto, subset = train)
```

Теперь мы применим функцию `predict()`, чтобы предсказать отклик для всех 392 наблюдений, а затем — функцию `mean()` для вычисления MSE по 196 наблюдениям из проверочной выборки. Обратите внимание, что индекс `-train` в приведенной ниже команде выбирает только те наблюдения, которые не являются частью обучающей выборки.

```
> attach(Auto)
> mean((mpg - predict(lm.fit, Auto))[-train]^2)
[1] 26.14
```

Таким образом, оценка среднеквадратичной ошибки на контрольной выборке для этой линейной регрессионной модели составляет 26.14. Мы можем воспользоваться функцией poly() для оценивания ошибки на контрольной выборке для квадратичной и кубической регрессионных моделей.

```
> lm.fit2 = lm(mpg ~ poly(horsepower, 2), data = Auto,
                subset = train)
> mean((mpg - predict(lm.fit2, Auto))[-train]^2)
[1] 19.82
> lm.fit3 = lm(mpg ~ poly(horsepower, 3), data = Auto,
                subset = train)
> mean((mpg - predict(lm.fit3, Auto))[-train]^2)
[1] 19.78
```

Эти ошибки составляют 19.82 и 19.78 соответственно. Если мы выберем несколько отличающуюся обучающую выборку, то получим немного другие ошибки на проверочной выборке.

```
> set.seed(2)
> train = sample(392, 196)
> lm.fit = lm(mpg ~ horsepower, subset = train)
> mean((mpg - predict(lm.fit, Auto))[-train]^2)
[1] 23.30
> lm.fit2 = lm(mpg ~ poly(horsepower, 2), data = Auto,
                subset = train)
> mean((mpg - predict(lm.fit2, Auto))[-train]^2)
[1] 18.90
> lm.fit3 = lm(mpg ~ poly(horsepower, 3), data = Auto,
                subset = train)
> mean((mpg - predict(lm.fit3, Auto))[-train]^2)
[1] 19.26
```

Разбивая наблюдения на обучающую и проверочную выборки, мы обнаружим, что ошибки на проверочной выборке для линейной, квадратичной и кубической моделей составляют 23.30, 18.90 и 19.26 соответственно.

Эти результаты согласуются с нашими предыдущими выводами: модель, которая предсказывает mpg на основе квадратичной функции horsepower, обладает более высоким качеством, чем модель, включающая только линейную функцию horsepower, и есть мало оснований предпочесть модель, основанную на кубической функции horsepower.

5.3.2 Перекрестная проверка по отдельным наблюдениям

LOOCV–оценку ошибки можно вычислить автоматически для любой обобщенной линейной модели при помощи функций glm() и cv.glm(). В лабораторной работе к главе 4 для построения логистической регрессии мы применили функцию glm() в сочетании с аргументом family = "binomial". Но если мы вызовем glm() без указания аргумента family, то она рассчитает линейную регрессию, в точности как функция lm(). Например, команды

cv.glm()

```
> glm.fit = glm(mpg ~ horsepower, data = Auto)
> coef(glm.fit)
(Intercept)  horsepower
     39.936      -0.158
```

и

```
> lm.fit = lm(mpg ~ horsepower, data = Auto)
> coef(lm.fit)
(Intercept)  horsepower
     39.936      -0.158
```

дают идентичные линейные регрессионные модели. В этой лабораторной работе мы будем строить линейные регрессионные модели при помощи функции `glm()`, а не `lm()`, поскольку первую из этих функций можно использовать совместно с `cv.glm()`. Функция `cv.glm()` является частью библиотеки `boot`.

```
> library(boot)
> glm.fit = glm(mpg ~ horsepower, data = Auto)
> cv.err = cv.glm(Auto, glm.fit)
> cv.err$delta
    1     1
24.23 24.23
```

Функция `cv.glm()` возвращает список с несколькими компонентами. Два числа, хранящихся в векторе `delta`, представляют собой результаты перекрестной проверки. В данном случае эти числа одинаковы (с точностью до второго знака) и соответствуют LOOCV–оценке, определение которой дано в (5.1). Ниже мы обсуждаем ситуацию, в которой эти два числа различаются. Наша LOOCV–оценка ошибки на контрольном множестве примерно равна 24.23.

Мы можем повторить эту процедуру для полиномиальных моделей с возрастающей сложностью структуры. Чтобы автоматизировать данный процесс, мы применяем функцию `for()` для инициализации цикла, который последовательно подгоняет полиномиальные регрессионные модели со степенью от $i = 1$ до $i = 5$, вычисляет соответствующую ошибку при помощи перекрестной проверки и сохраняет результат в i–м элементе вектора `cv.error`. Мы начинаем с инициализации этого вектора. Скорее всего, на выполнение цикла потребуется пара минут.

```
> cv.error = rep(0, 5)
> for(i in 1:5){
+ glm.fit = glm(mpg ~ poly(horsepower, i), data = Auto)
+ cv.error[i] = cv.glm(Auto, glm.fit)$delta[1]
+ }
> cv.error
[1] 24.23 19.25 19.33 19.42 19.03
```

Как и на рис. 5.4, мы наблюдаем резкое снижение оценки MSE на контрольной выборке при переходе от линейной к квадратичной модели и отсутствие какого–либо выраженного улучшения при дальнейшем увеличении степени полиномов.

5.3.3 k–кратная перекрестная проверка

Функцию `cv.glm()` можно применять также для выполнения k–кратной перекрестной проверки. Ниже для набора данных `Auto` мы используем $k = 10$ — часто применяемое значение k. Как и ранее, мы устанавливаем значение зерна генератора случайных чисел и инициализируем вектор, в котором мы будем сохранять оценки ошибок, соответствующие полиномиальным моделям со степенью от 1 до 10.

```
> set.seed(17)
> cv.error.10 = rep(0, 10)
> for (i in 1:10) {
+ glm.fit = glm(mpg ~ poly(horsepower, i), data = Auto)
+ cv.error.10[i] = cv.glm(Auto, glm.fit, K = 10)$delta[1] }
> cv.error.10
[1] 24.21 19.19 19.31 19.34 18.88 19.02 18.90 19.71 18.95 19.50
```

Заметьте, что длительность вычислений намного короче, чем в случае с LOOCV. (В принципе, благодаря формуле (5.2) длительность вычислений у LOOCV для линейной модели, подгоняемой по методу наименьших квадратов, должна быть короче времени, уходящего на реализацию k–кратной перекрестной проверки, но, к сожалению, функция `cv.glm()` не использует преимущество этой формулы.) Мы по–прежнему не наблюдаем убедительного свидетельства в пользу того, что полиномы третьей или более высокой степени приводят к более низкой ошибке на контрольном множестве, по сравнению с квадратичной моделью.

В подразделе 5.3.2 мы видели, что при выполнении LOOCV два числа из вектора `delta` оказались практически одинаковыми. Однако при использовании k–кратной перекрестной проверки они несколько различаются. Первое из этих чисел — это стандартная оценка, полученная при помощи k–кратной перекрестной проверки, согласно (5.3). Второе число — это та же оценка, но с поправкой на смещение. Для этого набора данных эти две оценки очень похожи.

5.3.4 Бутстреп

Мы продемонстрируем использование бутстрепа на примере простого примера из раздела 5.2, а также на примере оценивания точности предсказаний линейной регрессионной модели, построенной по данным `Auto`.

Оценивание точности статистического параметра

Одним из больших преимуществ метода бутстрепа является то, что его можно применять почти во всех ситуациях. Не требуется никаких сложных математических вычислений. Выполнение бутстрепа в `R` состоит из двух шагов. На первом шаге мы должны создать функцию, которая вычисляет интересующий нас параметр. На втором шаге мы применяем функцию `boot()` из библиотеки `boot` для реализации бутстрепа путем многократного извлечения выборок из данных с возвращением.

`boot()`

Набор данных `Portfolio` из пакета `ISLR` описан в разделе 5.2. Чтобы применить бутстреп к этим данным, вы сначала должны создать функцию

alpha.fn(), которая принимает данные (X, Y) и вектор с индексными номерами наблюдений, используемых для вычисления α. Затем эта функция возвращает оценку α, рассчитанную на основе соответствующих наблюдений.

```
> alpha.fn = function(data, index) {
+ X = data$X[index]
+ Y = data$Y[index]
+ return((var(Y) - cov(X,Y)) / (var(X) + var(Y) - 2*cov(X, Y)))
+ }
```

Эта функция возвращает оценку α, полученную в результате применения формулы (5.7) к наблюдениям с индексными номерами index. Например, следующая команда говорит R рассчитать α по всем 100 наблюдениям:

```
> alpha.fn(Portfolio, 1:100)
[1] 0.576
```

Следующая команда использует функцию sample() для случайного извлечения с возвращением 100 наблюдений из диапазона от 1 до 100. Это аналогично созданию новой бутстреп–выборки и повторному вычислению $\hat{\alpha}$ на основе этих новых данных.

```
> set.seed(1)
> alpha.fn(Portfolio, sample(100, 100, replace = TRUE))
[1] 0.596
```

Мы можем реализовать бутстреп–анализ путем многократного повторения этой команды и сохранения соответствующих оценок α для дальнейшего вычисления их стандартного отклонения. Однако функция boot() автоматизирует этот процесс. Ниже мы создаем $R = 1000$ бутстреп–оценок α.

```
> boot(Portfolio, alpha.fn, R = 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Portfolio, statistic = alpha.fn, R = 1000)

Bootstrap Statistics :
     original      bias        std.error
t1*  0.5758      -7.315e-05   0.0886
```

Эти результаты показывают, что оценка α, рассчитанная на основе исходных данных, составляет 0.5758, а бутстреп–оценка $\mathrm{SE}(\hat{\alpha})$ — 0.0886.

Оценивание точности линейной регрессионной модели

Метод бутстрепа можно применить для оценивания вариабельности коэффициентов, а также точности предсказаний той или иной статистической модели. Здесь мы применим бутстреп для оценивания вариабельности оценок коэффициентов β_0 и β_1 — свободного члена и углового коэффициента

линейной регрессионной модели, построенной по данным Auto для предсказания mpg по horsepower. Мы сравним бутстреп–оценки с оценками, вычисленными по формулам для $\mathrm{SE}(\hat{\beta}_0)$ и $\mathrm{SE}(\hat{\beta}_1)$ из подраздела 3.1.2.

Сначала мы создаем простую функцию boot.fn(), которая принимает данные Auto и набор индексных номеров наблюдений и возвращает оценки свободного члена и углового коэффициента линейной регрессионной модели. Затем мы применяем эту функцию ко всей совокупности из 392 наблюдений для вычисления оценок β_0 и β_1 по обычным формулам для коэффициентов линейной регрессионной модели (глава 3). Заметьте, что нам не нужны фигурные скобки в начале и в конце тела функции, поскольку оно состоит только из одной строки.

```
> boot.fn = function(data, index) {
return(coef(lm(mpg ~ horsepower, data = data, subset = index)))}
> boot.fn(Auto, 1:392)
(Intercept)  horsepower
   39.936   -0.158
```

Функцию boot.fn() можно использовать также для создания бутстреп–оценок свободного члена и углового коэффициента путем случайного извлечения наблюдений с возвращением. Ниже мы приводим два примера.

```
> set.seed(1)
> boot.fn(Auto, sample(392, 392, replace = TRUE))
(Intercept)  horsepower
  38.739     -0.148
> boot.fn(Auto, sample(392, 392, replace = TRUE))
(Intercept)  horsepower
  40.038     -0.160
```

Далее мы применяем функцию boot() для вычисления стандартных ошибок 1000 бутстреп–оценок свободного члена и углового коэффициента.

```
> boot(Auto, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto, statistic = boot.fn, R = 1000)

Bootstrap Statistics :
    original    bias     std. error
t1* 39.936      0.0297   0.8600
t2* -0.158     -0.0003   0.0074
```

Из этих результатов следует, что бутстреп–оценка $\mathrm{SE}(\hat{\beta}_0)$ составляет 0.86, а бутстреп–оценка $\mathrm{SE}(\hat{\beta}_1)$ — 0.0074. Как обсуждалось в подразделе 3.1.2, для вычисления стандартных ошибок коэффициентов линейной регрессионной модели можно применять готовые формулы. Такие оценки можно получить при помощи функции summary().

Стандартные ошибки оценок $\hat{\beta}_0$ и $\hat{\beta}_1$, вычисленные по формулам из подраздела 3.1.2, составляют 0.717 для свободного члена и 0.0064 для углового коэффициента. Интересно, что эти значения несколько отличаются

от оценок, полученных с применением бутстрепа. Указывает ли это на проблему с бутстрепом? В действительности дело обстоит иначе. Вспомните, что стандартные формулы, приведенные в уравнении на стр. 78, основаны на определенных допущениях. Например, они зависят от неизвестного параметра σ^2 — остаточной дисперсии. В связи с этим мы оцениваем σ^2 по RSS. Хотя формулы для стандартных ошибок коэффициентов не зависят от правильности линейной спецификации модели, формула для оценки σ^2 — зависит. На рис. 3.8 (стр. 104) мы видим, что в данных имеет место нелинейная зависимость, в связи с чем остатки линейной модели, а следовательно и $\hat{\sigma}^2$, будут завышенными. Во–вторых, стандартные формулы предполагают (отчасти нереалистично), что x_i являются фиксированными значениями и вся вариабельность обусловлена разбросом остатков ϵ_i. Метод бутстрепа не полагается ни на одно из этих допущений и поэтому с большой вероятностью дает более точные оценки стандартных ошибок $\hat{\beta}_0$ и $\hat{\beta}_1$, чем функция `summary()`.

Ниже мы вычисляем бутстреп–оценки и обычные оценки стандартных ошибок коэффициентов квадратичной модели, подогнанной к тем же данным. Поскольку квадратичная модель хорошо описывает эти данные (рис. 3.8), здесь мы наблюдаем более тесное соответствие между бутстреп–оценками и обычными оценками $\mathrm{SE}(\hat{\beta}_0)$, $\mathrm{SE}(\hat{\beta}_1)$ и $\mathrm{SE}(\hat{\beta}_2)$.

```
> boot.fn = function(data, index)
+ coefficients(lm(mpg ~ horsepower + I(horsepower^2),
                  data = data, subset = index))
> set.seed(1)
> boot(Auto, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = Auto, statistic = boot.fn, R = 1000)

Bootstrap Statistics :
     original    bias     std. error
t1*  56.900    6.098e-03  2.0945
t2* -0.466    -1.777e-04  0.0334
t3*  0.001     1.324e-06  0.0001
> summary(lm(mpg ~ horsepower + I(horsepower^2),
             data = Auto))$coef
                 Estimate  Std. Error   t value   Pr(>|t|)
(Intercept)       56.9001     1.80043        32  1.7e-109
horsepower       -0.4662      0.03112       -15   2.3e-40
I(horsepower^2)   0.0012      0.00012        10   2.2e-21
```

5.4 Упражнения

Теоретические

1. Используя основные статистические свойства дисперсии, а также приемы интегрального исчисления для одной переменной, выведите

формулу (5.6). Другими словами, докажите, что α из (5.6) действительно минимизирует $\mathrm{Var}(\alpha X + (1-\alpha)Y)$.

2. Теперь мы определим вероятность попадания того или иного наблюдения в бутстреп–выборку. Предположим, что мы делаем бутстреп–выборку из совокупности, содержащей n наблюдений.

   (a) Какова вероятность того, что первое наблюдение в бутстреп–выборке *не является* j–м наблюдением в исходной совокупности?

   (b) Какова вероятность того, что второе наблюдение *не является* j–м наблюдением в исходной совокупности?

   (c) Покажите, что вероятность *невхождения* j–го наблюдения в бутстреп–выборку составляет $(1-1/n)^n$.

   (d) Чему равна вероятность того, что j–е наблюдение входит в бутстреп–выборку при $n = 5$?

   (e) Чему равна вероятность того, что j–е наблюдение входит в бутстреп–выборку при $n = 100$?

   (f) Чему равна вероятность того, что j–е наблюдение входит в бутстреп–выборку при $n = 10\,000$?

   (g) Постройте график, который показывает вероятность вхождения j–го наблюдения в бутстреп-выборку для целых чисел n от 1 до 100 000. Прокомментируйте, что вы видите.

   (h) Теперь мы количественно исследуем вероятность того, что бутстреп–выборка объемом $n = 100$ содержит j–е наблюдение. Здесь $j = 4$. Мы многократно создаем бутстреп–выборки и каждый раз отмечаем факт вхождения четвертого наблюдения в конкретную бутстреп–выборку.

```
> store = rep(NA, 10000)
> for (i in 1:10000) {
store[i] = sum(sample(1:100, rep = TRUE) == 4) > 0
}
> mean(store)
```

   Прокомментируйте полученные результаты.

3. Теперь мы повторим пройденный материал по k–кратной перекрестной проверке.

   (a) Объясните, как работает k–кратная перекрестная проверка.

   (b) Каковы преимущества и недостатки k–кратной перекрестной проверки, по сравнению с

      i. методом проверочной выборки;

      ii. LOOCV?

4. Предположим, что мы применяем тот или иной метод статистического обучения для предсказания отклика Y по некоторому значению предиктора X. Опишите подробно, как мы могли бы оценить стандартное отклонение этого предсказываемого значения.

Практические

5. В главе 4 мы воспользовались логистической регрессией для предсказания вероятности неуплаты долга по кредитной карте (`default`) на основе дохода клиента (`income`) и величины долга (`balance`) (данные из таблицы `Default`). Теперь мы применим метод проверочной выборки для оценивания частоты ошибок, совершаемых этой логистической регрессионной моделью. Не забудьте перед началом анализа установить зерно генератора случайных чисел.

   (a) Постройте логистическую регрессионную модель, которая использует `income` и `balance` для предсказания `default`.

   (b) Применив метод проверочной выборки, оцените частоту ошибок, совершаемых этой моделью. Для этого вы должны выполнить следующие шаги:

      i. Разбейте исходные данные на обучающую и проверочную выборки.

      ii. Подгоните множественную логистическую регрессионную модель, используя только обучающие данные.

      iii. Получите предсказания неуплаты долга для каждого клиента из проверочной выборки, вычислив апостериорную вероятность дефолта для соответствующего клиента и отнеся его к категории неплательщиков в случае, если эта апостериорная вероятность превышает 0.5.

      iv. Вычислите частоту ошибок на проверочной выборке, которая представляет собой долю неверно классифицированных наблюдений в проверочной выборке.

   (c) Повторите процесс из (b) три раза, используя разные разбиения наблюдений на обучающую и проверочную выборки. Прокомментируйте полученные результаты.

   (d) Теперь мы рассмотрим логистическую регрессионную модель, которая предсказывает вероятность дефолта на основе `income`, `balance` и индикаторной переменной `student`. Оцените частоту ошибок, совершаемых этой моделью на новых данных, используя метод проверочной выборки. Дайте комментарий относительно того, приводит ли включение переменной `student` к снижению частоты ошибок на проверочной выборке.

6. Мы продолжаем рассмотрение логистической регрессионной модели, предсказывающей вероятность `default` на основе `income` и `balance` по данным из таблицы `Default`. В частности, теперь мы вычислим оценки стандартных ошибок коэффициентов `income` и `balance` в этой модели, применив два разных способа: (1) бутстреп и (2) использование готовых формул, реализованных в функции `glm()`. Не забудьте перед началом анализа установить зерно генератора случайных чисел.

   (a) Воспользовавшись функциями `summary()` и `glm()`, определите стандартные ошибки коэффициентов переменных `income` и `balance` из множественной логистической регрессионной модели, которая включает оба предиктора.

(b) Напишите функцию `boot.fn()`, которая принимает таблицу данных `Default` и вектор с индексными номерами наблюдений и возвращает оценки коэффициентов `income` и `balance` из множественной логистической регрессионной модели.

(c) Примените функцию `boot` вместе с вашей функцией `boot.fn()` для нахождения оценок стандартных ошибок регрессионных коэффициентов `income` и `balance`.

(d) Прокомментируйте величины стандартных ошибок, полученных при помощи функции `glm()` и вашей бутстреп–функции.

7. В подразделах 5.3.2 и 5.3.3 мы видели, что для вычисления LOOCV–оценок ошибки на контрольной выборке можно использовать функцию `cv.glm()`. В качестве альтернативы эти вычисления можно было бы проделать просто с помощью функций `glm()` и `predict.glm()`, организовав `for`–цикл. Сейчас вы реализуете этот подход для вычисления LOOCV–ошибки для простой логистической регрессионной модели на примере данных `Weekly`. Напомним, что определение LOOCV–ошибки для задач классификации дано в формуле (5.4).

(a) Постройте логистическую регрессионную модель, которая предсказывает `Direction` по `Lag1` и `Lag2`.

(b) Постройте логистическую регрессионную модель, которая предсказывает `Direction` на основе `Lag1` и `Lag2` *по всем наблюдениям, за исключением первого.*

(c) Примените модель из (b), чтобы предсказать направление движения рынка для первого наблюдения. Вы можете сделать это, предсказав рост рынка при $P(\texttt{Direction = "Up"} \mid \texttt{Lag1}, \texttt{Lag2}) > 0.5$. Правильным ли оказалось предсказание для этого наблюдения?

(d) Напишите `for`–цикл для i от 1 до n, где n — это число наблюдений в таблице данных, который выполняет следующие шаги:

i. Подгонка логистической регрессионной модели для предсказания `Direction` по `Lag1` и `Lag2` с использованием всех наблюдений, за исключением i–го.

ii. Вычисление апостериорной вероятности роста рынка для i–го наблюдения.

iii. Предсказание роста рынка на основе апостериорной вероятности для i–го наблюдения.

iv. Определение того, была ли допущена ошибка в отношении направления движения рынка для i–го наблюдения. Если ошибка был допущена, покажите это при помощи 1, а если нет — при помощи 0.

(e) Вычислите среднее значение из n чисел, полученных в (d)iv, для нахождения LOOCV–оценки ошибки на контрольной выборке. Прокомментируйте свои результаты.

8. Теперь мы выполним перекрестную проверку на имитированном наборе данных.

(a) Создайте набор имитированных данных следующим образом:

```
> set.seed(1)
> y = rnorm(100)
> x = rnorm(100)
> y = x - 2*x^2 + rnorm(100)
```

Что в этом наборе данных n, а что — p? Запишите модель, породившую эти данные, в виде уравнения.

(b) Постройте график зависимости Y от X. Прокомментируйте, что вы видите.

(c) Задайте зерно генератора случайных чисел и вычислите LOOCV–ошибки предсказаний, возникающие в результате подгонки следующих четырех моделей по методу наименьших квадратов:

i. $Y = \beta_0 + \beta_1 X + \epsilon$

ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$

iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$

iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$.

Обратите внимание: для создания единой таблицы данных, содержащей X и Y, полезной может оказаться функция `data.frame()`.

(d) Повторите (c), применив другое зерно генератора случайных чисел, и опишите свои результаты. Похожи ли эти результаты на то, что вы получили в (c)? Почему?

(e) Какая из четырех моделей в (c) обладала наиболее низкой LOOCV–ошибкой? Ожидали ли вы такой результат? Объясните свой ответ.

(f) Дайте комментарии в отношении статистической значимости оценок коэффициентов, полученных для каждой модели в (c) методом наименьших квадратов. Согласуются ли эти результаты с выводами, сделанными на основе результатов перекрестной проверки?

9. Теперь мы рассмотрим набор данных `Boston` из библиотеки `MASS`.

(a) Используя этот набор данных, вычислите оценку среднего значения `medv` в генеральной совокупности. Обозначьте эту оценку как $\hat{\mu}$.

(b) Вычислите оценку стандартной ошибки $\hat{\mu}$. Интерпретируйте этот результат.

Подсказка: мы можем вычислить стандартную ошибку выборочной средней, разделив выборочное стандартное отклонение на квадратный корень из объема выборки.

(c) Теперь оцените стандартную ошибку $\hat{\mu}$, применив бутстреп. Насколько она отличается от величины, полученной вам в (b)?

(d) Используя свою бутстреп–оценку из (c), вычислите 95%–ный доверительный интервал для среднего значения `medv`. Сравните его с результатами, полученными при помощи `t.test(Boston$medv)`.

Подсказка: вы можете вычислить приблизительный 95%–ный доверительный интервал по формуле $[\hat{\mu} - 2\mathrm{SE}(\hat{\mu}), \hat{\mu} + 2\mathrm{SE}(\hat{\mu})]$.

(e) Используя этот набор данных, вычислите оценку медианы ($\hat{\mu}_{med}$) переменной `medv` в генеральной совокупности.

(f) Теперь мы хотели бы оценить стандартную ошибку $\hat{\mu}_{med}$. К сожалению, простой формулы для вычисления стандартной ошибки медианы не существует. Поэтому для получения такой оценки примените бутстреп. Прокомментируйте свои результаты.

(g) Используя этот набор данных, вычислите десятый процентиль ($\hat{\mu}_{0.1}$) стоимости домов в пригородах Бостона. (Вы можете использовать для этого функцию `quantile()`.)

(h) Воспользуйтесь бутстрепом для нахождения стандартной ошибки $\hat{\mu}_{0.1}$. Прокомментируйте свои результаты.

Глава 6

Отбор и регуляризация линейных моделей

В регрессионных задачах для описания связи между откликом Y и набором переменных $X_1, X_2, \ldots, X_p$ часто используется стандартная линейная модель вида

$$Y = \beta_0 + \beta_1 X_1 + \cdots \beta_p X_p + \epsilon. \tag{6.1}$$

В главе 3 мы видели, что обычно эту модель подгоняют по методу наименьших квадратов.

В следующих главах мы рассмотрим некоторые подходы, расширяющие методологию линейных моделей. В главе 7 мы обобщим (6.1) на случай нелинейных, но все же аддитивных зависимостей, а в главе 8 мы рассмотрим еще более общие нелинейные модели. Однако линейная модель имеет выраженное преимущество с точки зрения возможности сделать статистические выводы, и при решении практических проблем она оказывается удивительно хорошим конкурентом нелинейных методов. Поэтому, прежде чем погрузиться в «нелинейный мир», в этой главе мы обсудим некоторые способы, позволяющие усовершенствовать простую линейную модель с помощью процедур оценки параметров, альтернативных обычному методу наименьших квадратов.

Зачем нам использовать какую–то другую процедуру вместо метода наименьших квадратов? Как мы вскоре увидим, альтернативные процедуры могут привести к более высокой *точности предсказаний* и *интерпретируемости модели*.

- *Точность предсказаний*: при условии, что истинная зависимость между откликом и предикторами является примерно линейной, метод наименьших квадратов будет обладать низким смещением. При $n \gg p$, т. е. когда количество наблюдений n намного превышает количество переменных p, метод наименьших квадратов обычно обладает также низкой дисперсией и, следовательно, обеспечивает хорошее качество предсказаний на контрольных наблюдениях. Однако если n превышает p незначительно, то модель, подогнанная по методу наименьших квадратов, может проявлять заметную дисперсию, что сопровождается переобучением и, как следствие, низким качеством

предсказаний на новых наблюдениях, которые не были задействованы в обучении модели. В случае же, когда $p > n$, метод наименьших квадратов не способен дать уникальные оценки параметров модели: дисперсия *бесконечна*, и этот метод вообще неприменим. *Накладывая ограничения* на оцененные коэффициенты, или «*сжимая*» их, мы можем существенно снизить дисперсию за счет незначительного увеличения смещения. Это может привести к существенным улучшениям точности предсказаний отклика на наблюдениях, которые не были использованы в обучении модели.

- *Интерпретируемость модели*: часто случается так, что некоторые или даже многие переменные, включенные в регрессионную модель, в действительности не связаны с откликом. Включение таких *несущественных* переменных приводит к ненужной сложности итоговой модели. Удалив такие переменные, т. е. приравняв их коэффициенты к нулю, мы можем получить легко интерпретируемую модель. При использовании метода наименьших квадратов получение каких–либо коэффициентов, в точности равных нулю, крайне маловероятно. В этой главе мы рассмотрим некоторые подходы для автоматического *отбора признаков*, или *отбора переменных*, т. е. исключения несущественных переменных из множественной регрессионной модели.

отбор признаков

отбор переменных

Существует много альтернатив (как классических, так и современных) использованию метода наименьших квадратов для подгонки модели (6.1). В этой главе мы обсуждаем три важных класса таких методов.

- *Отбор подмножества переменных*: этот подход включает определение некоторого набора переменных, которые, как мы думаем, связаны с откликом. Далее мы подгоняем модель по методу наименьших квадратов с использованием этого уменьшенного набора переменных.

- *Сжатие*: этот подход включает подгонку модели, содержащей все p предикторов. Однако, в отличие от коэффициентов, получаемых по методу наименьших квадратов, здесь оцененные коэффициенты «сжимаются» в направлении к нулю. Эффектом такого сжатия (также известного как «*регуляризация*») является снижение дисперсии. В зависимости от типа выполняемого сжатия оценки некоторых коэффициентов могут оказаться в точности равными нулю. Следовательно, методы сжатия могут выполнять отбор переменных.

- *Снижение размерности*: этот подход включает проецирование p предикторов на M–мерное подпространство, где $M < p$. Это достигается путем вычисления M различных *линейных комбинаций*, или *проекций*, переменных. Далее эти проекции используются в качестве предикторов для подгонки линейной регрессионной модели по методу наименьших квадратов.

В следующих разделах мы описываем каждый из этих подходов более детально, а также обсуждаем их преимущества и недостатки. Хотя в этой главе описываются дополнения и модификации линейной регрессионной модели из главы 3, те же концепции применимы и к другим методам, таким как методы классификации из главы 4.

6.1 Отбор подмножества переменных

В этом разделе мы рассматриваем некоторые методы отбора подмножества предикторов. Они включают процедуры нахождения оптимального подмножества предикторов и пошагового отбора моделей.

6.1.1 Отбор оптимального подмножества

отбор оптимального подмножества

Чтобы выполнить *отбор оптимального подмножества*[1], мы подгоняем отдельную регрессию по методу наименьших квадратов для всех возможных комбинаций из p предикторов. Другими словами, мы подгоняем все p моделей, содержащих только один предиктор, все $\binom{p}{2} = p(p-1)/2$ моделей, содержащих в точности два предиктора, и т. д. Далее мы проверяем все получившиеся модели с целью определить наиболее *оптимальную* из них.

Проблема нахождения *оптимальной* модели среди 2^p возможных вариантов путем отбора оптимального подмножества нетривиальна. Обычно эта процедура разбивается на две стадии, описанные в алгоритме 6.1.

Алгоритм 6.1 Отбор оптимального подмножества переменных

1. Пусть $\mathcal{M}_0$ обозначает *нулевую модель*, которая не содержит предикторов. Для каждого наблюдения эта модель предсказывает просто выборочное среднее значение.
2. Для $k = 1, 2, \ldots, p$:
   (a) Постройте все $\binom{p}{k}$ моделей, которые содержат в точности k предикторов.
   (b) Выберите *наиболее оптимальную* среди этих $\binom{p}{k}$ моделей и назовите ее $\mathcal{M}_k$. Здесь под оптимальной понимается модель, имеющая наименьшую сумму квадратов остатков (RSS), или, что эквивалентно, максимальное значение R^2.
3. Выберите единственную оптимальную модель среди $\mathcal{M}_0, \ldots, \mathcal{M}_p$, используя такие полученные при помощи перекрестной проверки показатели, как ошибка на контрольной выборке, C_p (AIC), BIC или скорректированный коэффициент детерминации R^2.

На втором шаге алгоритма 6.1 для каждого подмножества предикторов определяется оптимальная модель (по обучающим данным) с целью снижения размерности задачи с 2^p до $p + 1$ возможных моделей. На рис. 6.1 эти модели образуют нижнюю границу, обозначенную красным цветом.

Далее для нахождения единственной оптимальной модели мы просто должны выбрать среди $p + 1$ вариантов. Эту задачу следует выполнять с большой осторожностью, поскольку по мере увеличения количества предикторов RSS этих $p + 1$ моделей монотонно снижается, а R^2 монотонно

[1] В оригинале используется термин «best subset selection». — *Прим. пер.*

возрастает. Следовательно, если для выбора оптимальной модели мы применяем эти критерии, то в итоге мы всегда выберем модель, содержащую весь набор переменных. Проблема заключается в том, что низкая RSS или высокий коэффициент детерминации R^2 указывают на модель с низкой ошибкой на *обучающей* выборке, тогда как мы хотим выбрать модель с низкой ошибкой на контрольной выборке. (Как было показано в главе 2 на рис. 2.9–2.11, ошибка на обучающей выборке обычно существенно ниже ошибки на контрольной выборке, но при этом низкая ошибка на обучающей выборке никоим образом не гарантирует низкой ошибки на контрольной выборке.) Поэтому на шаге 3 для выбора среди $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_p$ мы применяем такие полученные при помощи перекрестной проверки показатели, как ошибка на контрольной выборке, C_p, BIC или скорректированный коэффициент детерминации R^2. Эти подходы обсуждаются в подразделе 6.1.3.

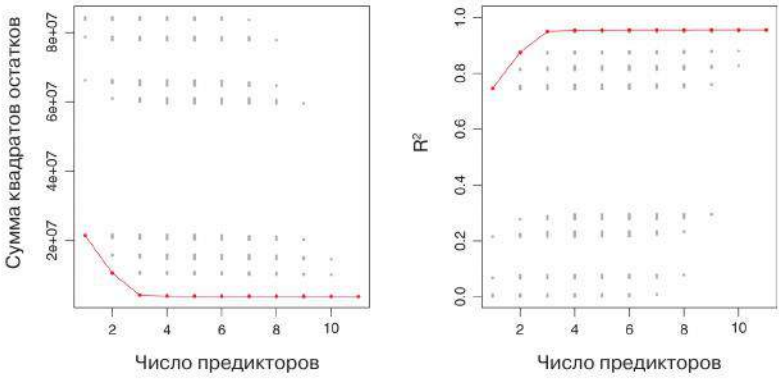

РИСУНОК 6.1. *Показаны RSS и R^2 для всех возможных моделей, содержащих определенное подмножество из десяти предикторов из набора данных* `Credit`. *Граница, показанная красным цветом, проходит через точки, которые соответствуют оптимальным (по RSS и R^2) моделям соответствующего размера. Хотя набор данных содержит десять предикторов, значения по оси X изменяются от 1 до 11, поскольку одна из переменных является категориальной и принимает три значения, что сопровождается созданием двух индикаторных переменных*

Пример применения отбора оптимального подмножества предикторов показан на рис. 6.1. Каждая точка на этом рисунке соответствует регрессионной модели, которая была подогнана по методу наименьших квадратов на основе определенного подмножества из 11 исходных предикторов, входящих в таблицу `Credit` (см. главу 3). Переменная `ethnicity` является качественной переменной с тремя уровнями, и поэтому она представлена двумя индикаторными переменными, которые в данном случае рассматриваются в качестве отдельных предикторов. Мы изобразили RSS и R^2 для каждой модели в зависимости от количества предикторов. Красная кривая соединяет точки, соответствующие оптимальным моделям каждого размера (согласно RSS и R^2). Как и следовало ожидать, рисунок показывает, что эти критерии качества улучшаются по мере увеличения

числа переменных; однако, начиная с модели с тремя переменными, добавление дополнительных предикторов сопровождается весьма незначительным улучшением RSS и R^2.

Хотя мы описали отбор оптимального подмножества переменных для регрессии, рассчитываемой по методу наименьших квадратов, та же идея применима и к другим типам моделей, таким как логистическая регрессия. В случае с логистической регрессией на втором шаге алгоритма 6.1 для упорядочивания моделей мы используем не RSS, а *аномальность*[2] — показатель, который играет роль RSS для более широкого класса моделей. Аномальность равна минус двум максимизированным значениям логарифма правдоподобия; чем меньше аномальность, тем лучше модель описывает данные.

аномальность

Несмотря на то что отбор оптимального подмножества предикторов является простым и концептуально привлекательным методом, он страдает ограничениями вычислительного характера. Число подлежащих рассмотрению возможных моделей быстро возрастает при увеличении p. В целом существует 2^p моделей, включающих подмножества p предикторов. Так, при $p = 10$ рассмотрению подлежит примерно 1000 возможных моделей, а при $p = 20$ число таких моделей превышает миллион! Поэтому при $p > 40$ отбор оптимального подмножества предикторов становится невозможным из–за громадного объема вычислений, даже если используются быстрые современные компьютеры. Существуют более экономные способы вычислений (так называемые методы «ветвей и границ»)[3], позволяющие исключать из рассмотрения некоторые модели, однако эти способы имеют свои ограничения при больших p. Кроме того, они работают только с линейной регрессией по методу наименьших квадратов. Далее мы рассмотрим более эффективные альтернативы отбору оптимального подмножества предикторов.

6.1.2 Пошаговый отбор

По причинам вычислительного характера, при очень большом p отбор оптимального подмножества предикторов неприменим. При большом p этот метод может также страдать от проблем статистического характера. Чем больше область поиска, тем выше вероятность нахождения моделей, которые хорошо предсказывают обучающие данные, несмотря на то что на будущих данных они могут не иметь никакой предсказательной силы. Поэтому огромная область поиска может приводить к переобучению и высокой дисперсии оценок коэффициентов.

По обеим этим причинам *пошаговые* методы, которые выполняют поиск среди намного меньшего числа моделей, являются привлекательными альтернативами отбору оптимального подмножества предикторов.

[2] В оригинале используется термин «deviance». Устойчивого перевода этого термина на русский язык не существует (помимо «аномальности», встречаются, например, «отклонение», «девианс», «девианса», «девиация»). В соответствии со словарем терминов, представленным на сайте Международного статистического института (ISI; http://isi.cbs.nl/glossary/term933.htm), здесь «deviance» переводится как «аномальность». — *Прим. пер.*

[3] В оригинале используется термин «branch–and–bound techniques». —*Прим. пер.*

Пошаговое включение переменных

пошаговое включение переменных

Пошаговое включение переменных является эффективной с вычислительной точки зрения альтернативой отбору оптимального подмножества. В то время как процедура отбора оптимального подмножества выполняет поиск среди 2^p возможных моделей, метод пошагового включения работает с гораздо меньшим набором моделей. Алгоритм пошагового включения начинает с модели, которая не содержит никаких переменных, а затем добавляет по одному предиктору в эту модель до тех пор, пока не будут включены все предикторы. В частности, на каждом шаге добавляется переменная, которая обеспечивает наибольшее *приращение* качества модели. Более формально процедура пошагового включения описана в алгоритме 6.2.

Алгоритм 6.2 Пошаговое включение переменных

1. Пусть $\mathcal{M}_0$ обозначает *нулевую модель*, которая не содержит предикторов.
2. Для $k = 0, \ldots, p-1$:
   (a) Постройте все $p-k$ моделей, которые увеличивают размер $\mathcal{M}_k$ путем добавления еще одного предиктора.
   (b) Выберите *оптимальную* из этих $p-k$ моделей и назовите ее $\mathcal{M}_{k+1}$. Здесь под оптимальной понимается модель с наименьшей RSS или наибольшей статистикой R^2.
3. Выберите единственную оптимальную модель среди $\mathcal{M}_0, \ldots, \mathcal{M}_p$, используя такие полученные при помощи перекрестной проверки показатели, как ошибка на контрольной выборке, C_p (AIC), BIC или скорректированный коэффициент детерминации R^2.

В отличие от метода отбора оптимального подмножества переменных, который предполагает построение 2^p моделей, метод пошагового включения переменных основан на подгонке одной нулевой модели и $p-k$ моделей на каждом k–м итеративном шаге (для $k = 0, \ldots, p-1$). В итоге получается $1 + \sum_{k=0}^{p-1}(p-k) = 1 + p(p+1)/2$ моделей. Это существенная разница: при $p = 20$ отбор оптимального подмножества требует подгонки 1 048 576 моделей, тогда как метод пошагового включения предикторов требует подгонки только 211 моделей[4].

На шаге 2(b) алгоритма 6.2 мы должны определить оптимальную среди $p-k$ моделей, увеличивающих размер $\mathcal{M}_k$ путем добавления еще одного предиктора. Мы можем сделать это, просто выбрав модель с наименьшей RSS или наибольшей статистикой R^2. Однако на шаге 3 мы должны выбрать единственную оптимальную модель среди нескольких моделей с разным числом предикторов. Это более сложная задача, и мы обсуждаем ее в подразделе 6.1.3.

[4] Хотя метод последовательного включения переменных рассматривает $p(p+1)/2+1$ моделей, он выполняет *направленный* поиск в пространстве моделей, и поэтому эффективная область поиска содержит намного больше, чем $p(p+1)/2+1$ моделей.

Вычислительное преимущество метода пошагового включения в сравнении с отбором оптимального подмножества переменных очевидно. Хотя пошаговое включение переменных обычно хорошо работает на практике, нет никаких гарантий, что оно приведет к нахождению наилучшей из всех 2^p возможных моделей. Представьте, например, что для некоторого набора данных с $p = 3$ наилучшая модель с одной переменной включает X_1, а наилучшая модель с двумя переменным включает X_2 и X_3. Процедура пошагового включения не сможет выбрать оптимальную модель с двумя переменными, поскольку $\mathcal{M}_1$ будет содержать X_1, и поэтому $\mathcal{M}_2$, помимо одной дополнительной переменной, также должна будет включать X_1.

Таблица 6.1, в которой представлены первые четыре модели, выбранные для данных `Credit` путем отбора оптимального подмножества переменных и при помощи пошагового включения переменных, иллюстрирует это явление. Оба метода выбирают `rating` для оптимальной модели с одной переменной, а затем добавляют `income` и `student` в модели с двумя и тремя переменными. Однако в модели с четырьмя переменными метод отбора оптимального подмножества переменных заменяет `rating` на `cards`, тогда как метод пошагового включения переменных вынужден в этой модели оставить `rating`. Как показано на рис. 6.1, для этого примера большой разницы по RSS между моделями с тремя и четырьмя предикторами нет, и поэтому любая из моделей с четырьмя предикторами, по–видимому, будет адекватной.

Пошаговое включение переменных можно применять даже для задач высокой размерности, где $n < p$; однако в таком случае можно построить только $\mathcal{M}_0, \ldots, \mathcal{M}_{n-1}$ моделей, поскольку каждая подмодель подгоняется по методу наименьших квадратов, который не способен давать уникальное решение при $p \geq n$.

ТАБЛИЦА 6.1. *Первые четыре модели для методов отбора оптимального подмножества переменных и пошагового включения переменных, полученные по данным* `Credit`. *Первые три модели идентичны, тогда как четвертые модели различаются*

Число предикторов	Отбор оптимального подмножества	Пошаговое включение
1	`rating`	`rating`
2	`rating, income`	`rating, income`
3	`rating, income, student`	`rating, income, student`
4	`cards, income, student, limit`	`rating, income, student, limit`

Пошаговое исключение переменных

пошаговое исключение переменных

Подобно пошаговому включению, *пошаговое исключение переменных* обеспечивает эффективную альтернативу методу отбора оптимального подмножества переменных. Однако, в отличие от пошагового включения переменных, этот метод начинает с полной модели, содержащей все p предикторов и подогнанной по методу наименьших квадратов, а затем последовательно, по одному за раз, удаляет наименее полезные предикторы. Более детальное описание представлено в алгоритме 6.3.

Подобно пошаговому включению переменных, метод пошагового исключения выполняет поиск только среди $1 + p(p + 1)/2$ моделей, и по-

Алгоритм 6.3 Пошаговое исключение переменных

1. Пусть $\mathcal{M}_p$ обозначает *полную* модель, которая содержит все предикторы.
2. Для $k = p, p-1, \ldots, 1$:
   (a) для $k-1$ предикторов постройте все k моделей, которые содержат все предикторы из $\mathcal{M}_k$, за исключением одного;
   (b) выберите *оптимальную* из этих k моделей и назовите ее $\mathcal{M}_{k-1}$. Здесь под оптимальной понимается модель, обладающая наименьшей RSS или наибольшей статистикой R^2.
3. Выберите единственную оптимальную модель среди $\mathcal{M}_0, \ldots, \mathcal{M}_p$, используя такие полученные при помощи перекрестной проверки показатели, как ошибка на контрольной выборке, C_p (AIC), BIC или скорректированный коэффициент детерминации R^2.

этому он применим в случаях, когда значение p слишком велико, чтобы использовать метод отбора оптимального подмножества переменных[5]. Кроме того, метод пошагового исключения также не может гарантировать нахождения оптимальной модели, включающей некоторое подмножество из исходных p предикторов.

Пошаговое исключение переменных требует, чтобы число наблюдений n превышало число переменных p (для обеспечения возможности подгонки полной модели). В то же время пошаговое включение переменных применимо даже при $n < p$ и поэтому является единственным действенным методом для отбора оптимальных предикторов при больших значениях p.

Гибридные методы

Методы отбора оптимального подмножества, пошагового включения и пошагового исключения переменных обычно дают похожие, хотя и не идентичные модели. Существуют также гибридные версии пошагового включения и исключения, в которых переменные последовательно добавляются в модель (по аналогии с методом пошагового включения). Однако после добавления каждой новой переменной любые переменные, которые больше не обеспечивают улучшения качества модели, могут быть удалены. Этот подход пытается близко подражать методу отбора оптимального подмножества переменных, сохраняя при этом вычислительные преимущества методов пошагового включения и пошагового исключения переменных.

6.1.3 Выбор оптимальной модели

Методы отбора оптимального подмножества, пошагового включения и пошагового исключения переменных приводят к созданию набора моделей,

[5] Аналогично пошаговому включению, метод пошагового исключения выполняет *направленный* поиск в пространстве моделей и поэтому в действительности рассматривает намного больше, чем $1 + p(p+1)/2$ моделей.

каждая из которых содержит определенное подмножество из исходных p предикторов. Для реализации этих методов нам необходимо выяснить, какая из этих моделей является *оптимальной*. Как было отмечено в подразделе 6.1.1, модель, включающая все предикторы, всегда будет обладать минимальной RSS и максимальным значением R^2, поскольку эти величины тесно связаны с ошибкой обучения. Однако мы хотим выбрать модель с наименьшей ошибкой на контрольной выборке. Ошибка на обучающей выборке может оказаться плохой оценкой ошибки на контрольной выборке (см. также главу 2). Следовательно, RSS и R^2 не подходят для выбора оптимальной модели среди нескольких моделей с разным числом входящих в них предикторов.

Для нахождения оптимальной модели, обеспечивающей минимальную ошибку на контрольной выборке, нам нужна оценка этой ошибки. Существуют два основных подхода:

1. Мы можем оценить ошибку на контрольных данных опосредованно, внося *поправку* в значение ошибки на обучающей выборке для учета смещения, которое возникает из–за переобучения.

2. Мы можем *непосредственно* оценить ошибку на контрольных данных, применив либо метод проверочной выборки, либо метод перекрестной проверки, как описано в главе 5.

Ниже мы рассмотрим оба этих подхода.

C_p, AIC, BIC и скорректированный коэффициент R^2

В главе 2 мы показали, что MSE, рассчитанная по обучающей выборке, обычно занижает MSE на контрольной выборке. (Вспомните, что $MSE = RSS/n$.) Это обусловлено тем, что при подгонке модели к обучающим данным по методу наименьших квадратов мы специально оцениваем регрессионные коэффициенты таким образом, чтобы RSS на обучающих (не контрольных!) данных была как можно меньшей. В частности, по мере добавления переменных в модель ошибка на обучающей выборке будет снижаться, тогда как ошибка на контрольной выборке не обязательно будет вести себя тем же образом. В связи с этим RSS и R^2, рассчитанные на обучающих данных, не могут быть использованы для отбора оптимального варианта из нескольких моделей с разным числом переменных.

Однако существует несколько методов, позволяющих скорректировать ошибку обучения с учетом размера модели. Эти методы можно использовать для отбора оптимального варианта из нескольких моделей с разным числом предикторов. Ниже мы рассмотрим четыре таких подхода: C_p, *информационный критерий Акаике* (AIC), *байесовский информационный критерий* (BIC) и *скорректированный коэффициент детерминации* R^2. На рис. 6.2 показаны C_p, BIC и R^2 для оптимальных моделей каждого размера, полученных по данным `Credit` методом отбора оптимального подмножества переменных.

C_p
AIC
BIC
скорректированный R^2

Для модели с d предикторами, подогнанной по методу наименьших квадратов, C_p–оценка среднеквадратичной ошибки на контрольной выборке вычисляется при помощи следующего уравнения:

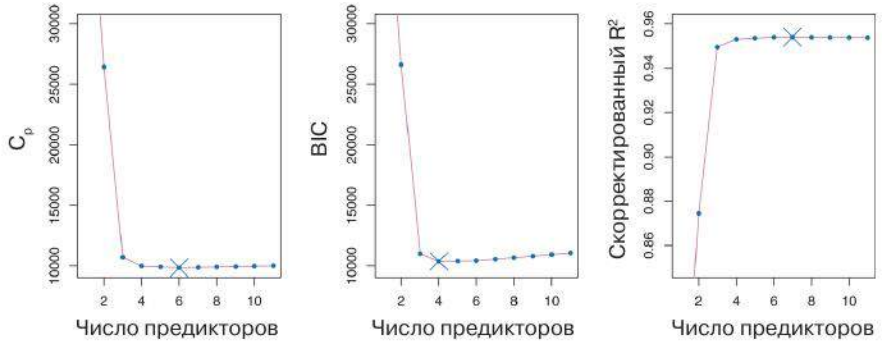

РИСУНОК 6.2. *C_p, BIC и скорректированный коэффициент детерминации R^2 для оптимальных моделей каждого размера, полученных по данным* `Credit` *(см. нижнюю границу на рис. 6.1). C_p и BIC являются оценками MSE на контрольной выборке. В центре рисунка мы видим, что BIC–оценка ошибки на контрольном множестве начинает возрастать после отбора четырех переменных. Остальные две кривые после четырех переменных довольно плоские*

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2), \tag{6.2}$$

где $\hat{\sigma}^2$ — это оценка дисперсии остатков ϵ, связанных с каждым значением отклика в (6.1)[6]. По сути, статистика C_p добавляет штраф величиной $2d\hat{\sigma}^2$ к RSS, рассчитанной по обучающей выборке, и тем самым учитывает свойство ошибки на обучающей выборке занижать ошибку на контрольной выборке. Очевидно, что этот штраф возрастает при увеличении числа входящих в модель предикторов: предполагается, что такой подход поможет скорректировать соответствующее увеличение RSS, полученной на обучающей выборке. Хотя это выходит за рамки данной книги, можно показать, что если $\hat{\sigma}^2$ из (6.2) является несмещенной оценкой σ^2, то C_p является несмещенной оценкой MSE на контрольной выборке. Как следствие статистика C_p обычно принимает некоторое малое значение для моделей с наименьшей ошибкой на контрольной выборке, и поэтому при отборе оптимальной модели из нескольких возможных мы выбираем модель с наименьшим значением C_p. На рис. 6.2 статистика C_p выбирает модель с шестью предикторами — `income`, `limit`, `rating`, `cards`, `age` и `student`. где $\hat{\sigma}^2$ — это оценка дисперсии остатков ϵ, связанных с каждым значением отклика в (6.1)[7]. По сути, статистика C_p добавляет штраф величиной $2d\hat{\sigma}^2$ к RSS, рассчитанной по обучающей выборке, и тем самым

[6] C_p–статистика Мэллоу иногда рассчитывается как $C_p' = \text{RSS}/\hat{\sigma}^2 + 2d - n$. Это определение эквивалентно приведенному выше в том смысле, что $C_p = \frac{1}{n}\hat{\sigma}^2(C_p' + n)$, и поэтому модель с наименьшим значением C_p также имеет наименьшее значение C_p'.

[7] C_p–статистика Мэллоу иногда рассчитывается как $C_p' = \text{RSS}/\hat{\sigma}^2 + 2d - n$. Это определение эквивалентно приведенному выше в том смысле, что $C_p = \frac{1}{n}\hat{\sigma}^2(C_p' + n)$, и поэтому модель с наименьшим значением C_p также имеет наименьшее значение C_p'.

учитывает свойство ошибки на обучающей выборке занижать ошибку на контрольной выборке. Очевидно, что этот штраф возрастает при увеличении числа входящих в модель предикторов: предполагается, что такой подход поможет скорректировать соответствующее увеличение RSS, полученной на обучающей выборке. Хотя это выходит за рамки данной книги, можно показать, что если $\hat{\sigma}^2$ из (6.2) является несмещенной оценкой σ^2, то C_p является несмещенной оценкой MSE на контрольной выборке. Как следствие статистика C_p обычно принимает некоторое малое значение для моделей с наименьшей ошибкой на контрольной выборке, и поэтому при отборе оптимальной модели из нескольких возможных мы выбираем модель с наименьшим значением C_p. На рис. 6.2 статистика C_p выбирает модель с шестью предикторами — `income`, `limit`, `rating`, `cards`, `age` и `student`.

Критерий AIC определен для большого класса моделей, которые подгоняются по методу максимального правдоподобия. В случае модели (6.1) с нормально распределенными остатками методы максимального правдоподобия и наименьших квадратов дают идентичные решения. В этом случае AIC рассчитывается как

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d\hat{\sigma}^2),$$

где для простоты мы опустили слагаемое–константу. Следовательно, для моделей, рассчитываемых по методу наименьших квадратов, C_p и AIC пропорциональны друг другу, и поэтому на рис. 6.2 показана только статистика C_p.

BIC получают на основе байесовской теории, но в конечном итоге этот критерий тоже похож на C_p (AIC). Для модели с d предикторами, рассчитываемой по методу наименьших квадратов, BIC (с точностью до не играющей роли константы) рассчитывается как

$$\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n)d\hat{\sigma}^2). \tag{6.3}$$

Подобно C_p, у модели с низкой ошибкой на контрольной выборке BIC обычно принимает некоторое малое значение, и поэтому мы обычно выбираем модель с наименьшим значением BIC. Заметьте, что в формуле BIC величина $2d\hat{\sigma}^2$ из формулы C_p заменяется на $\log(n)d\hat{\sigma}^2$, где n — это число наблюдений. Поскольку $\log n > 2$ для любого $n > 7$, статистика BIC обычно накладывает более высокий штраф на модели с большим числом предикторов, что приводит к выбору моделей меньшего размера, чем в случае с C_p. На рис. 6.2 мы видим, что для данных `Credit` это действительно так — BIC выбирает модель, содержащую только четыре предиктора (`income`, `limit`, `cards` и `student`). В данным случае кривые очень плоские и похоже, что точность предсказаний у моделей с четырьмя и с шестью предикторами отличается незначительно.

Скорректированный коэффициент детерминации R^2 — это еще один популярный критерий для выбора оптимального варианта среди моделей с несколькими предикторами. Вспомните из главы 3, что обычно R^2 вычисляют как $1 - \text{RSS}/\text{TSS}$, где $\text{TSS} = \sum(y_i - \bar{y})^2$ — это *общая сумма квадратов* для отклика. Поскольку по мере добавления переменных в модель

RSS всегда снижается, R^2 будет при этом всегда возрастать. Для модели с d предикторами, рассчитываемой по методу наименьших квадратов, скорректированный коэффициент R^2 вычисляется как

$$R^2_{adj.} = 1 - \frac{\text{RSS}/(n-d-1)}{TSS/(n-1)}. \tag{6.4}$$

В отличие от C_p, AIC и BIC, где на модель с низкой ошибкой на контрольной выборке указывают *низкие* значения, в случае со скорректированным коэффициентом R^2 на модель с низкой ошибкой на контрольной выборке указывают *высокие* значения. Максимизация скорректированного коэффициента R^2 эквивалентна минимизации $\frac{\text{RSS}}{n-d-1}$. В то время как RSS всегда снижается при увеличении числа переменных в модели, $\frac{\text{RSS}}{n-d-1}$ может увеличиваться или снижаться из–за наличия d в знаменателе.

Логика скорректированного коэффициента детерминации R^2 состоит в том, что как только в модель были добавлены все правильные переменные, включение дополнительных *шумовых* переменных будет приводить только к очень небольшому снижению RSS. Поскольку добавление шумовых переменных сопровождается увеличением d, такие переменные будут приводить к увеличению $\frac{\text{RSS}}{n-d-1}$, а следовательно, к снижению скорректированного коэффициента R^2. Поэтому теоретически модель с наибольшим скорректированным коэффициентом R^2 будет содержать только правильные переменные и никаких шумовых переменных. В отличие от простого коэффициента R^2, скорректированный коэффициент R^2 «*расплачивается*» за включение ненужных переменных в модель. На рис. 6.2 показан скорректированный коэффициент R^2 для набора данных `Credit`. Использование этой статистики приводит к выбору модели с 7 переменными за счет добавления `gender` к модели, выбранной на основе C_p и AIC.

C_p, AIC и BIC имеют строгие теоретические основания, обсуждение которых выходит за рамки этой книги. Эти основания строятся на асимптотических сценариях, т. е. ситуациях, когда объем выборки n очень большой. Несмотря на свою популярность и интуитивно понятную интерпретацию, скорректированный коэффициент детерминации R^2 не так хорошо обоснован в статистической теории, как AIC, BIC и C_p. Все эти критерии легко применять и вычислять. Здесь мы представили формулы AIC, BIC и C_p для случая линейной модели, подогнанной по методу наименьших квадратов, однако эти критерии можно определить также и для более общих типов моделей.

Методы проверочной выборки и перекрестной проверки

В качестве альтернативы только что описанным подходам мы можем непосредственно оценить ошибку на контрольной выборке при помощи методов проверочной выборки и перекрестной проверки, описанных в главе 5. Мы можем вычислить ошибку на контрольном множестве с помощью этих методов для каждой рассматриваемой модели, а затем выбрать модель с наименьшей оцененной ошибкой. Эта процедура имеет преимущества, по сравнению с AIC, BIC, C_p и скорректированным коэффициентом детерминации R^2, в том, что она дает прямую оценку ошибки на контрольном множестве и делает меньше предположений в отношении истинной модели. Ее можно применять также для более широкого круга задач по отбору

моделей, даже в случаях, когда бывает трудно определить число степеней модели (т. е. число предикторов в модели) и оценить дисперсию остатков σ^2.

В прошлом выполнение перекрестной проверки для многих задач с большим p и/или n было затруднительным с вычислительной точки зрения, и поэтому AIC, BIC, C_p и скорректированный коэффициент детерминации R^2 были более привлекательными методами для отбора оптимальной модели из нескольких возможных. Однако сегодня, когда у нас есть быстрые компьютеры, выполнение вычислений, необходимых для перекрестной проверки, едва ли является проблемой. Таким образом, перекрестная проверка — это очень привлекательный метод выбора оптимальной модели из нескольких рассматриваемых кандидатов.

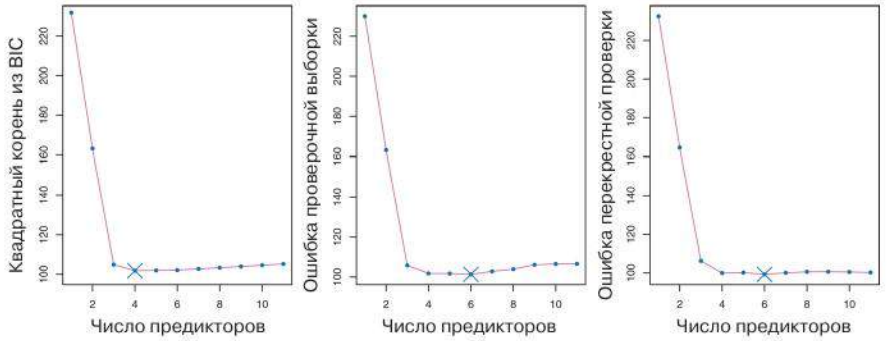

РИСУНОК 6.3. *Показаны три критерия качества для оптимальных моделей с d предикторами, рассчитанных по данным* `Credit`*, где d варьирует от 1 до 11. Оптимальная модель, определенная по каждому из этих критериев, отмечена голубым крестиком.* Слева: *квадратный корень из BIC.* В центре: *ошибки, полученные на основе метода проверочной выборки.* Справа: *ошибки, полученные методом перекрестной проверки*

На рис. 6.3 показано изменение BIC, а также ошибок на контрольной выборке, полученных методом проверочной выборки и перекрестной проверкой по данным `Credit` для оптимальных моделей с d предикторами. Ошибки на проверочных выборках были рассчитаны путем случайного распределения трех четвертей исходных наблюдений в обучающую выборку, а остальных наблюдений — в проверочную выборку. Для вычисления ошибок методом перекрестной проверки было использовано $k = 10$ блоков. В этом примере и метод проверочной выборки, и метод перекрестной проверки дают модель с шестью переменными. Однако все три подхода говорят о том, что модели с четырьмя, пятью и шестью переменными примерно эквивалентны по величине ошибок, допускаемых ими на контрольной выборке.

Действительно, кривые ошибок на контрольной выборке, показанные в центре и справа на рис. 6.3, довольно плоские. В то время как модель с тремя переменными четко демонстрирует меньшую ошибку на контрольных данных, чем модель с двумя переменными, ошибки у моделей с 3 и 11 переменными довольно похожи. Более того, если бы мы повторно при-

менили метод проверочной выборки с использованием другого разбиения данных на обучающую и проверочную выборки, или повторно применили перекрестную проверку с другим набором блоков, то модель, обеспечивающая наименьшую оценку ошибки на контрольной выборке, однозначно оказалась бы другой. При таком сценарии мы можем выбрать модель при помощи *правила одной стандартной ошибки*. Сначала мы вычисляем стандартную ошибку оцененной MSE на контрольной выборке для каждой модели, а затем выбираем модель с наименьшим числом переменных, у которой ошибка на контрольных данных находится в пределах одной стандартной ошибки от минимальной точки на кривой. Обоснование этого подхода состоит в том, что если несколько моделей обеспечивают примерно одинаково качество предсказаний, то нам стоило бы выбрать наиболее простую модель, т. е. модель с наименьшим числом предикторов. В рассматриваемом примере применение правила одной стандартной ошибки к методам проверочной выборки и перекрестной проверки приводит к выбору модели с тремя переменными.

правило одной стандартной ошибки

6.2 Методы сжатия

Описанные в разделе 6.1 методы отбора подмножества переменных предполагают применение метода наименьших квадратов для подгонки линейной модели с некоторым набором предикторов. В качестве альтернативы мы можем построить модель со всеми p предикторами при помощи метода, который выполняет *ограничение*, или *регуляризацию*, оценок коэффициентов, или, другими словами, *сжимает* оценки коэффициентов в направлении к нулю. На первый взгляд не очень ясно, почему такое ограничение может улучшить модель, но оказывается, что сжатие оценок коэффициентов может значительно снизить их дисперсию. Двумя наиболее хорошо известными методами сжатия регрессионных коэффициентов до нуля являются *гребневая регрессия* и *лассо*.

6.2.1 Гребневая регрессия

Вспомните из главы 3, что процедура подгонки модели по методу наименьших квадратов оценивает $\beta_0, \beta_1, \ldots, \beta_p$ путем минимизации

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

гребневая регрессия

Гребневая регрессия очень похожа на регрессию по методу наименьших квадратов, за тем исключением, что коэффициенты оцениваются путем минимизации несколько отличающейся величины. В частности, оценки коэффициентов гребневой регрессии $\hat{\beta}^R$ представляют собой значения, которые минимизируют

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2, \qquad (6.5)$$

где $\lambda \geq 0$ — это *гиперпараметр*, который находят отдельно. Уравнение 6.5 обеспечивает компромисс между двумя критериями. Как и в случае с методом наименьших квадратов, гребневая регрессия ищет оценки коэффициентов, которые приводят к хорошему описанию данных за счет минимизации RSS. Однако второй член — $\lambda \sum_j \beta_j^2$, который называют *штрафным слагаемым*[8], принимает малые значения, когда $\beta_1, \ldots, \beta_p$ близки к нулю, и поэтому оказывает эффект *сжатия* оценок β_j в направлении к нулю. Гиперпараметр λ служит для контроля относительного эффекта этих двух членов уравнения на оценки регрессионных коэффициентов. При $\lambda = 0$ штрафное слагаемое не имеет никакого эффекта, и гребневая регрессия даст те же оценки, что и метод наименьших квадратов. Однако при $\lambda \to \infty$ влияние штрафного слагаемого возрастает, и оценки коэффициентов гребневой регрессии будут приближаться к нулю. В отличие от метода наименьших квадратов, который дает только один набор коэффициентов, гребневая регрессия будет порождать отдельный набор оценок коэффициентов $\hat{\beta}^R$ для каждого значения λ. Выбор хорошего значения λ является критичным; мы отложим обсуждение этого вопроса до подраздела 6.2.3, где будет применена перекрестная проверка.

гиперпараметр

Заметьте, что штрафное слагаемое в (6.5) применяется к $\beta_1, \ldots, \beta_p$, но не к свободному члену β_0. Мы хотим сжать величину эффекта каждой переменной на отклик, но при этом нам нет необходимости сжимать свободный член, который попросту равен среднему значению отклика при $x_{i1} = x_{i2} = \cdots = x_{ip} = 0$. Если мы предположим, что переменные, т. е. столбцы матрицы данных $\mathbf{X}$, были стандартизованы до выполнения гребневой регрессии так, что их средние значения стали равными нулю, то свободный член примет форму $\hat{\beta}_0 = \bar{y} = \sum_{i=1}^{n} y_i / n$.

Применение к данным по кредитному долгу

На рис. 6.4 показаны оценки коэффициентов гребневой регрессии для данных `Credit`. На графике слева каждая кривая соответствует оценке коэффициента одной из десяти переменных в зависимости от λ. Например, черная сплошная линия показывает оценки коэффициента для переменной `income` при разных значениях λ. В крайней левой части этого графика параметр λ практически равен нулю, и поэтому соответствующие оценки коэффициентов гребневой регрессии совпадают с обычными оценками, полученными по методу наименьших квадратов. Однако по мере увеличения λ оценки коэффициентов гребневой регрессии уменьшаются до нуля. При очень больших значениях λ все оценки коэффициентов гребневой регрессии практически равны нулю — эта ситуация соответствует *нулевой модели*, не содержащей никаких предикторов. На этом графике переменные `income`, `limit`, `rating` и `student` показаны разными цветами, поскольку эти переменные в целом имеют наиболее высокие оценки коэффициентов. В то время как оценки большинства коэффициентов гребневой регрессии при увеличении λ обычно уменьшаются, коэффициенты отдельных переменных (например, `rating` и `income`) иногда при увеличении λ могут возрастать.

Справа на рис. 6.4 показаны те же оценки коэффициентов гребневой

[8] В оригинале используется термин «shrinkage penalty». — *Прим. пер.*

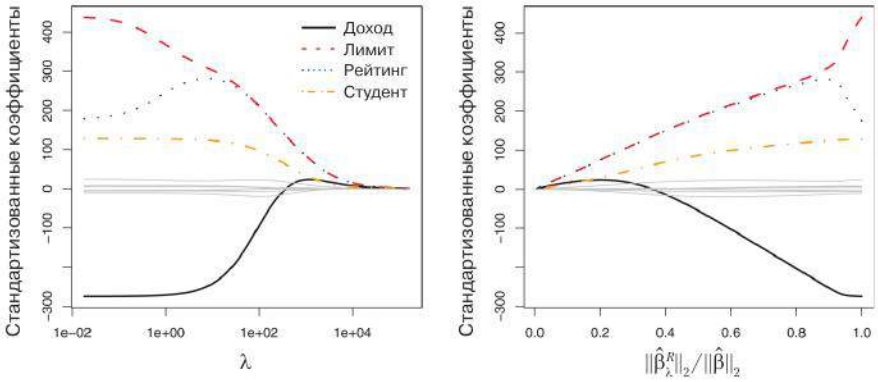

РИСУНОК 6.4. *Стандартизованные коэффициенты гребневой регрессии для набора данных* `Credit` *показаны в зависимости от* λ *и* $||\hat{\beta}^R_\lambda||_2/||\hat{\beta}||_2$

регрессии, что и на графике слева, однако вместо λ здесь по оси X мы откладываем значения $||\hat{\beta}^R_\lambda||_2/||\hat{\beta}||_2$, где $\hat{\beta}$ обозначает вектор коэффициентов, оцененных по методу наименьших квадратов. Нотация $||\beta||_2$ обозначает норму ℓ_2 (произносится как «эль два»)[9] некоторого вектора и имеет форму $||\beta||_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$. Эта величина показывает, как далеко β находится от нуля. При увеличении λ норма ℓ_2 вектора $\hat{\beta}^R_\lambda$, а следовательно, и $||\hat{\beta}^R_\lambda||_2/||\hat{\beta}||_2$ *всегда* будут снижаться. Последняя величина изменяется от 1 (при $\lambda = 0$, когда оценки коэффициентов, полученные методом гребневой регрессии и методом наименьших квадратов, совпадают, в связи с чем их нормы ℓ_2 одинаковы) до 0 (при $\lambda = \infty$, когда оценки коэффициентов гребневой регрессии представлены вектором, состоящим из нулей, и норма ℓ_2 равна нулю). Следовательно, ось X справа на рис. 6.4 можно интерпретировать как степень сжатия оценок коэффициентов гребневой регрессии в направлении к нулю; при этом малое значение указывает на то, что эти оценки были сжаты почти до нуля.

норма ℓ_2

Обсуждавшиеся в главе 3 обычные оценки коэффициентов, которые получают методом наименьших квадратов, являются *эквивариантными*[10]: умножение X_j на некоторую константу c просто приводит к пропорциональному снижению этих оценок коэффициентов в c раз. Другими словами, вне зависимости от шкалы измерения j–го предиктора величина $X_j\hat{\beta}_j$ останется неизменной. В то же время оценки коэффициентов гребневой регрессии могут *существенно* изменяться при умножении того или иного предиктора на некоторую константу. Рассмотрим, например, переменную `income`, которая выражается в долларах. Вполне резонно можно было бы измерять доход в тысячах долларов, что вызвало бы снижение наблюденных значений `income` в 1000 раз. Из–за наличия суммы квадратов коэффициентов в формуле гребневой регрессии (6.5) такое изменение не привело бы к простому снижению оценки коэффициента `income`

[9] Эта величина известна также под названиями «метрика L2» и «евклидова норма». — *Прим. пер.*

[10] В оригинале используется термин «scale equivariant». — *Прим. пер.*

в 1000 раз. Иными словами, $X_j\hat{\beta}^R_{j,\lambda}$ будет зависеть не только от значения λ, но и от шкалы измерения j–го предиктора. Более того, значение $X_j\hat{\beta}^R_{j,\lambda}$ может зависеть даже от шкалы измерения *других* предикторов! Поэтому лучше всего применять гребневую регрессию после *стандартизации предикторов* по формуле

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}}, \tag{6.6}$$

в результате которой все предикторы приводятся к одной шкале измерения. Знаменатель в (6.6) представляет собой выборочное стандартное отклонение j–го предиктора. Следовательно, у всех стандартизованных предикторов стандартное отклонение будет равно 1. Благодаря этому итоговая модель не будет зависеть от шкалы изменения отдельных предикторов. На рис. 6.4 по оси Y показаны стандартизованные оценки коэффициентов гребневой регрессии, т. е. оценки коэффициентов, полученные путем подгонки гребневой регрессии на основе стандартизованных предикторов.

Почему гребневая регрессия улучшает качество модели?

Основная причина преимущества гребневой регрессии над методом наименьших квадратов заключается в обеспечении более оптимального *баланса между смещением и дисперсией*. При увеличении λ гибкость гребневой регрессии снижается, приводя при этом к более низкой дисперсии, но более высокому смещению. Это показано слева на рис. 6.5 на примере имитированного набора данных с $p = 45$ предикторов и $n = 50$ наблюдений. Зеленая кривая на этом графике соответствует дисперсии модельных значений гребневой регрессии в зависимости от λ. При $\lambda = 0$, что совпадает с оценками коэффициентов по методу наименьших квадратов, дисперсия высока, но смещение отсутствует. Однако сжатие оценок коэффициентов гребневой регрессии при увеличении λ приводит к существенному снижению дисперсии предсказанных значений за счет лишь незначительного роста смещения. Вспомните, что MSE на контрольной выборке (показана в виде фиолетовой линии) является функцией от дисперсии и квадрата уровня смещения. Для значений λ вплоть до 10 дисперсия быстро снижается, что сопровождается очень небольшим увеличением смещения (черная кривая). Как следствие MSE значительно снижается при увеличении λ с от 0 до 10. За пределами этой точки снижение дисперсии по мере увеличения λ замедляется, и сжатие коэффициентов вызывает их значительную недооценку, что приводит к заметному увеличению смещения. Минимум MSE достигается примерно при $\lambda = 30$. Интересно, что из–за своей высокой дисперсии MSE у модели, рассчитанной по методу наименьших квадратов (при $\lambda = 0$), оказывается практически такой же высокой, как и MSE у нулевой модели, в которой оценки всех коэффициентов равны нулю (при $\lambda = \infty$). Однако для промежуточных значений λ MSE намного меньше.

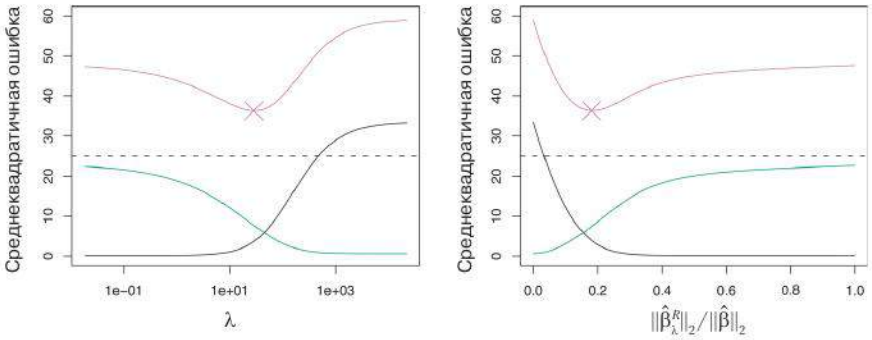

РИСУНОК 6.5. *Изменение квадрата смещения (черная кривая), дисперсии (голубая кривая) и среднеквадратичной ошибки на контрольной выборке (фиолетовая кривая) для значений, предсказанных гребневой регрессией по имитированным данным, в зависимости от* λ *и* $||\hat{\beta}^R_\lambda||_2/||\hat{\beta}||_2$. *Горизонтальная прерывистая линия показывает минимально возможную MSE. Фиолетовые крестики отмечают гребневые регрессионные модели с наименьшей MSE*

Справа на рис. 6.5 показаны те же кривые, что и на графике слева, но на этот раз ось X соответствует отношению ℓ_2–нормы оценок коэффициентов гребневой регрессии к ℓ_2–норме оценок, полученных по методу наименьших квадратов. При увеличении этого отношения модели становятся более гибкими, и поэтому смещение и дисперсия возрастают.

Когда связь между откликом и предикторами близка к линейной, оценки по методу наименьших квадратов обычно будут обладать низким смещением, но при этом могут иметь высокую дисперсию. Это означает, что небольшое изменение в обучающих данных может вызвать заметное изменение подобных оценок коэффициентов. В частности, когда число переменных p почти равно числу наблюдений n (как в примере на рис. 6.5), оценки по методу наименьших квадратов будут чрезвычайно изменчивыми. При $p > n$ оценки по методу наименьших квадратов не имеют уникального решения, тогда как метод гребневой регрессии по–прежнему способен дать качественное решение за счет значительного снижения дисперсии при одновременном небольшом увеличении смещения. Следовательно, гребневая регрессия работает лучше всего в ситуациях, когда оценки по методу наименьших квадратов обладают высокой дисперсией.

В сравнении с методом отбора оптимального подмножества переменных, который требует поиска среди 2^p моделей, гребневая регрессия обладает также существенными вычислительными преимуществами. Как мы обсуждали ранее, даже для умеренных значений p такой поиск может оказаться неосуществимым с вычислительной точки зрения. В то же время для любого фиксированного значения λ метод гребневой регрессии подгоняет единственную модель, и эта процедура может быть выполнена очень быстро. Более того, можно показать, что вычисления, необходимые для решения (6.5) *одновременно для всех значений* λ, почти идентичны вычислениям, выполняемым при подгонке модели по методу наименьших квадратов.

6.2.2 Лассо

У гребневой регрессии есть один очевидный недостаток. В отличие от методов отбора оптимального подмножества, пошагового включения и пошагового исключения переменных, которые обычно выбирают модели с некоторым ограниченным подмножеством переменных, гребневая регрессия будет включать в итоговую модель все p предикторов. Штрафное слагаемое $\lambda \sum \beta_j^2$ в (6.5) будет сжимать все коэффициенты в направлении к нулю, но никогда не приравняет ни один из этих коэффициентов в точности к нулю (если только параметр λ не равен ∞). Это не обязательно является проблемой с точки зрения точности предсказаний, однако данное обстоятельство может затруднить интерпретацию модели в случаях, когда число предикторов p довольно велико. Похоже, например, что в случае с набором данных `Credit` наиболее важными переменными являются `income`, `limit`, `rating` и `student`. В связи с этим мы, возможно, хотели бы построить модель, которая включает только эти предикторы. Однако гребневая регрессия всегда будет давать модель со всеми десятью предикторами. Увеличение значения λ обычно будет сопровождаться снижением коэффициентов, но не приведет к полному исключению какой–либо из переменных.

Метод *лассо* является относительно недавно разработанной альтернативой гребневой регрессии и преодолевает этот недостаток. Лассо–коэффициенты $\hat{\beta}_\lambda^L$ минимизируют величину

лассо

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|. \qquad (6.7)$$

Сравнивая (6.7) и (6.5), мы видим, что лассо и гребневая регрессия имеют похожие определения. Единственное отличие состоит в том, что член β_j^2 в штрафном слагаемом гребневой регрессии (6.5) заменен на $|\beta_j|$ в штрафном слагаемом лассо (6.7). На языке статистики мы говорим, что лассо использует ℓ_1–штраф (произносится как «эль один») вместо ℓ_2–штрафа. Норма ℓ_1 вектора коэффициентов β определяется как $\|\beta\| = \sum|\beta_j|$.

Подобно гребневой регрессии, лассо сжимает оценки коэффициентов в направлении к нулю. Однако в случае с лассо ℓ_1–штраф при достаточно большом параметре λ делает некоторые оценки коэффициентов в точности равными нулю. Следовательно, аналогично методу отбора оптимального подмножества переменных, метод лассо выполняет *отбор переменных*. В результате этого модели, полученные при помощи метода лассо, обычно бывает легче интерпретировать, чем модели, полученные при помощи гребневой регрессии. Мы говорим, что лассо дает *разреженные* модели[11], т. е. модели, которые включают только некоторое ограниченное подмножество переменных. Как и в случае с гребневой регрессией, выбор подходящего значения λ для лассо является критичным; мы отложим обсуждение этого вопроса до подраздела 6.2.3, где будет применена перекрестная проверка.

В качестве примера рассмотрим графики коэффициентов на рис. 6.6, полученные в результате применения метода лассо к данным `Credit`. При

[11] В оригинале используется термин «sparse models». — *Прим. пер.*

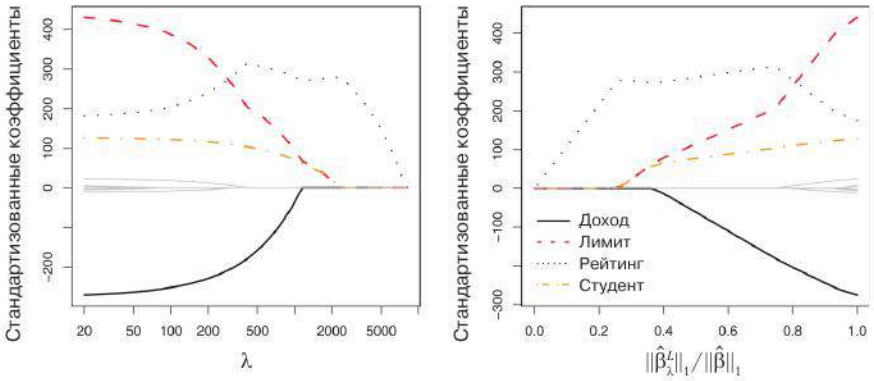

РИСУНОК 6.6. *Зависимость стандартизованных коэффициентов лассо–регрессии для данных* `Credit` *от* λ *и* $||\hat{\beta}^L_\lambda||_1/||\hat{\beta}||_1$

$\lambda = 0$ метод лассо дает ту же модель, что и метод наименьших квадратов, а при достаточно большом значении λ метод лассо дает нулевую модель, в которой все коэффициенты равны нулю. Однако, за исключением этих двух экстремальных случаев, гребневая регрессия и лассо–регрессия довольно сильно различаются. Справа на рис. 6.6 мы видим, что при увеличении значений на оси X метод лассо сначала приводит к модели, которая содержит только один предиктор — `rating`. Далее в модель почти одновременно входят `student` и `limit`, а вскоре и переменная `income`. Постепенно в модель попадают и другие переменные. Следовательно, в зависимости от значения λ метод лассо может привести к созданию модели с любым числом переменных. В то же время гребневая регрессия всегда включает в модель все переменные, хотя значения оценок коэффициентов в ней будут зависеть от λ.

Альтернативная формулировка методов гребневой регрессии и лассо

Можно показать, что оценки коэффициентов лассо–модели и гребневой регрессии решают следующие оптимизационные проблемы:

$$\underset{\beta}{\text{минимизировать}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \text{ при условии, что } \sum_{j=1}^{p} |\beta_j| \leq s \tag{6.8}$$

и

$$\underset{\beta}{\text{минимизировать}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \text{ при условии, что } \sum_{j=1}^{p} \beta_j^2 \leq s \tag{6.9}$$

соответственно. Другими словами, для каждого значения λ имеется такая константа s, при которой уравнения (6.7) и (6.8) дадут одинаковые лассо–оценки коэффициентов. Аналогичным образом для каждого значения λ имеется такая константа s, при которой уравнения (6.5) и (6.9) дадут одинаковые оценки коэффициентов гребневой регрессии. Как следует из (6.8), при $p = 2$ лассо–оценки коэффициентов дадут наименьшее из всех возможных значений RSS, ограниченных ромбом, стороны которого заданы неравенством $|\beta_1| + |\beta_2| \leq s$. Аналогично оценки коэффициентов гребневой регрессии характеризуются наименьшей RSS из всех точек в пределах круга, заданного неравенством $\beta_1^2 + \beta_2^2 \leq s$.

Мы можем думать о (6.8) следующим образом. Применяя метод лассо, мы пытаемся найти такой набор оценок коэффициентов, который приводит к наименьшей RSS при условии существования некоторого *бюджета* s, который задает максимальное значение величины $\sum_{j=1}^{p} |\beta_j|$. При высоком значении s этот бюджет не является ограничительным, и поэтому оценки коэффициентов могут быть большими. Более того, при таком высоком значении s, которое позволяет методу наименьших квадратов «уложиться» в бюджет, выражение (6.8) даст решение по методу наименьших квадратов. Однако чтобы не превысить бюджет при низком s, величина $\sum_{j=1}^{p} |\beta_j|$ также должна быть низкой. Аналогичным образом из (6.9) следует, что в ходе выполнения гребневой регрессии мы ищем набор таких оценок коэффициентов, которые минимизируют RSS с условием, что $\sum_{j=1}^{p} \beta_j^2$ не превышает бюджет s.

Формулировки (6.8) и (6.9) выявляют тесную связь между лассо, гребневой регрессией и методом отбора оптимального подмножества переменных. Рассмотрим следующую оптимизационную проблему:

$$\begin{aligned} &\underset{\beta}{\text{минимизировать}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \\ &\text{при условии, что } \sum_{j=1}^{p} I(\beta_j \neq 0) \leq s. \end{aligned} \tag{6.10}$$

Здесь $I(\beta_j \neq 0)$ представляет собой индикаторную переменную — она принимает значение 1 при $\beta_j \neq 0$ и 0 в других случаях. Тогда (6.10) сводится к нахождению таких оценок коэффициентов, которые минимизируют RSS при условии, что ненулевыми могут быть не более s коэффициентов. Проблема (6.10) эквивалентна методу отбора оптимального подмножества переменных. К сожалению, при большом p решение (6.10) невозможно по причинам вычислительного характера, поскольку оно требует рассмотрения всех $\binom{p}{s}$ моделей с s предикторами. Следовательно, мы можем интерпретировать гребневую регрессию и метод лассо как осуществимые с вычислительной точки зрения альтернативы методу отбора оптимального подмножества переменных, которые заменяют труднорешаемую форму бюджета в (6.10) на более легко решаемые формы. Конечно, метод лассо значительно теснее связан с методом отбора оптимального подмножества переменных, поскольку только лассо выполняет отбор переменных при достаточно малом значении s в (6.8).

Свойство, позволяющее лассо выполнять отбор переменных

Почему же метод лассо, в отличие от гребневой регрессии, приводит к тому, что оценки некоторых коэффициентов в точности раны нулю? Чтобы пролить свет на этот вопрос, можно воспользоваться выражениями (6.8) и (6.9). Рисунок 6.7 иллюстрирует эту ситуацию. Решение, полученное с помощью метода наименьших квадратов, отмечено как $\hat{\beta}$, тогда как залитые голубым цветом ромб и круг показывают области ограничений лассо и гребневой регрессии из (6.8) и (6.9) соответственно. Если значение s достаточно велико, то области ограничений будут включать $\hat{\beta}$, в связи с чем оценки лассо и гребневой регрессии будут совпадать с оценками метода наименьших квадратов. (Подобное высокое значение s соответствует $\lambda = 0$ в (6.5) и (6.7).) Однако на рис. 6.7 оценки, полученные по методу наименьших квадратов, лежат за пределами ромба и круга и поэтому не совпадают с оценками лассо и гребневой регрессии.

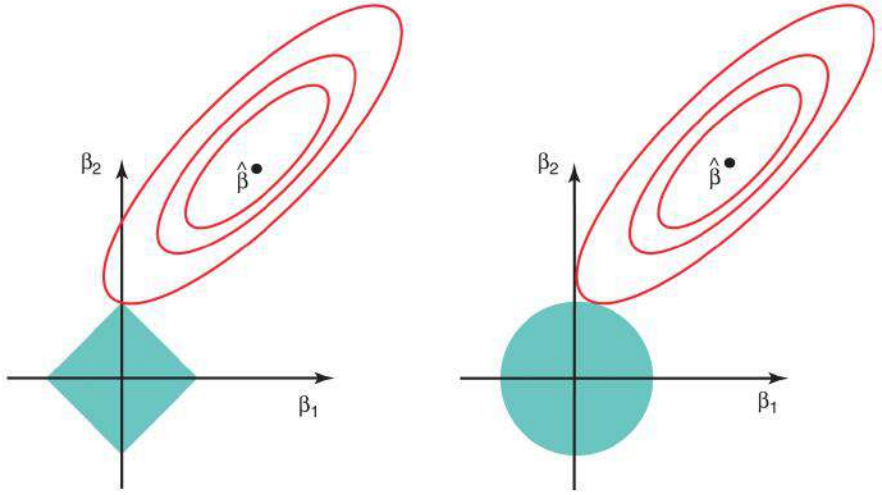

РИСУНОК 6.7. *Контурные диаграммы функций ошибок и ограничений для лассо-регрессии (слева) и гребневой регрессии (справа). Залитые голубым цветом фигуры представляют собой области ограничений* $|\beta_1| + |\beta_2| \leq s$ *и* $\beta_1^2 + \beta_2^2 \leq s$, *а красные эллипсы — контуры RSS*

Эллипсы, чьи центры соответствуют $\hat{\beta}$, представляют собой области постоянных значений RSS. Иными словами, все точки, лежащие на том или ином эллипсе, характеризуются одинаковым значением RSS. По мере удаления эллипсов от оценок, полученных методом наименьших квадратов, RSS возрастает. Уравнения (6.8) и (6.9) показывают, что оценки коэффициентов, полученные методами лассо и гребневой регрессии, соответствуют первой точке, в которой эллипс касается области ограничений. Поскольку область ограничений у гребневой регрессии имеет круглую форму без острых углов, эта точка касания обычно не приходится на координатную ось, и поэтому оценки коэффициентов гребневой регрессии всегда будут отличными от нуля. Однако область ограничений лассо име-

ет *углы* на каждой из осей, в связи с чем эллипс часто будет пересекать область ограничений на той или иной оси. Когда это происходит, один из коэффициентов принимает нулевое значение. В задачах более высокой размерности нулю могут быть равны одновременно несколько оценок коэффициентов. На рис. 6.7 пересечение происходит при $\beta_1 = 0$, и поэтому итоговая модель включает только β_2.

На рис. 6.7 мы рассмотрели простой случай с $p = 2$. При $p = 3$ область ограничений гребневой регрессии превращается в сферу, а область ограничений лассо становится полиэдром. Когда $p > 3$, область ограничений гребневой регрессии становится гиперсферой, а область ограничений лассо–регрессии — политопом. Однако ключевая идея, изображенная на рис. 6.7, остается той же. В частности, метод лассо выполняет отбор переменных при $p > 2$ благодаря наличию острых углов у полиэдра или политопа.

Сравнение методов лассо и гребневой регрессии

Очевидно, что метод лассо обладает существенным преимуществом, по сравнению с гребневой регрессией, которое заключается в порождении более простых и легче интерпретируемых моделей с ограниченным набором предикторов. Но какой из этих методов приводит к более высокой точности предсказаний? На рис. 6.8 показаны дисперсия, квадрат смещения и MSE на контрольной выборке для лассо–регрессии, примененной к имитированным данным из рис. 6.5. Хорошо видно, что метод лассо по своему поведению в целом похож на гребневую регрессию в том, что при увеличении λ происходят снижение дисперсии и рост смещения. На рис. 6.8 справа прерывистыми линиями показаны модели гребневой регрессии. Здесь мы упорядочиваем модели, подогнанные с помощью этих двух методов, в соответствии со значениями R^2. Это еще один полезный способ ранжирования моделей с разными типами регуляризации для последующего их сравнения, как в данном примере. В этом примере методы лассо и гребневой регрессии приводят к почти идентичным величинам смещения. Однако дисперсия гребневой регрессии несколько ниже дисперсии лассо. Как следствие минимум MSE у гребневой регрессии несколько ниже, чем у лассо.

Однако на рис. 6.8 данные были сгенерированы таким образом, чтобы все 45 предикторов коррелировали с откликом, т. е. ни один из истинных коэффициентов $\beta_1, \ldots, \beta_{45}$ не был равен нулю. Метод лассо неявным образом предполагает, что некоторые коэффициенты в действительности равны нулю. Поэтому не удивительно, что в данном случае гребневая регрессия превосходит лассо по ошибке предсказаний. На рис. 6.9 показана похожая ситуация, однако теперь отклик является функцией лишь 2 из 45 предикторов. Судя по уровням смещения, дисперсии и MSE, здесь метод лассо в целом срабатывает лучше гребневой регрессии.

Эти два примера показывают, что ни один из методов не превосходит другой во всех возможных случаях. В целом можно ожидать, что лассо будет работать лучше в ситуациях, когда относительно небольшая группа предикторов имеет высокие коэффициенты, а коэффициенты остальных предикторов очень малы или близки к нулю. Гребневая регрессия будет работать лучше, когда отклик является функцией многих предикторов,

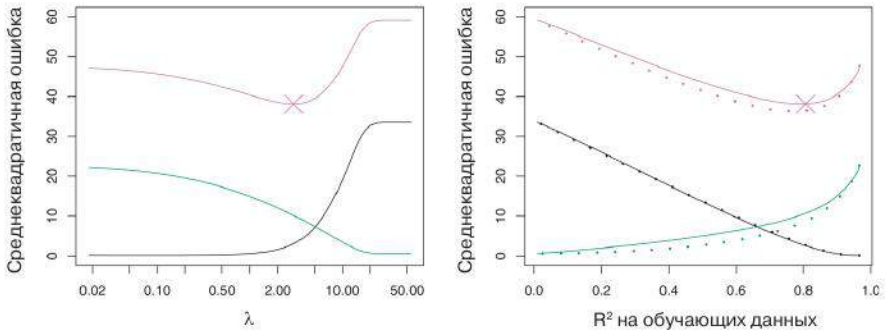

РИСУНОК 6.8. Слева: *кривые квадрата смещения (черная кривая), дисперсии (зеленая кривая) и MSE на контрольной выборке (фиолетовая кривая) для лассо–регрессии, подогнанной к имитированным данным.* Справа: *сравнение методов лассо (сплошные линии) и гребневой регрессии (прерывистые линии) по квадрату смещения, дисперсии и MSE на контрольном множестве. Модели, подогнанные при помощи обоих этих методов, упорядочены в соответствии с их значениями R^2 на обучающих данных (это распространенная форма ранжирования моделей для сравнения их качества). На обоих графиках крестики показывают лассо–модель с наименьшей MSE*

чьи коэффициенты примерно равны. Однако для реальных данных число связанных с откликом предикторов никогда не известно заранее. Чтобы найти оптимальный метод для того или иного набора данных, можно использовать такой подход, как перекрестная проверка.

Когда метод наименьших квадратов дает оценки коэффициентов с излишне высокой дисперсией, методы гребневой регрессии и лассо могут снижать дисперсию за счет небольшого увеличения смещения и, как следствие, давать более точные предсказания. Однако, в отличие от гребневой регрессии, лассо выполняет отбор переменных и поэтому приводит к легче интерпретируемым моделям.

Существуют очень эффективные алгоритмы подгонки гребневых и лассо–моделей; в обоих случаях полные пути эволюции коэффициентов можно вычислить с затратой тех же ресурсов, которые требуются для подгонки одной модели по методу наименьших квадратов. Мы рассмотрим это подробнее в лабораторной работе в конце данной главы.

Простой частный случай для гребневой регрессии и лассо

Чтобы лучше понять поведение гребневой регрессии и лассо, рассмотрим простой частный случай, в котором $n = p$, а $\mathbf{X}$ — диагональная матрица с единицами на диагонали и нулями во всех остальных элементах. Для дальнейшего упрощения проблемы предположим также, что мы строим регрессионную модель без свободного члена. С учетом этих предположений метод наименьших квадратов сводится к нахождению $\beta_1, \ldots, \beta_p$, которые минимизируют

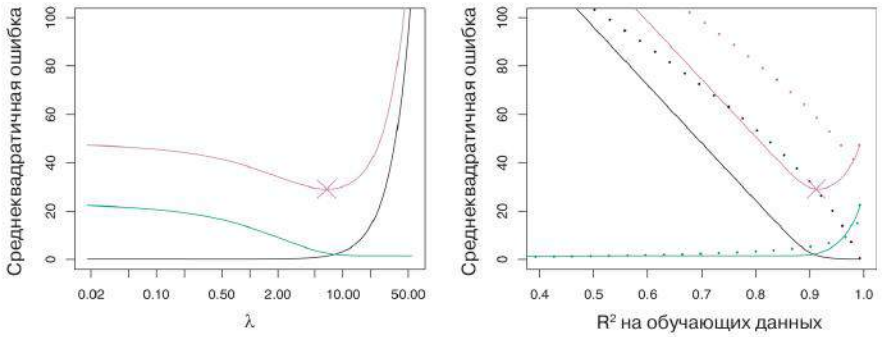

РИСУНОК 6.9. Слева: *кривые квадрата смещения (черная кривая), дисперсии (зеленая кривая) и MSE на контрольной выборке (фиолетовая кривая) для лассо–регрессии. Имитированные данные похожи на данные из рис. 6.8, однако здесь с откликом связаны только два предиктора.* Справа: *сравнение методов лассо (сплошные линии) и гребневой регрессии (прерывистые линии) по квадрату смещения, дисперсии и MSE на контрольной выборке. Модели, подогнанные при помощи этих двух методов, упорядочены в соответствии с их значениями R^2 на обучающей выборке (это распространенный способ ранжирования моделей для сравнения их качества). На обоих графиках крестики показывают лассо–модель с наименьшей MSE*

$$\sum_{j=1}^{p}(y_j - \beta_j)^2. \tag{6.11}$$

В данном случае метод наименьших квадратов дает следующее решение:

$$\hat{\beta} = y_i.$$

В свою очередь, решение по методу гребневой регрессии сводится к нахождению таких $\beta_1, \ldots, \beta_p$, которые минимизируют

$$\sum_{j=1}^{p}(y_j - \beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2, \tag{6.12}$$

а решение по методу лассо — к нахождению коэффициентов, которые минимизируют

$$\sum_{j=1}^{p}(y_j - \beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|. \tag{6.13}$$

Можно показать, что при таком сценарии оценки по методу гребневой регрессии принимают форму

$$\hat{\beta}_j^R = y_j/(1+\lambda), \tag{6.14}$$

а оценки по методу лассо принимают форму

$$\hat{\beta}_j^R = \begin{cases} y_j - \lambda/2 & \text{при } y_j > \lambda/2; \\ y_j + \lambda/2 & \text{при } y_j < -\lambda/2; \\ 0 & \text{при } |y_j| \leq \lambda/2. \end{cases} \tag{6.15}$$

Эта ситуация показана на рис. 6.10. Видно, что гребневая регрессия и лассо–регрессия выполняют два очень разных типа регуляризации. В гребневой регрессии все оценки коэффициентов, полученные по методу наименьших квадратов, сжимаются пропорционально своей величине в одинаковой степени. В то же время лассо сжимает каждый такой коэффициент в направлении к нулю на некоторую константу — $\lambda/2$; коэффициенты, чье абсолютное значение меньше $\lambda/2$, полностью сжимаются до нуля. Тип сжатия (6.15), выполняемого методом лассо в этом простом примере, известен как *мягкая регулировка порога*[12]. Тот факт, что некоторые лассо–коэффициенты полностью сжимаются до нуля, объясняет, почему метод лассо выполняет отбор переменных.

мягкая регулировка порога

В случае более общей структуры матрицы $\mathbf{X}$ ситуация несколько сложнее показанной на рис. 6.10, однако основная идея остается примерно такой же: гребневая регрессия пропорционально сжимает каждый коэффициент в одинаковой степени, тогда как лассо сжимает все коэффициенты в направлении к нулю примерно на одинаковую величину, а достаточно низкие коэффициенты полностью приравниваются нулю.

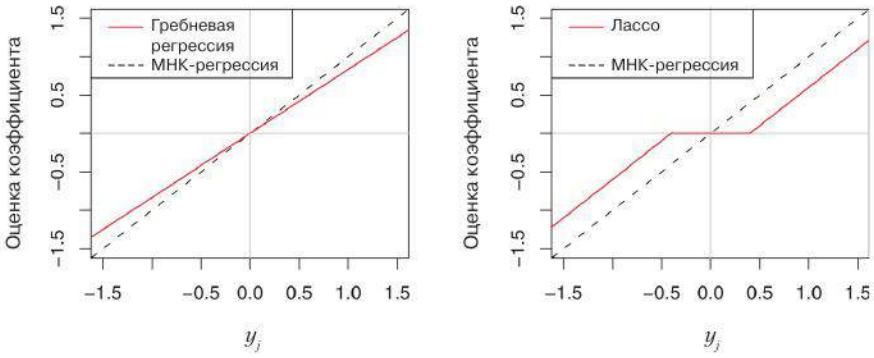

РИСУНОК 6.10. *Оценки коэффициентов гребневой регрессии и лассо–регрессии для простого случая, где $n = p$, а $\mathbf{X}$ — диагональная матрица с единицами по диагонали.* Слева: *в отличие от оценок по методу наименьших квадратов, оценки коэффициентов гребневой регрессии сжимаются до нуля пропорционально.* Справа: *лассо–оценки коэффициентов сжимаются до нуля с использованием мягкой регулировки порога*

Байесовская интерпретация гребневой регрессии и лассо

Теперь мы покажем, что методы гребневой регрессии и лассо можно рассматривать с точки зрения байесовской статистики. Эта точка

[12] В оригинале используется термин «soft–thresholding». — *Прим. пер.*

зрения в отношении регрессии предполагает, что вектор коэффициентов β имеет некоторое *априорное* распределение, например $p(\beta)$, где $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$. Правдоподобие данных можно представить как $f(Y|X, \beta)$, где $X = (X_1, \ldots, X_p)$. Умножение априорного распределения на правдоподобие дает нам (с точностью до некоторой константы) *апостериорное распределение* следующего вида:

апостериорное распределение

$$p(\beta|X, Y) \propto f(Y|X, \beta)p(\beta|X) = f(Y|X, \beta)p(\beta),$$

где пропорциональность вытекает из теоремы Байеса, а равенство — из допущения о том, что вектор X является фиксированным.

Мы принимаем обычную линейную модель

$$Y = \beta_0 + X_1\beta_1 + \cdots + X_p\beta_p + \epsilon$$

и предполагаем, что остатки независимы и происходят из нормального распределения. Допустим также, что $p(\beta) = \prod_{j=1}^{p} g(\beta_j)$ для некоторой функции плотности вероятности g. Оказывается, что методы гребневой регрессии и лассо естественным образом вытекают из двух частных случаев g:

- Если g представлена гауссовым распределением со средним значением, равным единице, и стандартным отклонением, равным нулю, то решение по методу гребневой регрессии дает *моду апостериорного распределения* β, т. е. наиболее вероятное значение β для имеющихся данных. (Кроме того, решение, получаемое при помощи гребневой регрессии, совпадает со средним значением этого апостериорного распределения.)

апостериорная мода

- Если g представлена двойным экспоненциальным (лапласовским) распределением со средним значением, равным нулю, и параметром масштаба, который является функцией от λ, то моду апостериорного распределения дает метод лассо. (Однако решение лассо *не совпадает* с апостериорным средним значением, и, более того, апостериорное среднее значение не порождает разреженного вектора коэффициентов.)

Гауссово и двойное экспоненциальное априорные распределения показаны на рис. 6.11. Таким образом, с точки зрения байесовской статистики методы гребневой регрессии и лассо непосредственно вытекают из допущения обычной линейной модели с нормально распределенными остатками наряду с допущением о некотором простом априорном распределении для β. Обратите внимание на то, что априорное распределение лассо имеет острый пик в области нуля, тогда как гауссово распределение становится все более и более плоским в окрестности нуля. Следовательно, метод лассо заведомо ожидает, что многие из коэффициентов в точности равны нулю, тогда как гребневая регрессия предполагает, что коэффициенты случайным образом распределены около нуля.

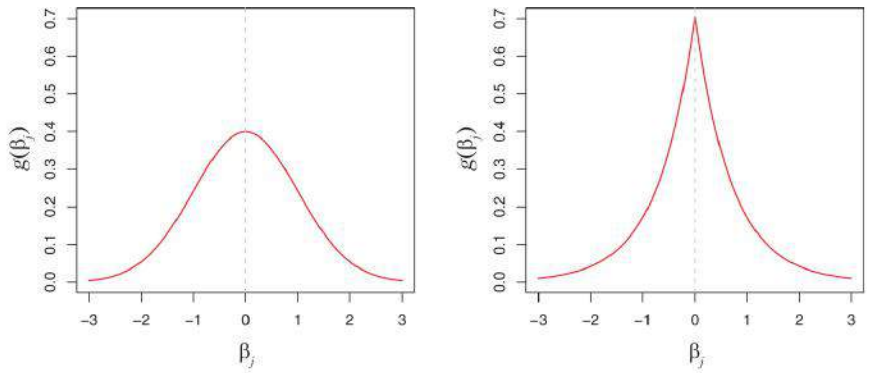

РИСУНОК 6.11. Слева: *решение по методу гребневой регрессии — это мода апостериорного распределения β при гауссовом априорном распределении.* Справа: *решение по методу гребневой регрессии — это мода апостериорного распределения β при двойном экспоненциальном априорном распределении*

6.2.3 Выбор гиперпараметра

Подобно рассмотренным в разделе 6.1 методам отбора переменных, которым необходим способ определения оптимальной модели, гребневой регрессии и методу лассо требуется способ нахождения гиперпараметра λ из (6.5) и (6.7), или, что эквивалентно, ограничительного параметра s из (6.8) и (6.9). Перекрестная проверка дает простое решение этой проблемы. Мы выбираем некоторый интервал значений λ и, как описано в главе 5, вычисляем ошибку перекрестной проверки для каждого из этих значений. Затем мы выбираем такое значение гиперпараметра, для которого ошибка перекрестной проверки минимальна. Наконец, мы заново подгоняем модель с использованием всех имеющихся наблюдений и найденного значения гиперпараметра.

На рис. 6.12 показан процесс нахождения λ гребневой регрессии для данных `Credit` с помощью метода перекрестной проверки по отдельным наблюдениям (LOOCV). Вертикальные прерывистые линии соответствуют оптимальному значению λ. В данном случае полученное значение относительно невелико, т. е. оптимальная модель выполняет лишь небольшое сжатие, по сравнению с решением, получаемым по методу наименьших квадратов. Кроме того, прогиб кривой выражен слабо, что указывает на существование довольно широкого диапазона значений λ, которые дали бы очень похожую ошибку на контрольной выборке. В ситуации, подобной этой, мы могли бы просто воспользоваться решением по методу наименьших квадратов.

На рис. 6.13 показан результат применения десятикратной перекрестной проверки к лассо–моделям, подогнанным по имитированным данным из рис. 6.9. Слева на рис. 6.13 изображены ошибки, полученная в ходе перекрестной проверки, а справа — оценки коэффициентов. Вертикальные прерывистые линии отмечают точку, в которой ошибка перекрестной проверки минимальна. Две цветные линии на графике справа соответствуют

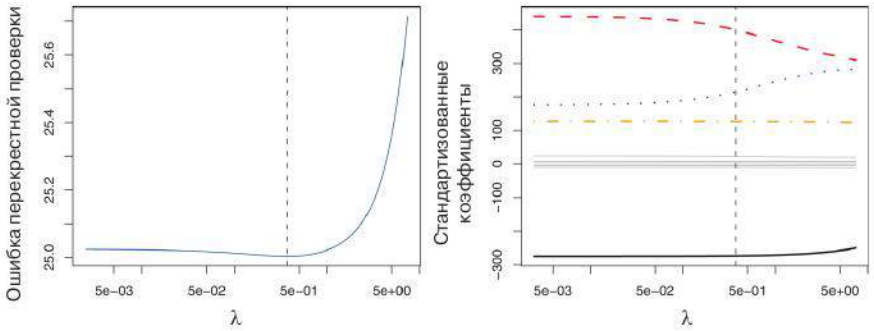

РИСУНОК 6.12. Слева: *ошибки перекрестной проверки, полученные в результате применения гребневой регрессии к данным* `Credit` *при разных значениях* λ. Справа: *оценки коэффициентов в зависимости от* λ. *Вертикальные линии соответствуют значению* λ, *выбранному методом перекрестной проверки*

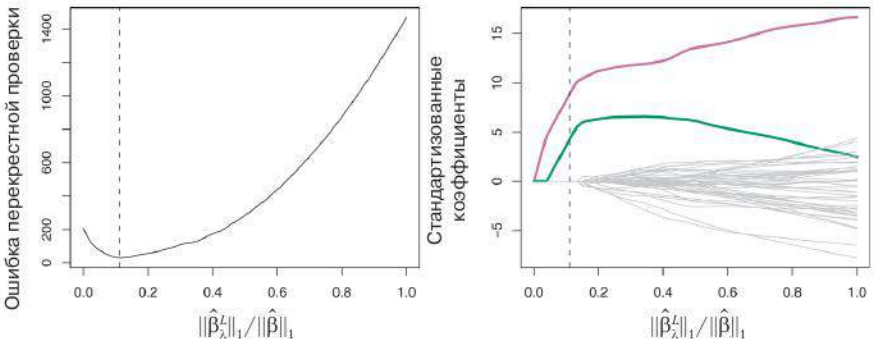

РИСУНОК 6.13. Слева: *среднеквадратичные ошибки, полученные в результате применения десятикратной перекрестной проверки к моделям лассо для имитированных данных из рис. 6.9.* Справа: *показаны соответствующие оценки коэффициентов. Вертикальные прерывистые линии отмечают лассо–модель с наименьшей ошибкой перекрестной проверки*

тем двум предикторам, которые тесно связаны с откликом, а серые линии — предикторам, которые с ним не связаны; такие предикторы часто называют *несущими сигнал переменными* и *шумовыми переменными* соответственно. Метод лассо здесь правильно присвоил более высокие коэффициенты двум несущим сигнал предикторам; кроме того, минимум ошибки перекрестной проверки соответствует набору оценок коэффициентов, в котором ненулевые значения имеются только у несущих сигнал предикторов. Таким образом, несмотря на сложность рассматриваемого случая ($p = 45$ переменных и только $n = 50$ наблюдений), перекрестная проверка в совокупности с методом лассо правильно обнаружила в модели две несущие сигнал переменные. В то же время решение, полученное по

сигнал

методу наименьших квадратов (показано в самом конце на графике справа), присваивает большую оценку коэффициента только одной из двух несущих сигнал переменных.

6.3 Методы снижения размерности

Методы, которые мы обсуждали до сих пор в этой главе, контролировали дисперсию двумя различными способами — либо используя некоторое подмножество предикторов, либо сжимая коэффициенты предикторов в направлении к нулю. Однако построение соответствующих моделей происходит с участием всех исходных предикторов $X_1, X_2, \ldots, X_p$. Теперь же мы рассмотрим класс методов, которые сначала *преобразуют* предикторы, а затем подгоняют модель по методу наименьших квадратов на основе этих преобразованных переменных. Мы будем называть их методами *снижения размерности*.

снижение размерности

Пусть $Z_1, Z_2, \ldots, Z_M$ представляют собой $M < p$ *линейных комбинаций* исходных p предикторов. Другими словами,

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j \tag{6.16}$$

для некоторых констант $\phi_{1m}, \phi_{2m}, \ldots, \phi_{pm}$, $m = 1, \ldots, M$. Тогда мы можем подогнать линейную регрессионную модель

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, \quad i = 1, \ldots, n \tag{6.17}$$

по методу наименьших квадратов. Заметьте, что в (6.17) регрессионные коэффициенты обозначены как $\theta_0, \theta_1, \ldots, \theta_M$. При удачном подборе констант $\phi_{1m}, \phi_{2m}, \ldots, \phi_{pm}$ такие методы снижения размерности часто могут превосходить регрессию по методу наименьших квадратов. Иными словами, подгонка модели (6.17) по методу наименьших квадратов может давать более высокие результаты, чем подгонка по этому же методу модели (6.1).

Происхождение термина «*снижение размерности*» связано с тем фактом, что проблема оценивания $p + 1$ коэффициентов $\beta_0, \beta_1, \ldots, \beta_p$ сводится к более простой проблеме по оцениванию $M + 1$ коэффициентов $\theta_0, \theta_1, \ldots, \theta_M$, где $M < p$. Другими словами, размерность проблемы снижается с $p + 1$ до $M + 1$.

Заметьте, что из (6.16) вытекает следующее соотношение:

$$\sum_{m=1}^{M} \theta_m z_{im} = \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{jm} x_{ij} = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m \phi_{jm} x_{ij} = \sum_{j=1}^{p} \beta_j x_{ij},$$

где

$$\beta_j = \sum_{m=1}^{M} \theta_m \phi_{jm}. \tag{6.18}$$

Таким образом, (6.17) можно рассматривать как частный случай линейной модели (6.1). Снижение размерности накладывает ограничения на оценки коэффициентов β_j, поскольку сейчас они должны принимать форму (6.18). Это ограничение потенциально может приводить к смещенным оценкам коэффициентов. Однако в ситуациях, когда p велико, по сравнению с n, значение $M \ll p$ может привести к существенному снижению дисперсии этих оценок. Когда $M = p$ и все Z_m линейно независимы, (6.18) не накладывает никаких ограничений. В этом случае снижения размерности не происходит, и модель (6.17) эквивалентна регрессии по методу наименьших квадратов, содержащей все исходные p предикторы.

Все методы снижения размерности работают в два шага. Сначала вычисляются преобразованные предикторы $Z_1, Z_2, \ldots, Z_M$. Далее подгоняется модель на основе этих M предикторов. Однако выбор $Z_1, Z_2, \ldots, Z_M$ или, что эквивалентно, выбор констант ϕ_{jm} можно сделать разными способами. В этой главе мы рассмотрим два подхода: *метод главных компонент* и *метод частных наименьших квадратов*.

6.3.1 Регрессия на главные компоненты

анализ главных компонент

Анализ главных компонент (PCA)[13] — это популярный метод, позволяющий получить набор признаков малой размерности из некоторой большой совокупности признаков. Более подробно PCA обсуждается в главе 10 как метод *обучения без учителя*. Здесь мы опишем его применение в качестве инструмента для снижения размерности данных в контексте регрессии.

Общее описание анализа главных компонент

PCA представляет собой метод снижения размерности некоторой матрицы $\mathbf{X}$, имеющей p столбцов и n строк. Направление *первой главной компоненты* данных соответствует направлению, вдоль которого имеет место *наибольшая дисперсия* наблюдений. Рассмотрим, например, рис. 6.14, где показана численность населения (pop, в тысячах человек) и затраты некоторой компании на рекламу (ad, в тысячах долларов) в 100 городах. Зеленая сплошная линия показывает направление первой главной компоненты данных. Хорошо видно, что это — направление, вдоль которого имеет место наибольшая дисперсия точек. Другими словами, если бы мы выполнили *проецирование* 100 точек на эту линию (как показано на рис. 6.15 слева), то получившиеся точки имели бы максимально возможную дисперсию; проецирование наблюдений на любую другую линию дало бы наблюдения с меньшей дисперсией. Проецирование точки на линию — это просто нахождение ближайшего к этой точке положения на линии.

Графическое изображение первой главной компоненты приведено на рис. 6.14, но как ее можно описать математически? Она задается следующей формулой:

$$Z_1 = 0.839 \times (\texttt{pop} - \overline{\texttt{pop}}) + 0.544 \times (\texttt{ad} - \overline{\texttt{ad}}). \tag{6.19}$$

Здесь $\phi_{11} = 0.893$ и $\phi_{21} = 0.544$ — это нагрузки компонент[14], которые задают упомянутое выше направление первой главной компоненты. В (6.19)

[13] Аббревиатура от «principal components analysis» — *Прим. пер.*

[14] В оригинале используется термин «principal component loadings». — *Прим. пер.*

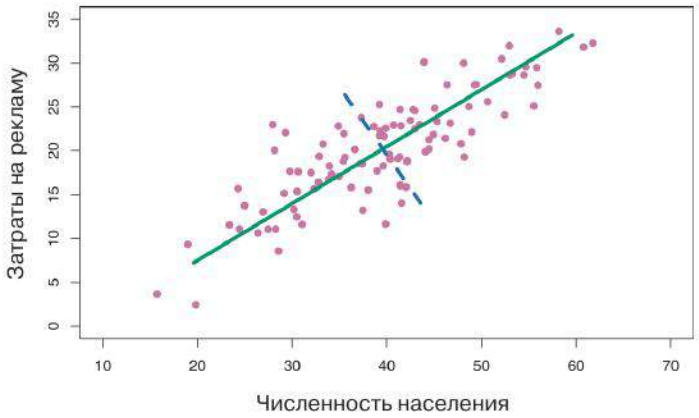

РИСУНОК 6.14. *Численность населения (*pop*) и затраты на рекламу (*ad*) в 100 городах показаны в виде фиолетовых точек. Зеленая сплошная линия соответствует первой главной компоненте, а голубая прерывистая линия — второй главной компоненте*

$\overline{\texttt{pop}}$ обозначает среднее из всех значений pop, а $\overline{\texttt{ad}}$ — среднее из всех затрат на рекламу. Идея состоит в том, что из всех возможных *линейных комбинаций* pop и ad, для которых $\phi_{11}^2 + \phi_{21}^2 = 1$, эта конкретная линейная комбинация дает наибольшую дисперсию, или, другими словами, максимизирует $\text{Var}(\phi_{11} \times (\texttt{pop} - \overline{\texttt{pop}}) + \phi_{21} \times (\texttt{ad} - \overline{\texttt{ad}}))$. Рассмотрению подлежат только линейные комбинации вида $\phi_{11}^2 + \phi_{21}^2 = 1$, поскольку иначе мы могли бы произвольным образом увеличить ϕ_{11} и ϕ_{21} для «раздутия» дисперсии. Оба коэффициента нагрузок компонент в (6.19) положительные и близки по своим значениям, в связи с чем Z_1 почти не отличается от *среднего* значения двух переменных.

Поскольку $n = 100$, то в уравнении (6.19) pop и ad, а следовательно и Z_1, представляют собой векторы длиной 100. Например,

$$z_{i1} = 0.839 \times (\texttt{pop}_i - \overline{\texttt{pop}}) + 0.544 \times (\texttt{ad}_i - \overline{\texttt{ad}}). \tag{6.20}$$

Величины $z_{11}, \ldots, z_{n1}$ известны как «*значения главных компонент*»[15] и показаны справа на рис. 6.15.

Имеется и другая интерпретация PCA: вектор первой главной компоненты определяет линию, которая находится *максимально близко* ко всем наблюдениям одновременно. Например, на рис. 6.14 линия первой главной компоненты минимизирует сумму квадратов перпендикулярных расстояний от каждой точки до этой линии. Эти расстояния показаны в виде прерывистых линий слева на рис. 6.15, где крестики отмечают места проекции каждой точки на линию первой главной компоненты. Первая главная компонента была выбрана таким образом, чтобы спроецированные наблюдения находились *максимально близко* к исходным наблюдениям.

Справа на рис. 6.15 исходный график был повернут так, чтобы направление первой главной компоненты совпало с осью X. Можно показать, что

[15] В оригинале используется термин «principal component scores». — *Прим. пер.*

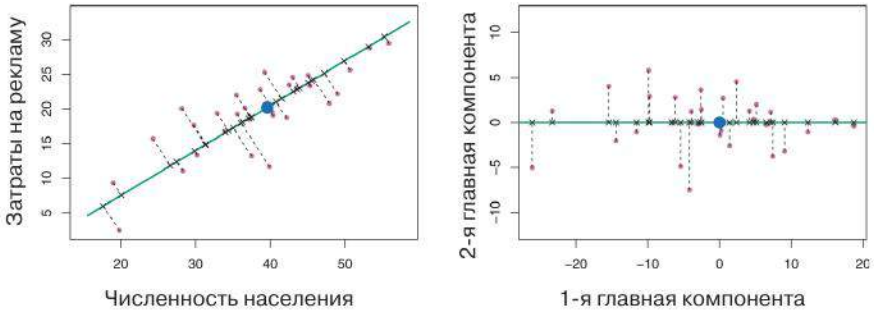

РИСУНОК 6.15. *Часть набора данных по расходам на рекламу. Средние значения численности населения и затрат на рекламу показаны в виде голубой точки.* Слева: *направление первой главной компоненты показано зеленой линией. Это направление, вдоль которого имеет место максимальная дисперсия данных; оно также определяет линию, которая максимально близко находится ко всем n наблюдениям. Расстояния от каждого наблюдения до главной компоненты показаны в виде прерывистых отрезков черного цвета. Голубая точка имеет координаты* $(\overline{\texttt{pop}}, \overline{\texttt{ad}})$. Справа: *график, приведенный слева, был повернут так, чтобы направление первой главной компоненты совпало с осью X*

значение первой главной компоненты для i–го наблюдения, вычисленное по (6.20), представляет собой расстояние на оси X от i–го крестика до нуля. Например, точка, расположенная в нижнем левом углу на исходном графике имеет большое отрицательное значение главной компоненты ($z_{i1} = -26.1$), тогда как точка в правом верхнем углу имеет большое положительное значение главной компоненты ($z_{i1} = 18.7$). Эти значения можно непосредственно рассчитать при помощи (6.20).

Мы можем думать о том или ином значении главной компоненты Z_1 как о числе, которое содержит информацию одновременно по численности населения и затратам на рекламу в соответствующем городе. В данном примере $z_{i1} = 0.893 \times (\texttt{pop} - \overline{\texttt{pop}}) + 0.544 \times (\texttt{ad} - \overline{\texttt{ad}}) < 0$ указывает на то, что численность населения и затраты на рекламу в соответствующем городе ниже среднестатистических значений. Положительное значение главной компоненты указывает на обратную ситуацию. Насколько хорошо одно число может одновременно описывать pop и ad? В данном случае рис. 6.14 говорит о том, что pop и ad связаны друг с другом примерно линейно, и поэтому можно ожидать, что такой количественный показатель будет работать хорошо. На рис. 6.16 показаны зависимости z_{i1} от pop и ad. Эти графики указывают на сильную связь между первой главной компонентой и этими двумя переменными. Другими словами, первая главная компонента, по–видимому, содержит большую долю информации, заключенной в pop и ad.

До сих пор мы уделяли основное внимание первой главной компоненте. В целом можно построить вплоть до p отдельных главных компонент. Вторая главная компонента Z_2 представляет собой линейную комбинацию переменных, которая не коррелирует с Z_1 и обладает наибольшей

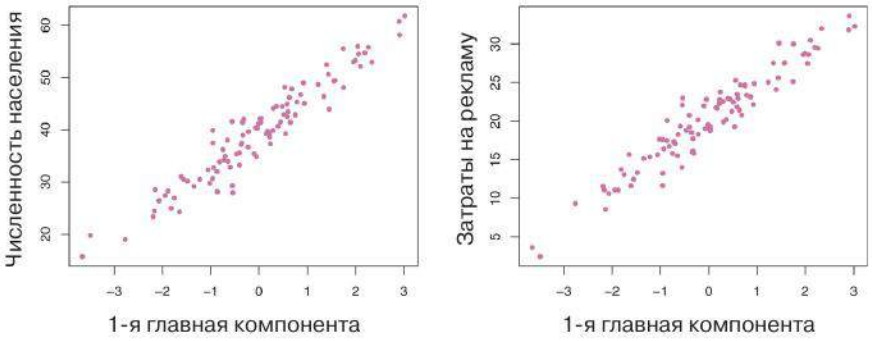

РИСУНОК 6.16. *Графики зависимости значений первой главной компоненты z_{i1} от* pop *и* ad. *Это четко выраженные зависимости*

дисперсией с учетом этого ограничения. Направление второй главной компоненты показано на рис. 6.14 в виде прерывистой голубой линии. Оказывается, что условие нулевой корреляции между Z_1 и Z_2 эквивалентно условию, согласно которому это направление должно быть *перпендикулярным*, или *ортогональным*, направлению первой главной компоненты. Вторая главная компонента задается следующей формулой:

$$Z_2 = 0.544 \times (\texttt{pop} - \overline{\texttt{pop}}) - 0.839 \times (\texttt{ad} - \overline{\texttt{ad}}).$$

Поскольку данные по рекламе содержат только две переменные, то первые две главные компоненты включают всю информацию, заключенную в pop и ad. Однако первая главная компонента, по определению, содержит большую часть информации. Например, обратите внимание на то, насколько вариабельность z_{i1} (ось X) больше вариабельности z_{i2} (ось Y) (рис. 6.15 справа). Тот факт, что значения второй главной компоненты намного ближе к нулю, указывает на меньшую долю информации, заключенной в этой компоненте. В качестве другого примера на рис. 6.17 приведены зависимости z_{i2} от pop и ad. Связь между этими двумя переменными и второй главной компонентой очень слабая — это еще раз подтверждает, что в данном примере для адекватного представления численности населения и затрат на рекламу достаточно первой главной компоненты.

Метод регрессии на главные компоненты

регрессия на главные компоненты

Метод регрессии на главные компоненты (PCR)[16] включает вычисление первых M главных компонент $Z_1, \ldots, Z_M$ и последующее использование этих компонент в качестве предикторов в линейной регрессионной модели, которая подгоняется по методу наименьших квадратов. Ключевая идея заключается в том, что для объяснения основной доли дисперсии в данных, а также их связи с откликом часто достаточным оказывается применение лишь небольшого числа главных компонент. Другими словами, мы предполагаем, что *направления, вдоль которых $X_1, \ldots, X_p$ демонстрируют наибольшую дисперсию, являются направлениями, которые связаны с Y.*

[16] Аббревиатура от «principal components regression». — *Прим. пер.*

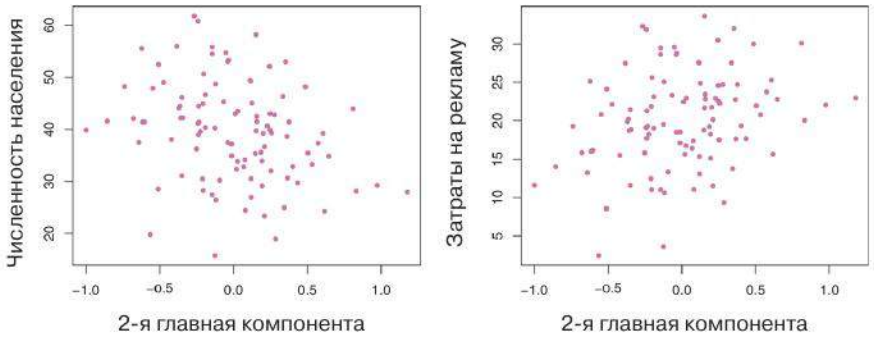

РИСУНОК 6.17. *Графики зависимости значений второй главной компоненты z_{i2} от* `pop` *и* `ad`. *Это очень слабые зависимости*

Несмотря на то что это предположение не обязательно является верным, часто оно оказывается достаточно обоснованной аппроксимацией, дающей хорошие результаты на практике.

Если лежащее в основе PCR предположение верно, то подгонка линейной модели по методу наименьших квадратов с использованием $Z_1, \ldots, Z_M$ даст более приемлемые результаты, чем подгонка аналогичной модели с использованием $X_1, \ldots, X_p$, поскольку основная или даже вся заключенная в данных информация о связи с откликом содержится в $Z_1, \ldots, Z_M$, и, оценивая $M \ll p$ коэффициентов, мы можем избежать проблему переобучения. В случае с данными по рекламе первая главная компонента объясняет основную долю дисперсии, заключенной в `pop` и `ad`, и поэтому регрессия на главные компоненты, предсказывающая интересующий нас отклик (`sales`) на основе этой единственной переменной, с большой вероятностью даст довольно качественную модель.

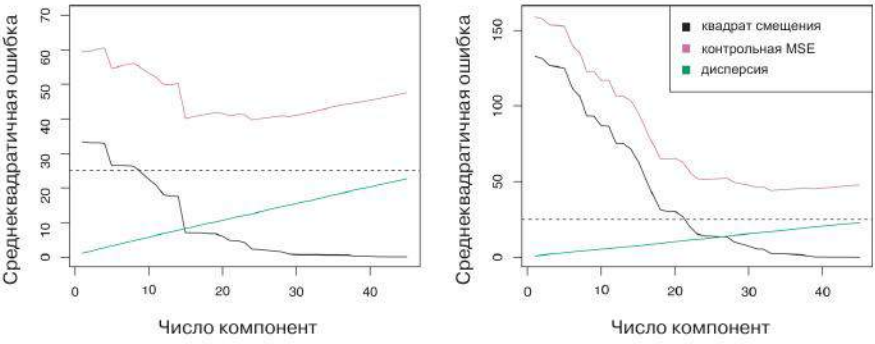

РИСУНОК 6.18. *Метод PCR был применен к двум наборам имитированных данных.* Слева: *имитированные данные из рис. 6.8.* Справа: *имитированные данные из рис. 6.9*

На рис. 6.18 показаны PCR–модели, подогнанные по имитированным данным из рис. 6.8 и 6.9. Напомним, что эти наборы данных содержат

$n = 50$ наблюдений и $p = 45$ переменных. Однако в первом наборе данных отклик был функцией от всех предикторов, а во втором — только от двух предикторов. Кривые изображают зависимость MSE от M — числа главных компонент, используемых в качестве предикторов в линейной модели. По мере добавления главных компонент в регрессионную модель смещение снижается, но дисперсия возрастает. Это приводит к возникновению типичной U–образной кривой MSE. При $M = p = 45$ PCR просто сводится к модели, подогнанной по методу наименьших квадратов по всем предикторам. Из этого рисунка следует, что выполнение PCR с хорошо подобранным значением M может дать существенно более качественную модель, нежели обычная регрессия по методу наименьших квадратов (особенно хорошо это видно на графике слева). Тем не менее приведенные на рис. 6.5, 6.8 и 6.9 результаты применения гребневой регрессии и лассо указывают на то, что PCR в данном случае работает хуже методов регуляризации.

Относительно более низкое качество PCR на рис. 6.18 обусловлено свойствами этих данных: они были сгенерированы таким образом, чтобы для адекватного моделирования отклика потребовалось большое число главных компонент. В то же время PCR обычно хорошо работает в случаях, когда первых нескольких компонент достаточно для описания основной доли дисперсии предикторов и их связи с откликом. Слева на рис. 6.19 приведены результаты для двух других наборов имитированных данных, которые по своим свойствам лучше подходят для PCR. Здесь отклик был сгенерирован таким образом, чтобы он зависел только от первых пяти компонент. Теперь смещение быстро снижается по мере увеличения M — числа главных компонент в PCR–модели. Среднеквадратичная ошибка демонстрирует четко выраженный минимум при $M = 5$. Справа на рис. 6.19 показаны результаты применения к этим данным методов гребневой регрессии и лассо. Все три метода обеспечивают существенное улучшение, по сравнению с методом наименьших квадратов. Тем не менее PCR и гребневая регрессия несколько превосходят метод лассо.

Следует отметить, что, несмотря на предоставляемый методом PCR простой способ построения регрессии с $M < p$ предикторами, он *не является* методом отбора переменных. Это связано с тем, что каждая из M главных компонент в регрессионной модели является линейной комбинацией всех p исходных переменных. Например, в (6.19) Z_1 была линейной комбинацией как `pop`, так и `ad`. Поэтому, несмотря на то что PCR часто дает довольно хорошие результаты на практике, этот метод не приводит к созданию моделей с некоторым ограниченным набором исходных переменных. В этом смысле метод PCR больше похож на гребневую регрессию, чем на лассо. Более того, можно показать, что PCR и гребневая регрессия являются близкородственными методами. О гребневой регрессии даже можно думать как о непрерывной версии PCR[17]!

Число главных компонент M для PCR обычно выбирают при помощи перекрестной проверки. Результаты применения PCR к данным `Credit` приведены на рис. 6.20; справа показаны ошибки перекрестной проверки, полученные для разных значений M. Для этих данных минимальная ошибка наблюдается при использовании $M = 10$ главных компонент; это

[17] Дополнительную информацию об этом можно найти в разделе 3.5 книги Hastie T., Tibshirani R., Friedman J. (2009) The Elements of Statistical Learning. Springer, 745 p.

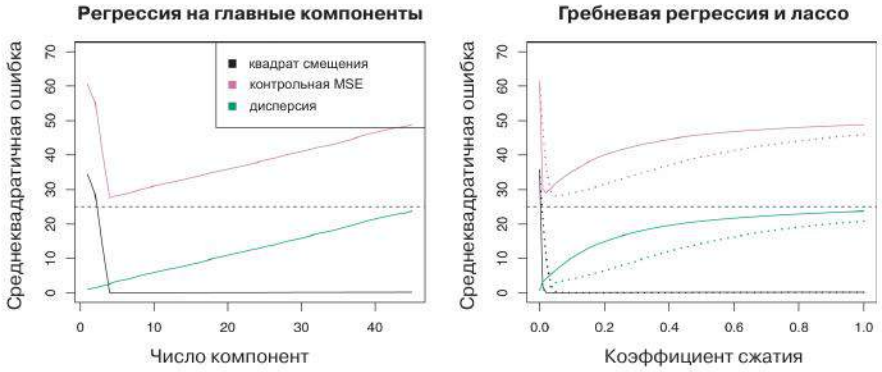

РИСУНОК 6.19. *Методы PCR, гребневой регрессии и лассо были применены к набору имитированных данных, в котором первые пять компонент содержат всю информацию об отклике Y. На каждом графике неустранимая ошибка* $\mathrm{Var}(\epsilon)$ *показана в виде горизонтальной прерывистой линии.* Слева: *результаты для PCR.* Справа: *результаты для лассо (сплошная линия) и гребневой регрессии (прерывистая линия). По оси X отложены значения фактора сжатия оценок регрессионных коэффициентов, который равен отношению нормы* ℓ_2 *сжатых оценок к норме* ℓ_2 *оценок, полученных по методу наименьших квадратов*

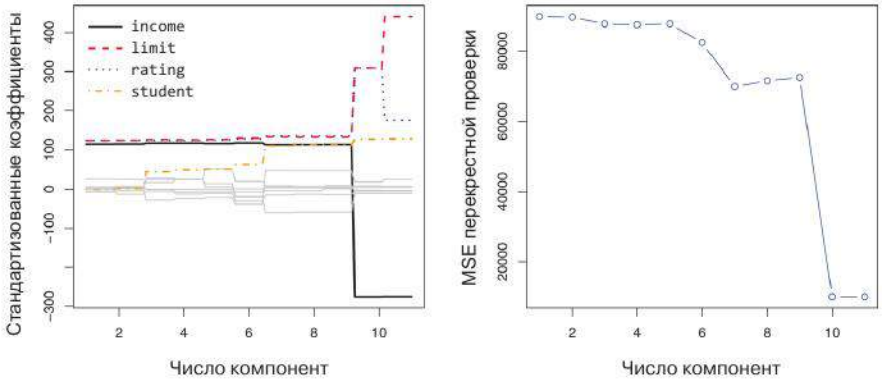

РИСУНОК 6.20. Слева: *оценки стандартизованных регрессионных коэффициентов, полученные для данных* `Credit` *по методу PCR при разных значениях M.* Справа: *среднеквадратичная ошибка десятикратной перекрестной проверки для PCR–моделей с разными значениями M*

соответствует почти полному отсутствию снижения размерности, поскольку PCR–модель с $M = 11$ эквивалентна обычной регрессии по методу наименьших квадратов.

Перед тем как вычислять главные компоненты для PCR, рекомендуется *стандартизовать* каждый предиктор по формуле (6.6). Подобная стандартизация позволит привести все переменные к одинаковой шкале

измерения. Без стандартизации предикторы, обладающие высокой дисперсией, обычно будут иметь больший вес в полученных главных компонентах, а шкалы, по которым измерены такие переменные, в итоге будут влиять на получаемую PCR–модель. Однако если все переменные изначально выражаются в одинаковых единицах (например, в килограммах или дюймах), то выполнять их стандартизацию не обязательно.

6.3.2 Метод частных наименьших квадратов

Описанный только что метод PCR основан на нахождении линейных комбинаций, или *направлений*, которые лучше всего описывают предикторы $X_1, \ldots, X_p$. Эти направления находят путем *обучения без учителя*, поскольку отклик Y не используется в расчете главных компонент. Другими словами, отклик *не помогает* нахождению главных компонент. В связи с этим PCR обладает следующим недостатком: нет гарантии, что хорошо объясненные предикторами компоненты окажутся также оптимальными для предсказания отклика. Более подробно методы обучения без учителя обсуждаются в главе 10.

метод частных наименьших квадратов

Теперь мы представим метод *частных наименьших квадратов* (PLS)[18] — альтернатива PCR, которая позволяет выполнять *обучение с учителем*. Подобно PCR, PLS является методом снижения размерности, который сначала вычисляет новый набор переменных $Z_1, \ldots, Z_M$, являющихся линейными комбинациями исходных переменных, а затем подгоняет линейную модель по методу наименьших квадратов на основе этих M новых переменных. Однако, в отличие от PCR, PLS определяет новые переменные по принципу обучения с учителем, т. е. он использует отклик Y для нахождения таких новых переменных, которые не только аппроксимируют исходные переменные, но и *связаны с откликом*. Проще говоря, метод PLS пытается найти направления, которые помогают объяснить как отклик, так и предикторы.

Рассмотрим процедуру вычисления первого PLS–направления. После стандартизации p предикторов PLS вычисляет первое направление Z_1, приравнивая каждую константу ϕ_{j1} в (6.16) к коэффициенту простой линейной регрессии Y по X_j. Можно показать, что этот коэффициент пропорционален корреляции между Y и X_j. Следовательно, при вычислении $Z_1 = \sum_{j=1}^{p} \phi_{j1} X_j$ PLS уделяет особое внимание переменным, которые наиболее тесно связаны с откликом.

На рис. 6.21 приведен пример применения PLS к данным по рекламе. Сплошная зеленая линия показывает первое PLS–направление, а прерывистая линия — направление первой главной компоненты PCR. Метод PLS выбрал направление, вдоль которого изменение `pop` на одну единицу вызывает менее выраженное изменение `ad`, чем в случае с PCR. Из этого следует, что переменная `pop` связана с откликом более тесно, чем `ad`. PLS–направление описывает предикторы не так хорошо, как PCA, но при этом оно лучше объясняет отклик.

Для нахождения второго направления PLS мы сначала корректируем все переменные по Z_1 — для этого мы строим регрессионные зависимости каждой переменной от Z_1 и находим *остатки*. Эти остатки можно интерпретировать как информацию, которая осталась необъясненной первым

[18] Аббревиатура от «partial least squares». — *Прим. пер.*

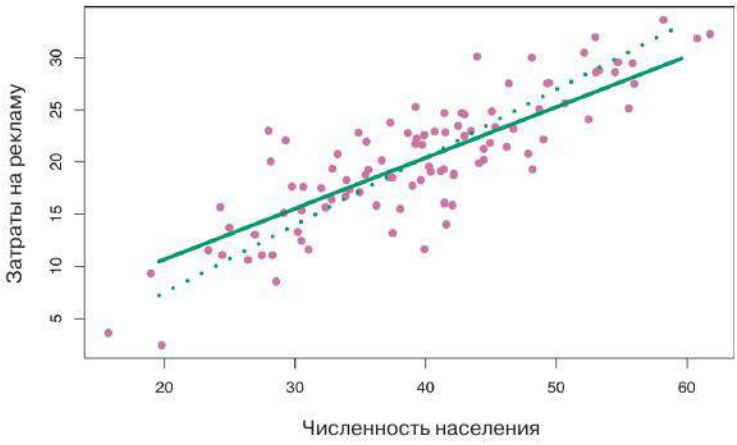

РИСУНОК 6.21. *Показаны первое PLS–направление (сплошная линия) и первое PCR–направление (прерывистая линия) для данных по рекламе*

PLS–направлением. Далее для вычисления Z_2 мы используем эти *ортогонализованные* данные точно так же, как мы использовали исходные данные для вычисления Z_1. Такой итеративный подход можно применить M раз для нахождения нескольких PLS–компонент $Z_1, \ldots, Z_M$. В конце этой процедуры мы применяем метод наименьших квадратов для подгонки линейной модели, которая предсказывает Y по $Z_1, \ldots, Z_M$ аналогично тому, как это происходит в PCR.

Как и в случае с PCR, число PLS–направлений M является гиперпараметром, который обычно выбирают при помощи перекрестной проверки. Как правило, перед применением PLS мы выполняем стандартизацию предикторов и отклика.

PLS является популярным методом в хемометрике[19], где в результате оцифровки спектрометрических сигналов возникает большое количество переменных. На практике этот метод работает не лучше гребневой регрессии или PCR. Хотя снижение размерности по принципу обучения с учителем может привести к снижению смещения, потенциально оно может также увеличить дисперсию, и поэтому в целом преимущество PLS, по сравнению с PCR, является весьма незначительным.

6.4 Особенности работы с данными большой размерности

6.4.1 Данные большой размерности

Большинство традиционных статистических методов регрессии и классификации предназначено для случаев *малой размерности*, когда число на-

малая размерность

[19] Хемометрика — это научная дисциплина, находящаяся на стыке химии и математики, предметом которой являются математические методы изучения химических явлений. — *Прим. пер.*

блюдений n намного больше числа переменных p. Отчасти это обусловлено тем, что на протяжении большей части истории статистики многие научные проблемы, требующие применения соответствующих методов, обладали малой размерностью. Например, представьте себе разработку модели для предсказания кровяного давления у пациента по его возрасту, полу и индексу массы тела (ИМТ). Есть три предиктора (или четыре, если в модель включен также свободный член) и, возможно, несколько тысяч пациентов, для которых известны возраст, пол и ИМТ. Следовательно, $n \gg p$, и поэтому проблема имеет малую размерность. (Под размерностью мы имеем в виду значение p.)

В последние 20 лет новые технологии изменили способы получения данных в самых разнообразных областях, таких как финансы, маркетинг и медицина. Сегодня является обычным учет практически не ограниченного числа переменных (большое значение p). В то время как p может быть очень большим, число наблюдений n часто ограничено в связи с затратами, доступностью образцов или другими причинами. Вот два примера:

1. Вместо предсказания давления крови по возрасту, полу и ИМТ можно было бы также получить данные для полумиллиона однонуклеотидных полиморфизмов (ОНП)[20] и включить их в предсказательную модель. Тогда $n \approx 200$ и $p \approx 500\,000$.

2. Маркетинговый аналитик, который желает понять закономерности поведения покупателей в онлайн–магазине, мог бы рассматривать в качестве предикторов все фразы, введенные посетителями в поисковую систему. Иногда такую модель называют «мешком слов»[21]. У того же исследователя мог бы быть доступ к истории команд только нескольких сотен или нескольких тысяч пользователей поисковой системы, согласившихся поделиться своей информацией с исследователем. Для того или иного пользователя каждый поисковый термин p выражается в виде 0 (отсутствует) или 1 (присутствует), что приводит к возникновению большого бинарного вектора признаков. Тогда $n \approx 1000$, а p намного превышает это значение.

большая размерность

Данные, в которых число переменных превышает число наблюдений, часто называют данными большой размерности. Классические методы вроде метода наименьших квадратов не подходят для такого сценария. Многие из проблем, возникающих в ходе анализа данных большой размерности, уже обсуждались в этой книге: речь идет о роли компромисса между смещением и дисперсией, а также об угрозе переобучения. Хотя этим проблемам всегда необходимо уделять внимание, они могут приобретать особую актуальность, когда число переменных значительно превышает число наблюдений.

Мы дали определение *сценарию большой размерности* как случаю, когда число переменных p превышает число наблюдений n. Однако рассмотренные ниже особенности работы с такими данными, безусловно, применимы также для случаев, когда p несколько меньше n, и о них всегда нужно помнить при реализации обучения с учителем.

[20] Распространенные в некоторой популяции единичные мутации ДНК.
[21] В оригинале используется термин «bag–of–words». — *Прим. пер.*

6.4.2 Что не так с большими размерностями?

Чтобы подчеркнуть необходимость проявления чрезвычайной осторожности при $p > n$ и продемонстрировать специальные методы регрессии и классификации для таких случаев, мы начнем с рассмотрения проблем, возникающих в ходе применения статистического метода, не предназначенного для задач большой размерности. Для этого мы обратимся к методу наименьших квадратов. Однако те же идеи применимы и к логистической регрессии, линейному дискриминантному анализу и другим классическим методам статистики.

Когда число переменных p равно или превышает число наблюдений n, описанный в главе 3 метод наименьших квадратов неприменим (во всяком случае, его *не следует* применять). Причина проста: вне зависимости от наличия истинной связи между переменными и откликом метод наименьших квадратов даст оценки коэффициентов, обеспечивающие идеальное описание данных, т. е. остатки будут равны нулю.

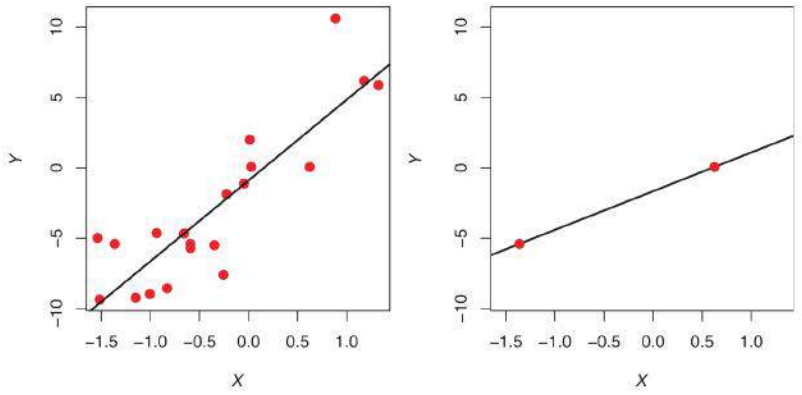

РИСУНОК 6.22. Слева: *регрессия по методу наименьших квадратов для случая малой размерности.* Справа: *линейная регрессия для случая с двумя наблюдениями, когда оценке подлежат два параметра (свободный член и коэффициент угла наклона)*

На рис. 6.22 приведен пример с $p = 1$ (плюс свободный член) для двух случаев — с 20 наблюдениями и только с двумя наблюдениями. При наличии 20 наблюдений $n > p$, и регрессионная линия, полученная по методу наименьших квадратов, пытается аппроксимировать 20 наблюдений максимально хорошо, но все же не описывает данные идеально. С другой стороны, при наличии только двух наблюдений, вне зависимости от их конкретных значений, регрессионная линия будет в точности соответствовать данным. Это проблематично, поскольку идеальное соответствие данным почти наверняка будет сопровождаться переобучением. Иными словами, несмотря на возможность идеального описания обучающих данных в задачах большой размерности, итоговая модель будет работать очень плохо на независимой контрольной выборке, и поэтому такая модель окажется бесполезной. Более того, на рис. 6.22 мы видим, что именно это и произошло: показанная справа модель очень плохо сработает на контрольной

выборке, образованной наблюдениями из графика слева. Проблема очевидна: при $p > n$ или $p \approx n$ модель простой линейной регрессии является слишком *гибкой* и, следовательно, переобученной.

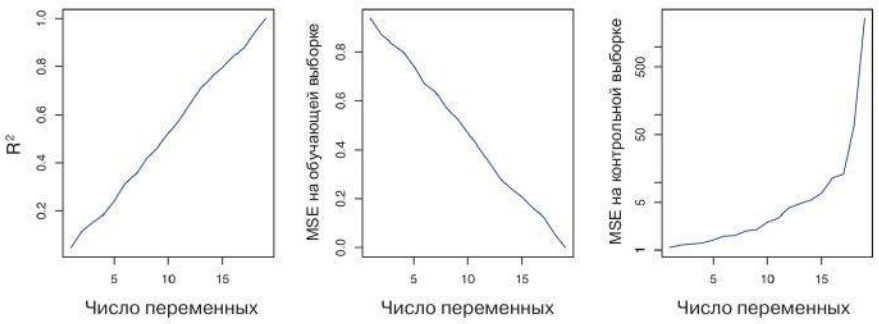

РИСУНОК 6.23. *В модель для имитированного набора данных с n = 20 наблюдениями добавлены переменные, которые никак не связаны с откликом.* Слева: *по мере добавления переменных R^2 увеличивается до 1.* В центре: *по мере добавления переменных MSE на обучающей выборке снижается до 0.* Справа: *по мере добавления переменных MSE на контрольной выборке возрастает*

На рис. 6.23 приведен дополнительный пример риска, возникающего в результате неосторожного применения метода наименьших квадратов при большом числе переменных p. К искусственно созданному набору данных с 20 наблюдениями подогнаны регрессионные модели, включающие от 1 до 20 переменных, которые никак не связаны с откликом. Видно, что по мере добавления переменных в модель коэффициент детерминации R^2 увеличивается, тогда как MSE на обучающей выборке снижается до нуля, *даже несмотря на то, что эти переменные никак не связаны с откликом.* В то же время MSE на независимой контрольной выборке достигает очень высоких значений, поскольку добавление новых предикторов в модель приводит к значительному увеличению дисперсии оценок коэффициентов. MSE на контрольной выборке четко указывает на то, что оптимальная модель содержит не больше нескольких переменных. Однако неосторожно принимая во внимание только значения R^2 или MSE, полученные на обучающей выборке, мы могли бы ошибочно заключить, что модель с максимальным числом предикторов является оптимальной. Этот пример показывает, как важно быть внимательным при анализе данных с большим числом переменных и всегда оценивать качество модели на независимой контрольной выборке.

В подразделе 6.1.3 мы рассмотрели несколько подходов для коррекции RSS и R^2, полученных на обучающих данных, с целью учета числа входящих в модель переменных. К сожалению, подходы, основанные на C_p, AIC и BIC, неприменимы для задач большой размерности в связи с трудностями по оцениванию $\hat{\sigma}^2$. (Например, формула для $\hat{\sigma}^2$ из главы 3 в таких случаях дает оценку $\hat{\sigma}^2 = 0$.) Аналогичные проблемы в задачах большой размерности возникают при использовании скорректированного коэффициента детерминации R^2, поскольку очень легко можно получить

модель, у которой значение этого коэффициента равно 1. Очевидно, что для решения задач большой размерности необходимы альтернативные, более подходящие для этого методы.

6.4.3 Регрессия для данных большой размерности

Оказывается, что многие из рассмотренных в этой главе методов создания *менее гибких* моделей (последовательное включение переменных, гребневая регрессия, лассо и регрессия на главные компоненты) подходят для выполнения регрессионного анализа данных большой размерности. В сущности, эти методы помогают избежать переобучения за счет менее гибкой процедуры подгонки моделей, чем метод наименьших квадратов.

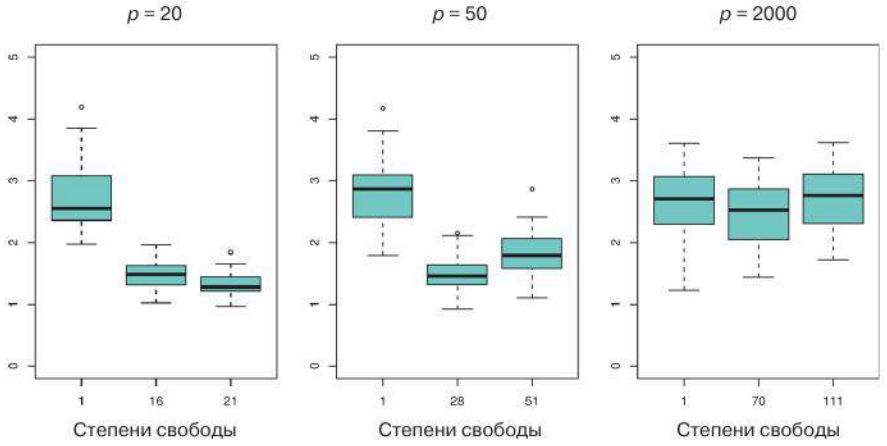

РИСУНОК 6.24. *Метод лассо был применен к набору данных с n = 100 наблюдениями и тремя разными значениями p (число переменных). Из всех p переменных только 20 были действительно связаны с откликом. Диаграммы размахов показывают среднеквадратичные ошибки на контрольных данных, полученные для трех разных значений гиперпараметра λ из (6.7). Для облегчения интерпретации результатов вместо λ приведены значения числа степеней свободы; в случае с лассо оказывается, что это соответствует просто числу ненулевых оценок коэффициентов. При p = 20 наименьшая MSE на контрольной выборке была получена при наименьшей степени регуляризации. При p = 50 наименьшая MSE на контрольной выборке была достигнута при значительной степени регуляризации. При p = 2000 метод лассо сработал плохо при всех уровнях регуляризации, поскольку только 20 из 2000 переменных действительно связаны с откликом*

Рисунок 6.24 иллюстрирует качество лассо–модели на примере простого имитированного набора данных. Имеется $p = 20$, 50 и 2000 переменных, из которых в действительности с откликом связаны только 20 переменных. Метод лассо был применен к 100 обучающим наблюдениями, а MSE была оценена на независимой контрольной выборке. При $p = 20$ минимальная ошибка на контрольной выборке была достигнута при низком значении λ

из (6.7), однако при более высоких p минимальная ошибка на контрольной выборке была достигнута при более высоком значении λ. На каждой диаграмме размахов вместо использованных значений λ приведены значения числа степеней свободы итоговой модели, полученной по методу лассо, — это просто количество ненулевых оценок коэффициентов, которое является мерой гибкости модели. Рисунок 6.24 подчеркивает три важных обстоятельства: (1) регуляризация, или сжатие, играет важную роль в решении задач большой размерности, (2) выбор подходящего значения гиперпараметра имеет критическое значение для обеспечения хорошего качества предсказаний и (3) ошибка на контрольной выборке имеет тенденцию возрастать при увеличении размерности задачи (т. е. увеличении числа переменных, или предикторов), если дополнительные переменные в действительности не связаны с откликом.

проклятие размерности

Оказывается, что третий из перечисленных пунктов является ключевым принципом в анализе данных большой размерности и носит название «*проклятие размерности*». Можно было бы думать, что по мере увеличения числа включенных в модель переменных качество этой модели также будет возрастать. Однако, сравнивая графики, приведенные на рис. 6.24 слева и справа, мы видим, что это необязательно так: в этом примере при увеличении p с 20 до 2000 MSE на контрольной выборке возрастает почти в два раза. В целом *включение в модель дополнительных несущих сигнал переменных улучшит качество этой модели*, что выразится в снижении ошибки на контрольной выборке. Однако добавление шумовых переменных, не связанных с откликом, приведет к снижению качества построенной модели и, следовательно, увеличению ошибки на контрольной выборке. Это обусловлено тем, что шумовые переменные увеличивают размерность задачи, повышая риск переобучения (поскольку коэффициенты таких переменных могут принять ненулевые значения в силу случайного возникновения связей с откликом в обучающей выборке), но не приводя к снижению ошибки на контрольных данных. Таким образом, новые технологии, позволяющие учитывать тысячи и миллионы переменных, — это палка о двух концах: они могут привести к более качественным предсказательным моделям, если эти переменные действительно имеют отношение к решаемой проблеме, но могут привести и к ухудшению результатов, если эти переменные к делу не относятся. Даже если переменные важны, то дисперсия, возникающая в результате оценивания их коэффициентов, может перевесить соответствующее снижение смещения.

6.4.4 Интерпретация результатов в задачах большой размерности

Когда мы применяем лассо, гребневую регрессию и другие регрессионные процедуры для решения задач большой размерности, мы должны быть очень осторожны при изложении получаемых результатов. В главе 3 мы узнали о *мультиколлинеарности* — явлении, при котором переменные в регрессионной модели могут коррелировать друг с другом. В задачах большой размерности проблема мультиколлинеарности принимает экстремальную форму: любую переменную в модели можно представить как линейную комбинацию всех других присутствующих в модели переменных. В сущности, это означает, что мы никогда не сможем узнать наверня-

ка, какие переменные (если таковые вообще имеются) в действительности связаны с откликом, и мы никогда не сможем определить *оптимальные* значения коэффициентов модели. Лучшее, на что мы можем надеяться, — это возможность присвоить большие коэффициенты переменным, которые коррелируют с переменными, действительно связанными с откликом.

Представьте, например, что мы пытаемся предсказать давление крови на основе полумиллиона ОНП и что процедура последовательного включения переменных приводит к созданию качественной предсказательной модели, содержащей 17 из этих ОНП. Было бы неправильным заключить, что эти 17 ОНП предсказывают кровяное давление более эффективно, чем другие ОНП, не включенные в модель. Есть вероятность существования большого числа подмножеств из 17 ОНП, которые предсказывали бы кровяное давление так же хорошо, как ОНП, вошедшие в выбранную модель. Если бы мы получили независимый набор данных и применили к нему метод последовательного включения переменных, то, вероятно, получили бы модель, содержащую другой, совершенно отличный от предыдущего набор ОНП. Это не уменьшает ценности полученной модели — она могла бы оказаться очень эффективной для предсказания кровяного давления у независимой группы пациентов и быть полезной с клинической точки зрения для врачей. Однако мы должны быть осторожны в том, чтобы не переоценить полученные результаты, и должны четко сообщить о нахождении только *одной из нескольких возможных моделей* для предсказания кровяного давления, подлежащей дополнительной проверке на независимых данных.

При решении задач большой размерности важно также соблюдать особую осторожность при описании параметров качества модели и совершаемых ею ошибок. Мы видели, что при $p > n$ очень просто получить бесполезную модель с нулевыми остатками. Следовательно, для описания качества модели, подогнанной по данным большой размерности, *ни при каких обстоятельствах* нельзя использовать сумму квадратов ошибок, p–значения, статистику R^2 и другие традиционные показатели. Например, как мы видели на рис. 6.23, при $p > n$ не составляет труда получить модель с $R^2 = 1$. Это значение может заставить других людей ошибочно думать, что была найдена статистически корректная и полезная модель, хотя на самом деле это значение совершенно ничего не говорит о получении достойной внимания модели. Вместо этого важно привести результаты проверки на независимой контрольной выборке или ошибку, рассчитанную по результатам перекрестной проверки. Так, MSE или R^2, полученные на независимой контрольной выборке, являются правильными показателями качества модели, тогда как MSE на обучающей выборке совершенно точно таким показателем не является.

6.5 Лабораторная работа 1: методы отбора подмножеств переменных

6.5.1 Отбор оптимального подмножества

Здесь мы применим метод отбора оптимального подмножества переменных к данным `Hitters`. Мы хотим предсказать заработную плату бейс-

больного игрока (Salary) на основе различных показателей его результативности за предыдущий год.

Прежде всего обратим внимание на то, что для некоторых игроков значения переменной Salary отсутствуют. Для идентификации пропущенных значений можно воспользоваться функцией is.na(). Она возвращает вектор той же длины, что и входной вектор, со значениями TRUE для элементов с пропущенными значениями и FALSE для остальных элементов. Далее для подсчета отсутствующих значений можно применить функцию sum().

is.na()

sum()

```
> library(ISLR)
> fix(Hitters)
> names(Hitters)
[1]  "AtBat"    "Hits"     "HmRun"    "Runs"     "RBI"
[6]  "Walks"    "Years"    "CAtBat"   "CHits"    "CHmRun"
[11] "CRuns"    "CRBI"     "CWalks"   "League"   "Division"
[16] "PutOuts"  "Assists"  "Errors"   "Salary"   "NewLeague"
> dim(Hitters)
[1] 322 20
> sum(is.na(Hitters$Salary))
[1] 59
```

Таким образом, мы видим, что значения Salary отсутствуют для 59 игроков. Функция na.omit() удаляет все строки с отсутствующими значениями любой из переменных.

```
> Hitters = na.omit(Hitters)
> dim(Hitters)
[1] 263  20
> sum(is.na(Hitters))
[1] 0
```

regsubsets()

Функция regsubsets() (из библиотеки leaps) реализует отбор оптимального подмножества переменных и находит модель с оптимальным числом предикторов; при этом степень *оптимальности* определяется на основе RSS. Применяется тот же синтаксис, что и в случае с lm(). Команда summary() возвращает список оптимальных переменных для моделей каждого размера.

```
> library(leaps)
> regfit.full = regsubsets(Salary ~ ., Hitters)
> summary(regfit.full)
Subset selection object
Call: regsubsets.formula(Salary ~ ., Hitters)
19 Variables (and intercept)
...
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         AtBat Hits HmRun Runs RBI  Walks Years CAtBat CHits
1 ( 1 ) " "   " "  " "   " "  " "  " "   " "   " "    " "
2 ( 1 ) " "   "*"  " "   " "  " "  " "   " "   " "    " "
3 ( 1 ) " "   "*"  " "   " "  " "  " "   " "   " "    " "
4 ( 1 ) " "   "*"  " "   " "  " "  " "   " "   " "    " "
5 ( 1 ) "*"   "*"  " "   " "  " "  " "   " "   " "    " "
```

```
6 ( 1 ) "*"   "*"   " "   " "   " "   "*"   " "   " "    " "
7 ( 1 ) " "   "*"   " "   " "   " "   "*"   " "   "*"    "*"
8 ( 1 ) "*"   "*"   " "   " "   " "   "*"   " "   " "    " "
        CHmRun  CRuns  CRBI  CWalks  LeagueN  DivisionW  PutOuts
1 ( 1 ) " "     " "    "*"   " "     " "      " "        " "
2 ( 1 ) " "     " "    "*"   " "     " "      " "        " "
3 ( 1 ) " "     " "    "*"   " "     " "      " "        "*"
4 ( 1 ) " "     " "    "*"   " "     " "      "*"        "*"
5 ( 1 ) " "     " "    "*"   " "     " "      "*"        "*"
6 ( 1 ) " "     " "    "*"   " "     " "      "*"        "*"
7 ( 1 ) "*"     " "    " "   " "     " "      "*"        "*"
8 ( 1 ) "*"     "*"    " "   "*"     " "      "*"        "*"
        Assists  Errors  NewLeagueN
1 ( 1 ) " "      " "     " "
2 ( 1 ) " "      " "     " "
3 ( 1 ) " "      " "     " "
4 ( 1 ) " "      " "     " "
5 ( 1 ) " "      " "     " "
6 ( 1 ) " "      " "     " "
7 ( 1 ) " "      " "     " "
8 ( 1 ) " "      " "     " "
```

Звездочка указывает на то, что данная переменная включена в соответствующую модель. Например, из приведенных результатов следует, что наилучшая модель с двумя переменными включает только `Hits` и `CRBI`. По умолчанию `regsubsets()` выдает результаты для моделей, содержащих не более восьми переменных. Однако можно воспользоваться опцией `nvmax` и получить результаты для любого желаемого числа переменных. Ниже мы подгоняем модели, включающие вплоть до 19 переменных.

```
> regfit.full = regsubsets(Salary ~., data = Hitters, nvmax=19)
> reg.summary = summary(regfit.full)
```

Функция `summary()` возвращает также R^2, RSS, скорректированный коэффициент детерминации R^2, C_p и BIC. Мы можем исследовать эти критерии для отбора модели, оптимальной одновременно по нескольким из них.

```
> names(reg.summary)
[1] "which"  "rsq"  "rss"  "adjr2"  "cp"  "bic"
[7] "outmat" "obj"
```

Видно, например, что коэффициент детерминации R^2 возрастает с 32% в модели с одной переменной до почти 55% в модели, содержащей все переменные. Как и следовало ожидать, R^2 монотонно возрастает по мере добавления переменных.

```
> reg.summary$rsq
[1]  0.321 0.425 0.451 0.475 0.491 0.509 0.514 0.529 0.535
[10] 0.540 0.543 0.544 0.544 0.545 0.545 0.546 0.546 0.546
[19] 0.546
```

Изображение RSS, скорректированного коэффициента R^2, C_p и BIC одновременно для всех моделей поможет нам определиться с выбором оптимальной модели. Заметьте, что аргумент `type = "l"` сообщает программе о необходимости соединить изображенные точки линиями.

```
> par(mfrow = c(2, 2))
> plot(reg.summary$rss, xlab = "Number of Variables",
       ylab = "RSS", type="l")
> plot(reg.summary$adjr2, xlab = "Number of Variables",
       ylab = "Adjusted RSq", type = "l")
```

`points()` Функция `points()` работает подобно функции `plot()`, но вместо построения нового графика она добавляет точки к уже имеющемуся графику. Функцию `which.max()` можно использовать для нахождения порядкового номера максимального значения некоторого вектора. Здесь мы добавим красную точку для обозначения модели с наибольшим значением скорректированного коэффициента R^2:

```
> which.max(reg.summary$adjr2)
[1] 11
> points(11, reg.summary$adjr2[11], col="red", cex=2, pch=20)
```

`which.min()` Аналогичным образом мы можем изобразить значения критериев C_p и BIC, а с помощью `which.min()` отметить модели с наименьшими значениями этих критериев.

```
> plot(reg.summary$cp, xlab = "Number of Variables",
       ylab = "Cp", type = "l")
> which.min(reg.summary$cp)
[1] 10
> points(10, reg.summary$cp[10], col="red", cex=2, pch=20)
> which.min(reg.summary$bic)
[1] 6
> plot(reg.summary$bic, xlab ="Number of Variables",
       ylab = "BIC", type = "l")
> points(6, reg.summary$bic[6], col="red", cex=2, pch=20)
```

Функция `regsubsets()` имеет встроенный метод `plot()`, который можно применять для отображения переменных из оптимальных моделей заданного размера, упорядоченных в соответствии с BIC, C_p, скорректированным коэффициентом детерминации R^2 или AIC. Для получения дополнительной информации выполните команду `?plot.regsubsets`.

```
> plot(regfit.full, scale = "r2")
> plot(regfit.full, scale = "adjr2")
> plot(regfit.full, scale = "Cp")
> plot(regfit.full, scale = "bic")
```

В верхнем ряду каждого графика представлены черные прямоугольники, обозначающие предикторы, вошедшие в оптимальную модель в соответствии с тем или иным критерием качества. Мы видим, например, что несколько моделей обладают близкими значениями BIC (около -150). Однако модель с наименьшим значением BIC содержит шесть переменных: `AtBat`, `Hits`, `Walks`, `CRBI`, `DivisionW` и `PutOuts`. Мы можем применить функцию `coef()` для вывода оценок коэффициентов этой модели.

```
> coef(regfit.full, 6)
(Intercept)       AtBat        Hits       Walks        CRBI
     91.512      -1.869       7.604       3.698       0.643
  DivisionW     PutOuts
    -122.952       0.264
```

6.5.2 Отбор путем пошагового включения и исключения переменных

Мы можем также использовать функцию `regsubsets()` в сочетании с аргументом `method = "forward"` или `method = "backward"` для отбора переменных путем их пошагового включения или исключения.

```
> regfit.fwd = regsubsets(Salary ~ ., data = Hitters,
                          nvmax = 19, method = "forward")
> summary(regfit.fwd)
> regfit.bwd = regsubsets(Salary ~., data = Hitters,
                          nvmax = 19, method = "backward")
> summary(regfit.bwd)
```

Как видим, при использовании пошагового включения переменных оптимальная модель с одной переменной содержит только `CRBI`, а оптимальная модель с двумя переменными включает еще и `Hits`. Для этих данных оптимальные модели, содержащие от одного до шести предикторов, одинаковы как для метода отбора оптимального подмножества переменных, так и для метода пошагового включения переменных. Однако оптимальные модели с семью предикторами, найденные при помощи методов пошагового включения, пошагового исключения и отбора оптимального подмножества переменных, различаются.

```
> coef(regfit.full, 7)
(Intercept)        Hits       Walks      CAtBat       CHits
     79.451       1.283       3.227      -0.375       1.496
     CHmRun   DivisionW     PutOuts
      1.442    -129.987       0.237
> coef(regfit.fwd, 7)
(Intercept)       AtBat        Hits       Walks        CRBI
    109.787      -1.959       7.450       4.913       0.854
     CWalks   DivisionW     PutOuts
     -0.305    -127.122       0.253
> coef(regfit.bwd, 7)
(Intercept)       AtBat        Hits       Walks       CRuns
    105.649      -1.976       6.757       6.056       1.129
     CWalks   DivisionW     PutOuts
     -0.716    -116.169       0.303
```

6.5.3 Нахождение оптимальной модели при помощи методов проверочной выборки и перекрестной проверки

Как мы только что видели, оптимальную модель из нескольких кандидатов с разным числом предикторов можно выбрать при помощи C_p, BIC и скорректированного коэффициента детерминации R^2. Теперь мы рассмотрим, как это можно сделать при помощи методов проверочной выборки и перекрестной проверки.

Для того чтобы эти методы верно оценили ошибку на контрольной выборке, в ходе реализации всех аспектов моделирования (включая отбор переменных) мы должны использовать *только обучающие наблюдения*. Следовательно, определение оптимальной модели с тем или иным числом предикторов должно выполняться на основе *только обучающих наблюдений*. Это небольшая, но важная деталь. Если для нахождения оптимального подмножества переменных используется весь набор данных, то получаемые нами ошибки на проверочной выборке и ошибки перекрестной проверки будут неверными оценками ошибки на независимых контрольных данных.

Для реализации метода проверочной выборки мы сначала разбиваем наблюдения на обучающую и проверочную выборки. Для этого создается случайный вектор `train` с элементами `TRUE`, которые соответствуют наблюдениям из обучающей выборки, и `FALSE` — остальным наблюдениям. Элемент вектора `test` принимает значение `TRUE`, если наблюдение входит в проверочную выборку, и `FALSE` — в противоположном случае. Заметьте, что в команде, создающей вектор `test`, `!` обращает значения `TRUE` в `FALSE`, и наоборот. Чтобы пользователь получил идентичное разбиение на обучающую и проверочную выборки, мы также устанавливаем зерно генератора случайных чисел.

```
> set.seed(1)
> train = sample(c(TRUE, FALSE), nrow(Hitters), rep = TRUE)
> test = (!train)
```

Теперь мы применим `regsubsets()` к обучающей выборке для отбора оптимального подмножества переменных.

```
> regfit.best = regsubsets(Salary ~., data = Hitters[train, ],
                           nvmax = 19)
```

Заметьте, что мы отбираем обучающие данные из таблицы `Hitters` непосредственно при выполнении команды, используя выражение `Hitters[train, ]`. Теперь мы вычислим ошибку на проверочной выборке для оптимальной модели каждого размера. Сначала мы создадим матрицу модели на основе проверочных данных.

```
test.mat = model.matrix(Salary ~ ., data = Hitters[test, ])
```

`model.matrix()`

Функция `model.matrix()` применяется во многих регрессионных пакетах для создания т. н. «X–матрицы» данных. Теперь мы выполняем цикл, в котором для каждого числа предикторов `i` извлекаем коэффициенты модели с этим числом предикторов из объекта `regfit.best`, умножаем их на соответствующие столбцы матрицы модели для получения предсказаний и вычисляем MSE на проверочной выборке.

```
> val.errors = rep(NA, 19)
> for(i in 1:19){
+ coefi = coef(regfit.best, id = i)
+ pred = test.mat[, names(coefi)]%*%coefi
+ val.errors[i] = mean((Hitters$Salary[test] - pred)^2)
}
```

Как видим, оптимальной является модель с десятью предикторами.

```
> val.errors
[1] 220968 169157 178518 163426 168418 171271 162377 157909
[9] 154056 148162 151156 151742 152214 157359 158541 158743
[17] 159973 159860 160106
> which.min(val.errors)
[1] 10
> coef(regfit.best, 10)
(Intercept)    AtBat     Hits    Walks    CAtBat
    -80.275   -1.468    7.163    3.643    -0.186
      CHits   CHmRun   CWalks  LeagueN  DivisionW
      1.105    1.384   -0.748   84.558    -53.029
    PutOuts
      0.238
```

Эта процедура была несколько трудоемкой, отчасти из–за того, что для `regsubsets()` нет метода `predict()`. Поскольку мы будем применять эту функцию повторно, можно собрать приведенные выше шаги воедино и написать наш собственный метод.

```
> predict.regsubsets = function(object, newdata, id, ...){
+ form = as.formula(object$call[[2]])
+ mat = model.matrix(form, newdata)
+ coefi = coef(object, id = id)
+ xvars = names(coefi)
+ mat[, xvars]%*%coefi}
```

В принципе, наша функция подражает тому, что мы делали выше. Единственная сложная часть здесь — это извлечение формулы, используемой при вызове `regsubsets()`. Применение этой формулы будет продемонстрировано ниже при выполнении перекрестной проверки.

Наконец, мы выполняем отбор оптимального подмножества переменных с использованием полного набора данных и выбираем оптимальную модель с десятью переменными. Важно, что мы работаем с полным набором данных для получения более точных оценок коэффициентов. Заметьте, что мы не просто используем переменные, полученные на основе обучающих данных, а выбираем оптимальную модель с десятью переменными на основе полного набора данных, поскольку оптимальная модель с десятью переменными для полного набора данных может отличаться от таковой для обучающих данных.

```
> regfit.best = regsubsets(Salary ~., data=Hitters, nvmax=19)
> coef(regfit.best, 10)
(Intercept)    AtBat     Hits        Walks    CAtBat
    162.535   -2.169    6.918        5.773    -0.130
```

```
     CRuns      CRBI    CWalks   DivisionW   PutOuts    Assists
     1.408     0.774    -0.831    -112.380     0.297     0.283
```

И действительно, мы видим, что оптимальная модель с десятью переменными для полного набора данных содержит другие переменные, нежели оптимальная модель с десятью переменными для обучающих данных.

Теперь мы попробуем выбрать оптимальную модель из нескольких кандидатов с разным числом предикторов, используя перекрестную проверку. Это относительно сложный метод, поскольку мы должны выполнить отбор оптимального подмножества переменных *в пределах каждой из k обучающих выборок*. Однако благодаря своему «смышленому» синтаксису для разбиения объектов на части R делает эту задачу довольно простой. Сначала мы создаем вектор, который относит каждое наблюдение к одному из $k = 10$ блоков, и создаем матрицу, в которой будут храниться результаты.

```
> k = 10
> set.seed(1)
> folds = sample(1:k, nrow(Hitters), replace = TRUE)
> cv.errors=matrix(NA, k, 19, dimnames=list(NULL, paste(1:19)))
```

Теперь мы напишем for–цикл для выполнения перекрестной проверки. Элементы folds в j–м блоке, которые равны j, являются частью проверочной выборки, а остальные элементы входят в обучающую выборку. Мы делаем предсказания на основе модели каждого размера (используя наш новый метод predict()), вычисляем ошибки на проверочных выборках для каждого блока и сохраняем их в соответствующих ячейках матрицы cv.errors.

```
> for(j in 1:k){
+ best.fit = regsubsets(Salary ~ ., data = Hitters[folds != j, ],
                        nvmax =19)
+ for(i in 1:19) {
+ pred = predict(best.fit, Hitters[folds == j, ], id = i)
+ cv.errors[j, i] = mean((Hitters$Salary[folds == j] - pred)^2)
+ }
+ }
```

Это дало нам матрицу размером 10×19, в которой (i, j)–й элемент соответствует MSE на проверочной выборке i–го блока для оптимальной модели j–го размера. Мы можем применить функцию apply() для расчета средних значений по столбцам этой матрицы и получить вектор, в котором j–й элемент представляет собой ошибку перекрестной проверки для модели с j переменными.

```
> mean.cv.errors = apply(cv.errors, 2, mean)
> mean.cv.errors
[1] 160093 140197 153117 151159 146841 138303 144346 130208
[9] 129460 125335 125154 128274 133461 133975 131826 131883
[17] 132751 133096 132805
> par(mfrow = c(1, 1))
> plot(mean.cv.errors, type = "b")
```

Как видим, перекрестная проверка выбирает модель с 11 переменными. Теперь мы выполняем отбор оптимального подмножества переменных на полном наборе данных для получения модели с 11 переменными.

```
> reg.best = regsubsets(Salary ~., data = Hitters, nvmax = 19)
> coef(reg.best, 11)
(Intercept)    AtBat     Hits    Walks     CAtBat
    135.751   -2.128    6.924    5.620     -0.139
      CRuns     CRBI  CWalks  LeagueN  DivisionW
      1.455    0.785  -0.823   43.112   -111.146
    PutOuts  Assists
      0.289    0.269
```

6.6 Лабораторная работа 2: гребневая регрессия и лассо

Для выполнения гребневой регрессии и регрессии по методу лассо мы воспользуемся пакетом `glmnet`. Главной в этом пакете является функция `glmnet()`, которую можно использовать для подгонки моделей гребневой регрессии, лассо и многих других типов моделей. Ее синтаксис несколько отличается от других похожих функций, с которыми мы уже встретились в этой книге. В частности, мы должны отдельно подать на `glmnet()` матрицу `x` и вектор `y`, а не формулу `y ~ x`. Ниже мы построим модели гребневой регрессии и лассо для предсказания `Salary` по данным из таблицы `Hitters`. Сначала убедитесь, что пропущенные значения были удалены из данных подобно тому, как это описано в разделе 6.5.

glmnet

```
> x = model.matrix(Salary ~., Hitters)[ ,-1]
> y = Hitters$Salary
```

Для создания матрицы `x` особенно полезной является функция `model.matrix()` — она не только создает матрицу с 19 предикторами, но также автоматически преобразует любые качественные переменные в индикаторные. Последнее свойство является важным, поскольку `glmnet()` может принимать только количественные входные переменные.

6.6.1 Гребневая регрессия

Функция `glmnet()` имеет аргумент `alpha`, который определяет тип подгоняемой модели. Если `alpha = 0`, то подгоняется гребневая регрессионная модель, а если `alpha = 1`, то подгоняется лассо–модель. Сначала мы построим гребневую регрессионную модель.

```
> library(glmnet)
> grid = 10^seq(10, -2, length = 100)
> ridge.mod = glmnet(x, y, alpha = 0, lambda = grid)
```

По умолчанию функция `glmnet()` строит гребневую регрессионную модель для автоматически выбранного диапазона значений λ. Однако здесь мы решили применить эту функцию к интервалу значений от $\lambda = 10^{10}$ до $\lambda = 10^{-2}$, который, по сути, охватывает все возможные сценарии —

от нулевой модели, содержащей только свободный член, до модели, подогнанной по методу наименьших квадратов. Как будет показано ниже, мы можем также получить модель для некоторого конкретного значения λ, не входящего в исходный интервал значений `range`. Заметьте, что по умолчанию функция `glmnet()` выполняет стандартизацию переменных таким образом, чтобы они выражались в одинаковых единицах измерения. Для отключения этой настройки примените аргумент `standardize` со значением `FALSE`.

Каждому значению λ соответствует вектор коэффициентов гребневой регрессии в специальной матрице, доступ к которой можно получить при помощи `coef()`. Здесь эта матрица имеет размер 20×100, с 20 строками (для каждого предиктора плюс свободный член) и 100 столбцами (для каждого значения λ).

```
> dim(coef(ridge.mod))
[1]  20 100
```

Мы ожидаем, что при больших значениях λ оценки коэффициентов будут намного меньше (в смысле их нормы ℓ_2), чем при малых значениях λ. При $\lambda = 11\,498$ получены следующие коэффициенты и норма ℓ_2:

```
> ridge.mod$lambda[50]
[1] 11498
> coef(ridge.mod)[, 50]
(Intercept)       AtBat        Hits       HmRun        Runs
    407.356       0.037       0.138       0.525       0.231
        RBI       Walks       Years      CAtBat      CHits
      0.240       0.290       1.108       0.003       0.012
     CHmRun       CRuns        CRBI      CWalks    LeagueN
      0.088       0.023       0.024       0.025       0.085
  DivisionW     PutOuts     Assists      Errors NewLeagueN
     -6.215       0.016       0.003      -0.021       0.301
> sqrt(sum(coef(ridge.mod)[-1, 50]^2))
[1] 6.36
```

Ниже для сравнения приведены коэффициенты и соответствующая норма ℓ_2 для $\lambda = 705$. Обратите внимание на значительно более высокое значение нормы ℓ_2 у коэффициентов этой модели.

```
> ridge.mod$lambda[50]
[1] 705
> coef(ridge.mod)[, 50]
(Intercept)       AtBat        Hits       HmRun        Runs
     54.325       0.112       0.656       1.180       0.938
        RBI       Walks       Years      CAtBat      CHits
      0.847       1.320       2.596       0.011       0.047
     CHmRun       CRuns        CRBI      CWalks    LeagueN
      0.338       0.094       0.098       0.072      13.684
  DivisionW     PutOuts     Assists      Errors NewLeagueN
    -54.659       0.119       0.016      -0.704       8.612
> sqrt(sum(coef(ridge.mod)[-1, 60]^2))
[1] 57.1
```

Функцию predict() можно применять для разных целей. Так, мы можем получить коэффициенты гребневой регрессии для нового значения λ (например, 50):

```
> predict(ridge.mod, s = 50, type = "coefficients")[1:20, ]
(Intercept)       AtBat        Hits       HmRun        Runs
     48.766      -0.358       1.969      -1.278       1.146
        RBI       Walks       Years      CAtBat       CHits
      0.804       2.716      -6.218       0.005       0.106
     CHmRun       CRuns        CRBI      CWalks     LeagueN
      0.624       0.221       0.219      -0.150      45.926
  DivisionW     PutOuts     Assists      Errors NewLeagueN
   -118.201       0.250       0.122      -3.279      -9.497
```

Теперь мы разобьем наблюдения на обучающую и проверочную выборки и оценим ошибку на контрольных данных для гребневой регрессии и лассо–модели. Существует два распространенных способа случайного разбиения данных на части. Первый заключается в создании вектора с элементами TRUE и FALSE и отборе наблюдений со значением TRUE в состав обучающей выборки. Второй способ состоит в случайном отборе целых чисел из интервала от 1 до n, которые затем можно использовать в качестве индексных номеров для обучающих наблюдений. Эти два способа работают одинаково хорошо. В подразделе 6.5.3 мы применяли первый из них. Здесь мы продемонстрируем второй способ.

Сначала мы устанавливаем значение зерна генератора случайных чисел для получения воспроизводимых результатов.

```
> set.seed(1)
> train = sample(1:nrow(x), nrow(x)/2)
> test = (-train)
> y.test = y[test]
```

Далее мы подгоняем гребневую регрессионную модель по обучающей выборке и оцениваем ее MSE на проверочной выборке, используя $\lambda = 4$. Заметьте, что мы снова применяем функцию predict(). Однако в этот раз при получении предсказаний для проверочной выборки мы заменяем type = "coefficients" на аргумент newx.

```
> ridge.mod = glmnet(x[train, ], y[train], alpha = 0,
                     lambda = grid, thresh = 1e-12)
> ridge.pred = predict(ridge.mod, s = 4, newx = x[test, ])
> mean((ridge.pred - y.test)^2)
[1] 101037
```

MSE на контрольной выборке составляет 101 037. Заметьте, что если бы мы построили модель, содержащую только свободный член, то каждое предсказанное значение было бы равно среднему значению обучающих наблюдений. При таком сценарии мы рассчитали бы MSE на контрольной выборке следующим образом:

```
> mean((mean(y[train]) - y.test)^2)
[1] 193253
```

Идентичный результат можно было бы получить путем подгонки гребневой регрессионной модели с *очень* большим значением λ. Заметьте, что `1e10` означает 10^{10}.

```
> ridge.pred = predict(ridge.mod, s = 1e10, newx = x[test, ])
> mean((ridge.pred - y.test)^2)
[1] 193253
```

Таким образом, гребневая регрессионная модель с $\lambda = 4$ имеет существенно более низкую MSE, чем модель, включающая только свободный член. Сейчас мы проверим, отличается ли качество предсказаний гребневой регрессии с $\lambda = 4$ от такового у обычной линейной регрессии. Вспомните, что линейная регрессия по методу наименьших квадратов — это просто гребневая регрессия с $\lambda = 0$[22].

```
> ridge.pred = predict(ridge.mod, s = 0, newx = x[test, ],
                       exact = TRUE)
> mean((ridge.pred - y.test)^2)
[1] 114783
> lm(y ~ x, subset = train)
> predict(ridge.mod, s = 0, exact = TRUE,
          type ="coefficients")[1:20, ]
```

В целом если мы хотим построить модель по методу наименьших квадратов (т. е. без наложения штрафа на оценки коэффициентов), то нам следует использовать функцию `lm()`, поскольку она выдает более полезные результаты (например, стандартные ошибки и p–значения коэффициентов).

Как правило, вместо произвольного выбора значения λ более оптимальным подходом для нахождения этого гиперпараметра является применение перекрестной проверки. Для выполнения такой перекрестной проверки можно воспользоваться встроенной функцией `cv.glmnet()`. По умолчанию эта функция выполняет десятикратную перекрестную проверку, однако это можно изменить при помощи аргумента `nfolds`. Обратите внимание: поскольку формирование блоков в ходе перекрестной проверки выполняется случайным образом, то для получения воспроизводимых результатов мы сначала задаем зерно генератора случайных чисел.

`cv.glmnet()`

```
> set.seed(1)
> cv.out = cv.glmnet(x[train, ], y[train], alpha = 0)
> plot(cv.out)
> bestlam = cv.out$lambda.min
> bestlam
[1] 212
```

Таким образом, мы видим, что значение λ, обеспечивающее минимальную ошибку перекрестной проверки, составляет 212. Чему равна среднеквадратичная ошибка на контрольных данных для этого значения λ?

[22] Для того чтобы функция `glmnet()` в точности дала решение по методу наименьших квадратов, при вызове `predict()` мы применяем аргумент `exact = TRUE`. Иначе функция `predict()` выполнит интерполяцию по всем значениям λ, использованным при подгонке `glmnet`–модели, и даст приблизительные результаты. При использовании аргумента `exact = TRUE` результаты, возвращаемые `lm()` и `glmnet()` с $\lambda = 0$, незначительно различаются в третьем знаке после запятой, что обусловлено числовыми приближениями, реализованными в функции `glmnet()`.

```
> ridge.pred = predict(ridge.mod, s = bestlam, newx = x[test, ])
> mean((ridge.pred - y.test)^2)
[1] 96016
```

Это значение MSE даже лучше того, которое мы получили для $\lambda = 4$. Наконец, мы подгоняем гребневую регрессионную модель на основе полного набора данных со значением λ, найденным путем перекрестной проверки, и просматриваем оценки коэффициентов.

```
> out = glmnet(x, y, alpha = 0)
> predict(out, type = "coefficients", s = bestlam )[1:20, ]
(Intercept)       AtBat        Hits       HmRun        Runs
     9.8849      0.0314      1.0088      0.1393      1.1132
        RBI       Walks       Years      CAtBat       CHits
     0.8732      1.8041      0.1307      0.0111      0.0649
     CHmRun       CRuns        CRBI       CWalks     LeagueN
     0.4516      0.1290      0.1374      0.0291     27.1823
  DivisionW     PutOuts     Assists      Errors NewLeagueN
   -91.6341      0.1915      0.0425     -1.8124      7.2121
```

Как и следовало ожидать, ни один из коэффициентов не равен нулю — гребневая регрессия не выполняет отбора переменных!

6.6.2 Лассо

Как мы выяснили, подогнанная к данным `Hitters` гребневая регрессия с хорошо подобранным значением λ может по своему качеству превзойти модель наименьших квадратов и нулевую модель. Теперь мы выясним, может ли метод лассо дать более точную и легче интерпретируемую модель, чем метод гребневой регрессии. Для подгонки лассо–модели мы снова применяем функцию `glmnet()`, однако на этот раз в сочетании с аргументом `alpha = 1`. За исключением этой разницы, мы действуем так же, как и при подгонке гребневой модели.

```
> lasso.mod = glmnet(x[train, ], y[train], alpha = 1,
                     lambda = grid)
> plot(lasso.mod)
```

На графике коэффициентов видно, что в зависимости от выбранного значения гиперпараметра некоторые коэффициенты будут в точности равны нулю. Теперь мы выполним перекрестную проверку и вычислим соответствующую ошибку предсказаний.

```
> set.seed(1)
> cv.out = cv.glmnet(x[train, ], y[train], alpha = 1)
> plot(cv.out)
> bestlam = cv.out$lambda.min
> lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test, ])
> mean((lasso.pred - y.test)^2)
[1] 100743
```

Это значение намного ниже, чем MSE на контрольной выборке у нулевой модели и модели наименьших квадратов, но очень похоже на ошибку

гребневой регрессии с параметром λ, выбранным путем перекрестной проверки.

Тем не менее метод лассо имеет существенное преимущество перед методом гребневой регрессии благодаря небольшому набору получаемых коэффициентов. В этом примере мы видим, что 12 из 19 оценок коэффициентов в точности равны нулю. Таким образом, лассо–модель со значением λ, выбранным при помощи перекрестной проверки, включает только семь переменных.

```
> out = glmnet(x, y, alpha = 1, lambda = grid)

> lasso.coef = predict(out, type = "coefficients",
                       s = bestlam)[1:20, ]
> lasso.coef
(Intercept)       AtBat        Hits       HmRun        Runs
     18.539       0.000       1.874       0.000       0.000
        RBI       Walks       Years      CAtBat       CHits
      0.000       2.218       0.000       0.000       0.000
     CHmRun       CRuns        CRBI      CWalks     LeagueN
      0.000       0.207       0.413       0.000       3.267
  DivisionW     PutOuts     Assists      Errors NewLeagueN
   -103.485       0.220       0.000       0.000       0.000

> lasso.coef[lasso.coef != 0]
(Intercept)        Hits       Walks       CRuns        CRBI
     18.539       1.874       2.218       0.207       0.413
   LeagueN   DivisionW     PutOuts
     3.267    -103.485       0.220
```

6.7 Лабораторная работа 3: регрессия при помощи методов PCR и PLS

6.7.1 Регрессия на главные компоненты

pcr() Регрессию на главные компоненты (PCR) можно выполнить при помощи функции `pcr()`из библиотеки `pls`. Здесь мы применим метод PCR к данным `Hitters` для предсказания `Salary`. Как и раньше, убедитесь, что пропущенные значения были удалены из данных так, как это описано в разделе 6.5.

```
> library(pls)
> set.seed(2)
> pcr.fit = pcr(Salary ~ ., data = Hitters, scale = TRUE,
                validation = "CV")
```

Синтаксис функции `pcr()` похож на синтаксис `lm()`, но включает несколько дополнительных опций. Аргумент `scale = TRUE` вызывает *стандартизацию* каждого предиктора перед вычислением главных компонент (в соответствии с (6.6)), что позволяет избежать влияния шкалы, на которой измерена та или иная переменная. При помощи `validation = "CV"` можно

заставить pcr() вычислить ошибку десятикратной перекрестной проверки для каждого возможного значения M, т. е. числа используемых главных компонент. Детали итоговой модели можно изучить при помощи команды summary().

```
> summary(pcr.fit)
Data:   X dimension: 263 19
        Y dimension: 263 1
Fit method: svdpc
Number of components considered: 19

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps
CV             452    348.9    352.2    353.5    352.8
adjCV          452    348.7    351.8    352.9    352.1
...

TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X         38.31    60.16    70.84    79.03    84.29    88.63
Salary    40.63    41.58    42.17    43.22    44.90    46.48
...
```

Ошибка перекрестной проверки приведена для каждого возможного числа главных компонент — от $M = 0$ и выше. (Мы показали результаты только для $M = 4$.) Заметьте, что pcr() выводит значения *квадратного корня из среднеквадратичной ошибки*; для получения обычных MSE нужно возвести эти значения в квадрат. Например, квадратный корень из MSE, равный 352.8, соответствует $\text{MSE} = 352.8^2 = 124\,468$.

Мы можем также изобразить ошибки перекрестной проверки графически при помощи функции validationplot(). Использование аргумента val.type = "MSEP" приведет к изображению значений MSE.

validation plot()

```
> validationplot(pcr.fit, val.type = "MSEP")
```

Как видим, минимальная ошибка перекрестной проверки наблюдается при использовании $M = 16$ компонент. Это лишь немногим меньше $M = 19$ — числа компонент, соответствующего выполнению регрессии по методу наименьших квадратов, поскольку при использовании всех компонент снижения размерности не происходит. Однако на этом графике наименьшая ошибка перекрестной проверки мало отличается от ошибки, наблюдаемой при включении в модель только одной компоненты. Это говорит о том, что удовлетворительной может оказаться модель с некоторым небольшим числом компонент.

Функция summary() выдает также *долю объясненной дисперсии* предикторов и отклика в зависимости от числа компонент. Эта идея обсуждается более подробно в главе 10. Вкратце: мы можем думать об этой величине как о количестве информации о предикторах и отклике, заключенной в модели с M главными компонентами. Например, использование $M = 1$ позволяет объяснить только 38.31% всей дисперсии, или информации, в предикторах. В то же время использование $M = 6$ увеличивает это

значение до 88.63%. Если бы мы использовали все $M = p = 19$ компонент, то оно возросло бы до 100%.

Теперь мы выполним PCR на обучающей выборке и оценим точность предсказаний на контрольной выборке.

```
> set.seed(1)
> pcr.fit = pcr(Salary ~., data = Hitters, subset = train,
                scale = TRUE, validation = "CV")
> validationplot(pcr.fit, val.type ="MSEP")
```

Теперь мы видим, что наименьшая ошибка перекрестной проверки имеет место при использовании $M = 7$ компонент. MSE на контрольной выборке вычисляется следующим образом:

```
> pcr.pred = predict(pcr.fit, x[test, ], ncomp = 7)
> mean((pcr.pred - y.test)^2)
[1] 96556
```

Это значение MSE сопоставимо с результатами, полученными при помощи методов гребневой регрессии и лассо. Однако в силу своего устройства метод PCR дает сложнее интерпретируемую модель, поскольку он не выполняет никакого отбора переменных и даже не выдает оценки коэффициентов.

Наконец, мы подгоняем PCR–модель для всего набора данных, используя $M = 7$ — число компонент, найденное путем перекрестной проверки.

```
> pcr.fit = pcr(y ~ x, scale = TRUE, ncomp = 7)
> summary(pcr.fit)

Data:   X dimension: 263 19
        Y dimension: 263 1
Fit method: svdpc
Number of components considered: 7
TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X         38.31    60.16    70.84    79.03    84.29    88.63
y         40.63    41.58    42.17    43.22    44.90    46.48
        7 comps
X         92.26
y         46.69
```

6.7.2 Регрессия по методу частных наименьших квадратов

Мы выполняем регрессию по методу наименьших квадратов (PLS) при помощи функции `plsr()`, также входящей в состав библиотеки `pcr`.

```
> set.seed(1)
> pls.fit = plsr(Salary ~ ., data = Hitters, subset = train,
                 scale = TRUE, validation = "CV")
> summary(pls.fit)
Data:   X dimension: 131 19
        Y dimension: 131 1
```

```
Fit method: kernelpls
Number of components considered: 19

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps
CV           464.6    394.2    391.5    393.1    395.0
adjCV        464.6    393.4    390.2    391.1    392.9
...

TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X         38.12    53.46    66.05    74.49    79.33    84.56
Salary    33.58    38.96    41.57    42.43    44.04    45.59
...
> validationplot(pls.fit, val.type = "MSEP")
```

Наименьшая ошибка перекрестной проверки наблюдается при использовании только $M = 2$ компонент частных наименьших квадратов. Теперь мы вычислим соответствующую MSE на контрольной выборке.

```
> pls.pred = predict(pls.fit, x[test, ], ncomp = 2)
> mean((pls.pred - y.test)^2)
[1] 101417
```

Эта MSE на контрольной выборке сравнима (хотя и несколько превышает их) с ошибками гребневой регрессии, лассо и PCR.

Наконец, мы строим PLS–модель для полного набора данных с применением $M = 2$, т. е. оптимального числа компонент, найденного при помощи перекрестной проверки.

```
> pls.fit = plsr(Salary ~., data = Hitters,
                 scale = TRUE, ncomp = 2)
> summary(pls.fit)

Data:   X dimension: 263 19
        Y dimension: 263 1
Fit method: kernelpls
Number of components considered: 2
TRAINING : % variance explained
         1 comps  2 comps
X          38.08    51.03
Salary     43.05    46.40
```

Обратите внимание на то, что доля дисперсии `Salary`, объясненная PLS–моделью с двумя компонентами (46.40%) почти совпадает с долей дисперсии, объясненной итоговой PCR–моделью с семью компонентами (46.69%). Это обусловлено тем, что PCR пытается максимизировать долю объясненной дисперсии предикторов, тогда как PLS выполняет поиск в направлениях, объясняющих дисперсию как предикторов, так и отклика.

6.8 Упражнения

Теоретические

1. Мы применяем методы отбора оптимального подмножества, пошагового включения и пошагового исключения переменных к некоторому набору данных. Для каждого из этих методов мы получаем $p+1$ моделей, содержащих $0, 1, 2, \ldots, p$ предикторов. Объясните свои ответы на следующие вопросы:

   (a) Какая из трех моделей с k предикторами имеет наименьшую RSS на *обучающей* выборке?

   (b) Какая из трех моделей с k предикторами имеет наименьшую RSS на *контрольной* выборке?

   (c) Верны ли следующие утверждения?

      i. Предикторы в модели с k переменными, найденной при помощи метода пошагового включения переменных, представляют собой подмножество предикторов в модели с $(k+1)$ переменными, найденной при помощи того же метода.

      ii. Предикторы в модели с k переменными, найденной при помощи метода пошагового исключения переменных, представляют собой подмножество предикторов в модели с $(k+1)$ переменными, найденной при помощи того же метода.

      iii. Предикторы в модели с k переменными, найденной при помощи метода пошагового исключения переменных, представляют собой подмножество предикторов в модели с $(k+1)$ переменной, найденной методом пошагового включения переменных.

      iv. Предикторы в модели с k переменными, найденной при помощи метода пошагового включения переменных, представляют собой подмножество предикторов в модели с $(k+1)$ переменной, найденной методом пошагового исключения переменных.

      v. Предикторы в модели с k переменными, найденной путем отбора оптимального подмножества переменных, представляют собой подмножество предикторов в модели с $(k+1)$ переменной, найденной при помощи того же метода.

2. Какие из вариантов ответов (i)–(iv) верны для вопросов (a)–(c)? Объясните свой ответ.

   (a) По сравнению с методом наименьших квадратов, метод лассо является:

      i. Более гибким и, следовательно, будет давать более точные предсказания в случае, когда увеличение его смещения меньше, чем снижение его дисперсии.

      ii. Более гибким и, следовательно, будет давать более точные предсказания в случае, когда увеличение его дисперсии меньше, чем снижение его смещения.

iii. Менее гибким и, следовательно, будет давать более точные предсказания в случае, когда увеличение его смещения меньше, чем снижение его дисперсии.

iv. Менее гибким и, следовательно, будет давать более точные предсказания в случае, когда увеличение его дисперсии меньше, чем снижение его смещения.

(b) Повторите (a) для гребневой регрессии, сравнив ее с методом наименьших квадратов.

(c) Повторите (a) для нелинейных методов, сравнив их с методом наименьших квадратов.

3. Предположим, что мы оцениваем коэффициенты линейной регрессионной модели, минимизируя

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \quad \text{при условии, что} \quad \sum_{j=1}^{p}|\beta_j|\le s$$

для некоторого значения s. Какие из вариантов ответов (i)–(v) верны для задач (a)–(e)? Объясните свой ответ.

(a) При увеличении s от 0 и выше RSS на обучающих данных:

i. Сначала возрастет, а затем в определенный момент начнет снижаться, образуя перевернутую U–образную кривую.

ii. Сначала снизится, а затем в определенный момент начнет возрастать, образуя U–образную кривую.

iii. Будет постоянно возрастать.

iv. Будет постоянно снижаться.

v. Останется неизменной.

(b) Повторите (a) для RSS на контрольных данных.

(c) Повторите (a) для дисперсии.

(d) Повторите (a) для (квадрата) смещения.

(e) Повторите (a) для неустранимой ошибки.

4. Предположим, что мы оцениваем коэффициенты линейной регрессионной модели, минимизируя

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

для некоторого конкретного значения λ. Какие из вариантов ответов (i)–(v) верны для задач (a)–(e)? Объясните свой ответ.

(a) При увеличении λ от 0 и выше RSS на обучающих данных:

i. Сначала возрастет, а затем в определенный момент начнет снижаться, образуя перевернутую U–образную кривую.

ii. Сначала снизится, а затем в определенный момент начнет возрастать, образуя U–образную кривую.

iii. Будет постоянно возрастать.

iv. Будет постоянно снижаться.

v. Останется неизменной.

(b) Повторите (a) для RSS на контрольных данных.

(c) Повторите (a) для дисперсии.

(d) Повторите (a) для (квадрата) смещения.

(e) Повторите (a) для неустранимой ошибки.

5. Хорошо известно, что гребневая регрессия имеет тенденцию присваивать похожие значения коэффициентам коррелирующих переменных, тогда как метод лассо может присваивать довольно разные значения коэффициентам таких переменных. Сейчас мы рассмотрим это свойство для очень простого случая.

Предположим, что $n = 2$, $p = 2$, $x_{11} = x_{12}$, $x_{21} = x_{22}$. Более того, предположим, что $y_1 + y_2 = 0$, $x_{11} + x_{21} = 0$ и $x_{12} + x_{22} = 0$, в связи с чем оценка коэффициента свободного члена в моделях, построенных по методу наименьших квадратов, методу гребневой регрессии и методу лассо, равна нулю: $\hat{\beta}_0 = 0$.

(a) Напишите для этого случая уравнение оптимизации по методу гребневой регрессии.

(b) Покажите, что в этом случае оценки коэффициентов гребневой регрессии удовлетворяют условию $\hat{\beta}_1 = \hat{\beta}_2$.

(c) Напишите для этого случая уравнение оптимизации по методу лассо.

(d) Покажите, что в этом случае оценки коэффициентов лассо–модели $\hat{\beta}_1$ и $\hat{\beta}_2$ не являются уникальными, т. е. имеется большое число возможных решений оптимизационной проблемы из (c). Опишите эти решения.

6. Теперь мы исследуем подробнее уравнения (6.12) и (6.13).

(a) Рассмотрим (6.12) для случая с $p = 1$. Для некоторых произвольных значений y_1 и $\lambda > 0$ изобразите (6.12) как функцию от β_1. Ваш график должен подтвердить, что (6.12) решается при помощи (6.14).

(b) Рассмотрим (6.13) для случая с $p = 1$. Для некоторых произвольных значений y_1 и $\lambda > 0$ изобразите (6.13) как функцию от β_1. Ваш график должен подтвердить, что (6.13) решается при помощи (6.15).

7. Сейчас мы продемонстрируем связь методов лассо и гребневой регрессии с идеями байесовской статистики, обсуждавшуюся в подразделе 6.2.2.

(a) Предположим, что $y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i$, где $\epsilon_1, \ldots, \epsilon_n$ являются независимыми и одинаково распределенными величинами, происходящими из нормального распределения $N(0, \sigma^2)$. Запишите уравнение правдоподобия для данных.

(b) Допустим, что для β верна следующая априорная информация: $\beta_1, \ldots, \beta_p$ являются независимыми и одинаково распределенными величинами, происходящими из двойного экспоненциального распределения со средним значением, равным 0, и общим параметром масштаба b, т. е. $p(\beta) = \frac{1}{2b}\exp(-|\beta|/b)$. Запишите апостериорную функцию плотности вероятности β для этого сценария.

(c) Покажите, что оценка β по методу лассо представляет собой *моду* этого апостериорного распределения.

(d) Теперь допустим, что для β верна следующая априорная информация: $\beta_1, \ldots, \beta_p$ являются независимыми и одинаково распределенными величинами, происходящими из нормального распределения со средним значением, равным 0, и дисперсией c. Запишите апостериорную функцию плотности вероятности β для этого сценария.

(e) Покажите, что оценка β по методу гребневой регрессии представляет собой как *моду*, так и *среднее значение* этого апостериорного распределения.

Практические

8. В этом упражнении мы создадим искусственные данные, а затем выполним для них отбор оптимального подмножества переменных.

(a) Примените функцию `rnorm()` для создания предиктора X длиной $n = 100$, а также вектора остатков ϵ длиной $n = 100$.

(b) Создайте вектор отклика Y длиной $n = 100$ в соответствии с моделью

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon,$$

где $\beta_0, \beta_1, \beta_2$ и β_3 — произвольно выбранные вами константы.

(c) Используя функцию `regsubsets()`, выполните отбор оптимального подмножества переменных для нахождения оптимальной модели с предикторами $X, X^2, \ldots, X^{10}$. Какая модель оказывается оптимальной, согласно C_p, BIC и скорректированному коэффициенту детерминации R^2? Постройте графики, подтверждающие ваш ответ, и приведите коэффициенты полученной оптимальной модели. Обратите внимание: для создания таблицы данных, содержащей как X, так и Y, вам потребуется функция `data.frame()`.

(d) Повторите (c), используя методы пошагового включения и пошагового исключения переменных. Как ваши ответы для этих методов отличаются от результатов в (c)?

(e) Теперь подгоните к этим имитированным данным лассо–модель, снова используя в качестве предикторов $X, X^2, \ldots, X^{10}$. Выполните перекрестную проверку для нахождения оптимального значения λ. Постройте график зависимости ошибки перекрестной проверки от λ. Приведите итоговые оценки коэффициентов и обсудите полученные результаты.

(f) Теперь создайте вектор отклика Y в соответствии с моделью

$$Y = \beta_0 + \beta_7 X^7 + \epsilon$$

и примените методы отбора оптимального подмножества переменных и лассо. Обсудите полученные результаты.

9. В этом упражнении мы построим модель для предсказания числа заявлений от абитуриентов на основе переменных из набора данных `College`.

(a) Разбейте данные на обучающую и контрольную выборки.

(b) Постройте линейную модель по методу наименьших квадратов, используя обучающую выборку, и сообщите полученную ошибку на контрольной выборке.

(c) Постройте гребневую регрессионную модель по обучающей выборке со значением λ, выбранным при помощи перекрестной проверки. Сообщите полученную ошибку на контрольной выборке.

(d) Постройте лассо–модель по обучающей выборке со значением λ, выбранным при помощи перекрестной проверки. Сообщите полученную ошибку на контрольной выборке, а также количество ненулевых оценок коэффициентов.

(e) Постройте PCR–модель по обучающей выборке со значением M, выбранным при помощи перекрестной проверки. Сообщите полученную ошибку на контрольной выборке, а также выбранное путем перекрестной проверки значение M.

(f) Постройте PLS–модель по обучающей выборке со значением M, выбранным при помощи перекрестной проверки. Сообщите полученную ошибку на контрольных данных, а также выбранное путем перекрестной проверки значение M.

(g) Прокомментируйте полученные результаты. Насколько точно мы можем предсказать число заявлений от абитуриентов? Есть ли заметная разница в значениях ошибок на контрольной выборке, полученных при помощи этих пяти методов?

10. Ранее мы видели, что при увеличении числа включенных в модель переменных ошибка на обучающей выборке всегда будет снижаться, тогда как ошибка на контрольной выборке не обязательно будет вести себя тем же образом. Сейчас мы рассмотрим это свойство на примере имитированных данных.

(a) Создайте набор данных с $p = 20$ переменными, $n = 1000$ наблюдениями и соответствующим вектором количественного отклика, сгенерированным согласно модели

$$Y = X\beta + \epsilon,$$

где некоторые элементы β в точности равны нулю.

(b) Разбейте свои данные на обучающую выборку со 100 наблюдениями и контрольную выборку с 900 наблюдениями.

(c) Примените метод отбора оптимального подмножества переменных к обучающей выборке и постройте график зависимости MSE на обучающей выборке от размера оптимальной модели каждого размера.

(d) Постройте график зависимости MSE на контрольной выборке от размера оптимальной модели.

(e) Для модели какого размера MSE на контрольной выборке принимает наименьшее значение? Прокомментируйте свои результаты. Если ошибка принимает наименьшее значение для модели, содержащей только свободный член, или для модели, включающей все переменные, тогда попробуйте несколько разных способов создания данных в (a) до тех пор, пока вы не получите ситуацию, в которой MSE на контрольной выборке достигает минимума для модели некоторого промежуточного размера.

(f) Каковы отличия модели с наименьшим значением MSE на контрольной выборке от истинной модели, использованной для создания данных? Прокомментируйте полученные значения коэффициентов.

(g) Постройте график, изображающий $\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ для нескольких значений r, где $\hat{\beta}_j^r$ — это оценка j–го коэффициента оптимальной модели, содержащей r коэффициентов. Прокомментируйте свои наблюдения. Похож ли этот график на график MSE на контрольной выборке из (d)?

11. Сейчас мы постараемся предсказать уровень преступности в расчете на душу населения по данным `Boston`.

(a) Попробуйте применить некоторые из рассмотренных в этой главе регрессионных методов — например, отбор оптимального подмножества переменных, лассо, гребневую регрессию и PCR. Представьте и обсудите полученные результаты.

(b) Предложите модель (или несколько моделей), которая хорошо работает для этого набора данных, и объясните свой выбор. Убедитесь, что вы оцениваете качество модели при помощи проверочной выборки, перекрестной проверки или какого–либо другого подходящего метода, но не при помощи ошибки на обучающей выборке.

(c) Включает ли выбранная вами модель все имеющиеся в данных переменные? Почему?

Глава 7

Выходя за пределы линейности

До сих пор в этой книге мы в основном обсуждали линейные модели. Линейные модели относительно просто описать и построить, и они имеют преимущества в сравнении с другими подходами благодаря интерпретируемости и возможности сделать статистические выводы. Однако у стандартной линейной регрессии могут быть существенные недостатки, связанные с ее предсказательной силой. Это обусловлено тем, что предположение о линейности почти всегда является приближением, и иногда очень плохим приближением. Как было показано в главе 6, мы можем улучшить результаты, получаемые при помощи метода наименьших квадратов, применяя гребневую регрессию, метод лассо, регрессию на главные компоненты и другие подходы. Улучшение при использовании этих методов достигается путем упрощения линейной модели, а следовательно, благодаря снижению дисперсии оценок коэффициентов. Однако мы все еще используем линейную модель, которую дальше никак не улучшить! В этой главе мы ослабим допущение о линейности, пытаясь, однако, сохранить модель настолько интерпретируемой, насколько это возможно. Мы рассмотрим очень простые расширения линейной модели, такие как полиномиальная регрессия и ступенчатые функции, а также более сложные подходы, такие как сплайны, локальная регрессия и обобщенные аддитивные модели.

- *Полиномиальная регрессия* расширяет линейную модель введением дополнительных предикторов, полученных путем возведения каждого исходного предиктора в определенную степень. Например, *кубическая* регрессия использует три переменные в качестве предикторов — X, X^2 и X^3. Этот подход обеспечивает простой способ подгонки нелинейной модели к данным.

- *Ступенчатые функции* разбивают интервал значений некоторой переменной на K непересекающихся областей для создания качественной переменной. Это имеет эффект подгонки кусочно–постоянной функции.

- *Регрессионные сплайны* являются более гибкими, чем полиномы и ступенчатые функции, и, более того, представляют собой расширение этих двух методов. Они сводятся к разбиению интервала зна-

чений X на K непересекающихся областей. В пределах каждой области к данным подгоняется полиномиальная функция. Однако эти полиномы ограничены таким образом, чтобы происходило их гладкое соединение на границах областей, или в *узлах сочленения*. При условии, что интервал разбит на достаточное число областей, это может привести к формированию очень гибкой модели.

- *Сглаживающие сплайны* похожи на регрессионные сплайны, однако возникают в несколько иной ситуации. Их получают путем минимизации суммы квадратов остатков при одновременном введении штрафа на гладкость модели.
- *Локальная регрессия* похожа на сплайны, однако имеет одно важное отличие. Областям значений предиктора разрешено перекрываться, и это происходит очень гладким образом.
- *Обобщенные линейные модели* позволяют нам расширить перечисленные выше методы для работы с несколькими предикторами.

В разделах 7.1–7.6 мы представляем несколько подходов для описания связи между откликом Y и единственным предиктором X при помощи той или иной гибкой модели. В разделе 7.7 мы показываем, что эти подходы можно легко интегрировать для моделирования отклика Y по нескольким предикторам $X_1, \ldots, X_p$.

7.1 Полиномиальная регрессия

Исторически сложилось так, что стандартный способ расширения линейной регрессии на случаи, в которых зависимость между предикторами и откликом является нелинейной, сводится к замене стандартной линейной модели

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

на полиномиальную функцию

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d + \epsilon_i, \tag{7.1}$$

где ϵ_i — это остатки модели. Этот подход известен как *полиномиальная регрессия*, и более того — мы уже видели пример его применения в подразделе 3.3.2. Для достаточно большой степени d полиномиальная регрессия позволяет получить очень извилистую кривую. Заметьте, что коэффициенты в (7.1) можно оценить методом наименьших квадратов, поскольку это не что иное, как стандартная линейная модель с предикторами $x_i, x_i^2, x_i^3, \ldots, x_i^d$. Как правило, степень d выше 3 или 4 не используется, поскольку для больших значений d полиномиальная кривая может стать чрезмерно гибкой и принять довольно странный вид. Особенно часто это наблюдается на границах интервала значений переменной X.

полиномиальная регрессия

На рис. 7.1 слева показан график зависимости заработной платы (`wage`) от возраста (`age`) по данным из таблицы `Wage`, которая содержит информацию по доходу и демографическим характеристикам мужчин, проживающих в центрально–атлантическом регионе США. Мы видим результат подгонки полинома 4–й степени по методу наименьших квадратов (сплошная

синяя кривая). Несмотря на то что это линейная регрессионная модель, отдельные ее коэффициенты особого интереса не представляют. Вместо этого мы смотрим на подогнанную функцию сразу для всех 62 значений age, изменяющихся от 18 до 80 лет, с целью понять характер этой зависимости.

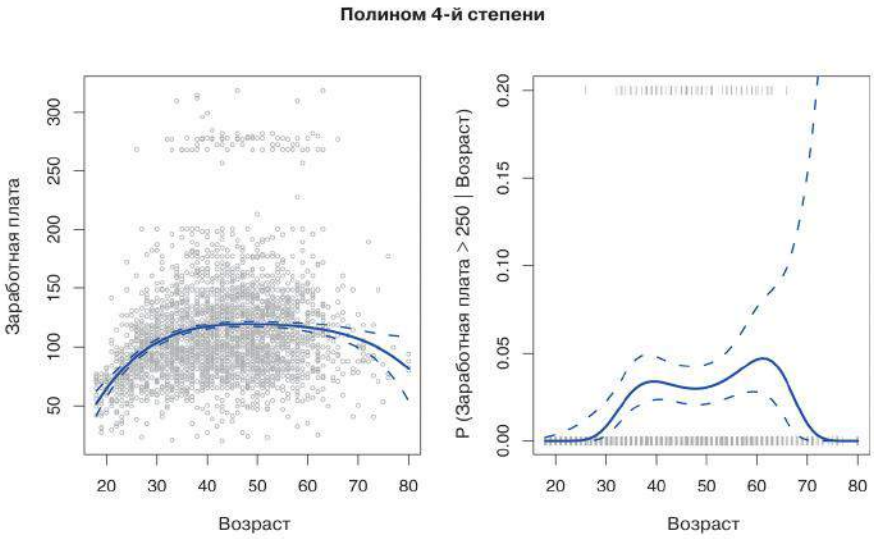

РИСУНОК 7.1. *Данные* Wage. Слева: *сплошная синяя линия представляет собой подогнанный по методу наименьших квадратов полином 4–й степени, который описывает зависимость* wage *(заработная плата, тыс. долларов) от* age (возраст, лет). *Прерывистые кривые показывают оцененный 95%–ный доверительный интервал.* Справа: *используя логистическую полиномиальную регрессии 4–й степени, мы моделируем вероятность того, что* wage > 250. *Оцененная апостериорная вероятность того, что* wage *превышает 250 000$, показана в виде синей кривой с соответствующим 95%–ным доверительным интервалом*

На рис. 7.1 подогнанная модель сопровождается парой прерывистых кривых, которые соответствуют двум стандартным ошибкам. Давайте выясним, как они были получены. Предположим, что мы вычислили модельное значение отклика для x_0 — некоторого конкретного значения age:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4. \tag{7.2}$$

Чему равна дисперсия этого предсказанного значения, т. е. $\mathrm{Var}\hat{f}(x_0)$? Метод наименьших квадратов возвращает оценки дисперсии каждого из рассчитанных коэффициентов $\hat{\beta}_j$, а также ковариацию для каждой пары оценок коэффициентов. Мы можем использовать эти величины для вычисления оцененной дисперсии $\hat{f}(x_0)$[1]. Оцененная стандартная ошибка $\hat{f}(x_0)$ в

[1] Если $\hat{\mathbf{C}}$ — это ковариационная матрица $\hat{\beta}_j$ размером 5×5, и если $\ell_0^T = (1, x_0, x_0^2, x_0^3, x_0^4)$, то $\mathrm{Var}[\hat{f}(x_0)] = \ell_0^T \hat{\mathbf{C}} \ell_0$.

заданной точке равна квадратному корню из этой дисперсии. Этот расчет повторяется для каждой опорной точки x_0, после чего мы изображаем подогнанную кривую и удвоенную стандартную ошибку по обе стороны от этой кривой. Использование удвоенной ошибки обусловлено тем, что для нормально распределенных остатков эта величина примерно соответствует 95%–му доверительному интервалу.

Похоже, что представленные на рис. 7.1 уровни заработной платы происходят из двух разных генеральных совокупностей: выглядит так, что есть группа *высокого достатка*, представители которой зарабатывают более 250 000$ в год, и группа *низкого достатка*. Мы можем рассматривать `wage` как бинарную переменную, разбив ее на эти две группы. Далее мы можем применить логистическую регрессию для предсказания этого бинарного отклика на основе полиномиальных функций `age`. Другими словами, мы подгоняем следующую модель:

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d^d)}. \tag{7.3}$$

Результат показан справа на рис. 7.1. Серые отметки вверху и внизу графика показывают значения возраста людей из групп высокого и низкого достатка. Сплошная синяя линия соответствует предсказанной вероятности принадлежности к группе высокого достатка в зависимости от `age`. Приведен также оцененный 95%–ный доверительный интервал. Как видим, доверительный интервал здесь довольно широкий, особенно в правой части графика. Хотя число наблюдений в этом наборе данных велико ($n = 3000$), имеются только 79 человек с высоким достатком, что приводит к высокой дисперсии оцененных коэффициентов и, как следствие, к широкому доверительному интервалу.

7.2 Ступенчатые функции

Использование полиномиальных функций переменных в качестве предикторов задает *глобальную* структуру нелинейной функции X. Во избежание подобной глобальной структуры можно воспользоваться *ступенчатыми функциями*. Мы разбиваем интервал значений X на *отрезки* и рассчитываем константы для каждого отрезка. Это эквивалентно преобразованию непрерывной переменной в *упорядоченную категориальную переменную*.

ступенчатая функция

упорядоченная категориальная переменная

Если быть точнее, мы выбираем точки сочленения $c_1, c_2, \ldots, c_K$ на интервале значений X, после чего создаем $K + 1$ новых переменных:

$$\begin{aligned}
C_0(X) &= I(X < c_1),\\
C_1(X) &= I(c_1 \le X < c_2),\\
C_2(X) &= I(c_2 \le X < c_3),\\
&\vdots\\
C_{K-1}(X) &= I(c_{K-1} \le X < c_K),\\
C_K(X) &= I(c_K \le X),
\end{aligned} \tag{7.4}$$

где $I(\cdot)$ представляет собой *индикаторную функцию*, которая возвращает 1 при выполнении условия и 0 в обратном случае. Например, $I(c_K \le X)$

индикаторная функция

равна 1, если $c_K \leq X$, и 0 в обратном случае. Иногда такие переменные называются также *фиктивными*. Заметьте, что для любого значения X $C_0(X)+C_1(X)+\cdots+C_K(X) = 1$, поскольку значения X должны быть разбиты в точности на $K+1$ отрезков. Далее мы подгоняем линейную модель по методу наименьших квадратов, используя $C_1(X), C_2(X), \ldots, C_K(X)$ в качестве предикторов[2]:

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_K C_K(x_i) + \epsilon. \tag{7.5}$$

Для заданного значения X ненулевой может быть только одна из функций $C_1(X), C_2(X), \ldots, C_K$. Заметьте, что при $X < c_1$ все предикторы в (7.5) равны нулю, в связи с чем β_0 можно интерпретировать как среднее значение Y для $X < c_1$. В то же время (7.5) предсказывает отклик $\beta_0 + \beta_j$ для $c_j \leq X < c_{j+1}$, и поэтому β_j представляет собой прирост отклика для X из интервала $c_j \leq X < c_{j+1}$, по сравнению с $X < c_1$.

Слева на рис. 7.2 приведен пример подгонки ступенчатых функций к данным `Wage`, изображенным на рис. 7.1. Мы также подгоняем логистическую регрессионную модель

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 C_1(x_i) + \cdots + \beta_K C_K(x_i))}{1 + \exp(\beta_0 + \beta_1 C_1(x_i) + \cdots + \beta_K C_K(x_i))} \tag{7.6}$$

для предсказания вероятности того, что некоторый человек определенного возраста принадлежит к группе высокого достатка. На рис. 7.2 справа показаны апостериорные вероятности, полученные с использованием этого подхода.

К сожалению, за исключением случаев, когда предикторы имеют естественные точки разрыва, кусочно–постоянные функции могут упустить важные свойства моделируемой зависимости. Например, слева на рис. 7.2 первый отрезок не отражает возрастания `wage` по мере увеличения `age`. Тем не менее подходы, основанные на ступенчатых функциях, очень популярны, в частности в биостатистике и эпидемиологии. Например, часто для определения возрастных групп используются 5–летние отрезки.

7.3 Базисные функции

базисная функция

Полиномиальные и кусочно–постоянные регрессионные модели в действительности являются частными случаями *базисных функций*. Идея заключается в том, чтобы иметь в распоряжении некоторое семейство функций, или преобразований, которые можно было бы применить к переменной X — $b_1(X), b_2(X), \ldots, b_K(X)$. Вместо подгонки линейной модели по X мы подгоняем модель вида

[2] Мы исключаем $C_0(X)$ из числа предикторов, перечисленных в (7.5), поскольку эта переменная дублирует свободный член и потому является избыточной. Это похоже на ситуацию, когда нам нужны только две индикаторные переменные для кодирования качественной переменной с тремя уровнями, при условии, что модель будет включать свободный член. Решение исключить $C_0(X)$ вместо какой–либо другой переменной $C_k(X)$ из (7.5) является произвольным. В качестве альтернативы мы могли бы включить $C_0(X), C_1(X), \ldots, C_K(X)$ и исключить свободный член.

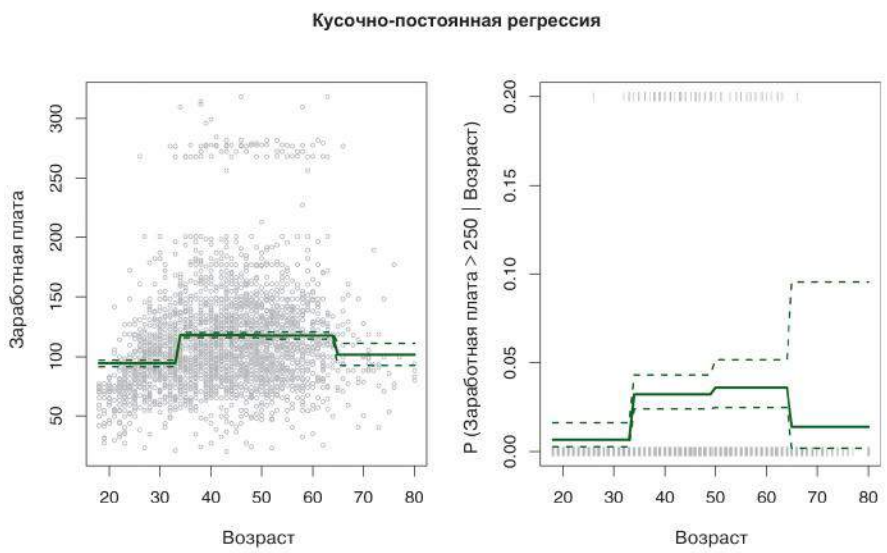

РИСУНОК 7.2. *Данные* `Wage`. Слева: *сплошная линия показывает значение* `wage` *(заработная плата, тыс. долларов), рассчитанное по методу наименьших квадратов на основе ступенчатых функций* `age` (возраст, лет). *Прерывистые кривые показывают оцененный 95%–ный доверительный интервал.* Справа: *мы моделируем бинарное событие* `wage > 250` *при помощи логистической регрессии на основе ступенчатых функций* `age`. *Показана предсказанная апостериорная вероятность того, что* `wage` *превышает 250 000$, а также соответствующий 95%–ный доверительный интервал*

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \cdots + \beta_K b_K(x_i) + \epsilon. \qquad (7.7)$$

Заметьте, что базисные функции $b_1(\cdot), b_2(\cdot), \ldots, b_K(\cdot)$ постоянны и известны. (Иными словами, мы выбираем эти функции заранее.) Для полиномиальной регрессии базисные функции представляют собой $b_j(x_i) = x_i^j$, а для кусочно–постоянных функций — $b_j(x_i) = I(c_j \leq x_i < c_{j+1})$. Мы думаем о (7.7) как о стандартной линейной модели с предикторами $b_1(x_i), b_2(x_i), \ldots, b_K(x_i)$. Как следствие мы можем использовать метод наименьших квадратов для оценивания неизвестных регрессионных коэффициентов из (7.7). Это важно, поскольку при таком сценарии будут применимы все методы, обсуждавшиеся в главе 3, такие как стандартные ошибки коэффициентов и F–статистика для оценивания статистической значимости модели в целом.

До этого момента в качестве базисных функций мы обсуждали полиномиальные и кусочно–постоянные функции, однако возможны и многие другие альтернативы. Например, для построения базисных функций мы можем применить вейвлет– или Фурье–преобразования. В следующем разделе мы рассмотрим один из очень часто используемых вариантов базисных функций — *регрессионные сплайны*.

регрессионный сплайн

7.4 Регрессионные сплайны

Теперь мы обсудим класс гибких базисных функций, которые расширяют только что рассмотренные методы полиномиальной и кусочно–постоянной регрессии.

7.4.1 Кусочно–полиномиальная регрессия

кусочно–полиномиальная регрессия

При выполнении кусочно–полиномиальной регрессии вместо подгонки полинома высокой степени для всего диапазона значений X подгоняются отдельные полиномы низкой степени к разным отрезкам значений X. Например, выполнение кусочной регрессии на основе кубического полинома состоит в подгонке следующей модели:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i, \tag{7.8}$$

где коэффициенты $\beta_0, \beta_1, \beta_2$ и β_3 в разных частях интервала значений X различаются. Точки, в которых происходит изменение коэффициентов, называют *узлами сочленения*[3].

узел сочленения

Например, кубический кусочный полином без узлов сочленения — это стандартный кубический полином, подобный представленному в (7.3) с $d = 3$. Кусочно–кубический полином с одним узлом в точке c принимает следующую форму:

$$\begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & \text{если } x_i < c \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & \text{если } x_i \geq c. \end{cases}$$

Другими словами, мы подгоняем две разные полиномиальные функции к данным: одну — для наблюдений с $x_i < c$, а другую — для наблюдений с $x_i \geq c$. Первая полиномиальная функция имеет коэффициенты $\beta_{01}, \beta_{11}, \beta_{21}, \beta_{31}$, а вторая — $\beta_{02}, \beta_{12}, \beta_{22}, \beta_{32}$. Каждую из этих функций можно подогнать при помощи метода наименьших квадратов, используя простые преобразования исходных предикторов.

Чем больше узлов сочленения, тем более гибкой будет кусочно–полиномиальная модель. Как правило, если мы применим K узлов к интервалу значений X, то получим $K+1$ различных кубических полиномов. Заметьте, что мы не обязаны применять кубический полином. Например, вместо этого мы можем подогнать кусочно–линейные функции. Более того, оказывается, что наши кусочно–постоянные функции из раздела 7.2 являются кусочными полиномами нулевой степени!

Вверху на рис. 7.3 показан кусочный кубический полином, подогнанный к данным `Wage` с использованием одного узла сочленения в точке `age = 50`. Сразу же обнаруживается проблема: эта функция не является непрерывной и выглядит очень странно! Поскольку каждый полином имеет четыре параметра, то при подгонке этой кусочно–полиномиальной модели мы в сумме задействуем восемь *степеней свободы*.

степени свободы

[3] В оригинале используется термин «knots». — *Прим. пер.*

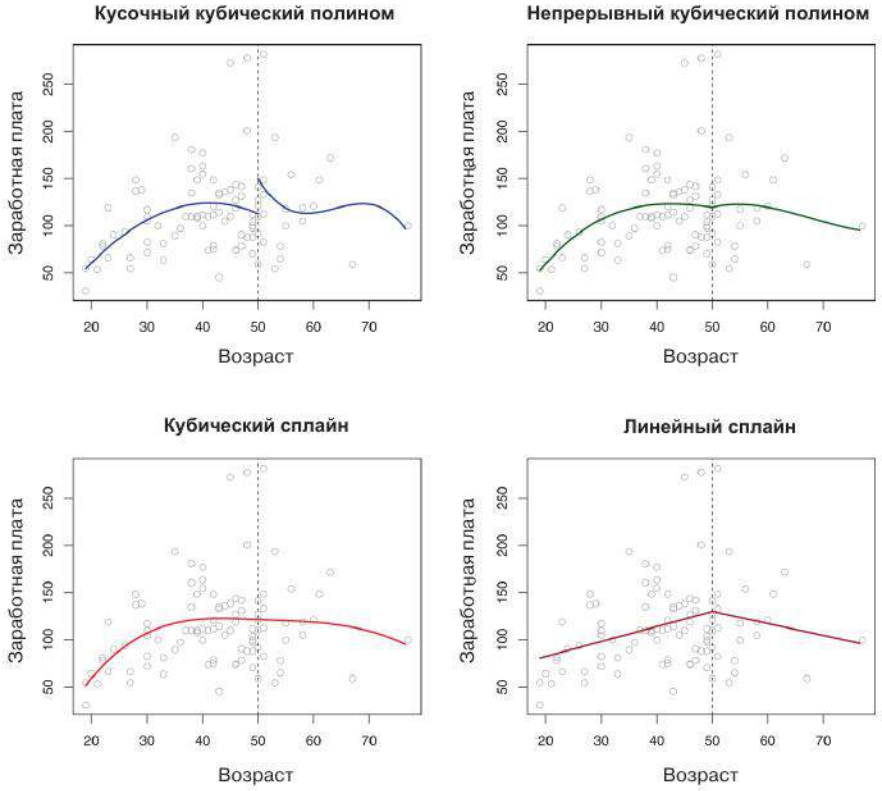

РИСУНОК 7.3. *Различные кусочно–полиномиальные модели, подогнанные к небольшой части данных* `Wage` *с узлом сочленения в точке* `age = 50`. Слева вверху: *кубические полиномы являются неограниченными.* Справа вверху: *кубические полиномы ограничены таким образом, чтобы обеспечить их непрерывность в точке* `age = 50`. Слева внизу: *наложены такие ограничения, которые делают непрерывными как сами кубические полиномы, так и их первую и вторую производные.* Справа внизу: *показан линейный сплайн, на который наложено ограничение на непрерывность*

7.4.2 Ограничения и сплайны

График, представленный слева вверху на рис. 7.3, выглядит странно из–за того, что подогнанная кривая является излишне гибкой. Для устранения этой проблемы мы можем подогнать кусочный полином, применив *ограничение*, которое позволяет сделать кривую непрерывной. Иными словами, в точке `age = 50` не должно быть скачка. Справа вверху на рис. 7.3 показан результат подгонки такой модели. Этот график выглядит лучше, чем представленный слева вверху, однако V–образное сочленение смотрится неестественно.

На графике, приведенном слева внизу, мы добавили еще два ограничения: теперь в точке `age = 50` непрерывными должны быть первая и вторая *производные* кусочных полиномов. Другими словами, мы требуем,

производная

чтобы кусочный полином в точке age = 50 был не только непрерывным, но еще и очень *гладким*. Каждое налагаемое нами ограничение в итоге приводит к высвобождению одной степени свободы, снижая тем самым сложность итоговой кусочно–полиномиальной модели. Так, на графике слева вверху мы тратим восемь степеней свободы, тогда как на графике, показанном слева внизу, мы наложили три ограничения (непрерывность, непрерывность первой производной и непрерывность второй производной) и поэтому потратили пять степеней свободы. Кривая на графике, приведенном слева внизу, называется *кубическим сплайном*[4]. Кубический сплайн с K узлами использует в сумме $4 + K$ степеней свободы.

кубический сплайн

Справа внизу на рис. 7.3 показан *линейный сплайн*, который является непрерывным в точке age = 50. Сплайну степени d можно дать следующее определение: это кусочный полином степени d с непрерывными производными вплоть до степени $d - 1$ в каждом узле сочленения. Следовательно, линейный сплайн получают путем подгонки линий ко всем заданным узлами сочленения интервалам значений предиктора, накладывая при этом ограничение на непрерывность в каждом узле.

линейный сплайн

На рис. 7.3 имеется только один узел сочленения в точке age = 50. Конечно, мы могли бы добавить большее количество узлов и наложить ограничения на непрерывность для каждого из них.

7.4.3 Представление сплайнов с помощью базисных функций

Регрессионные сплайны, рассмотренные только что в предыдущем разделе, могут показаться довольно сложными: как нам подогнать кусочный полином степени d при условии, чтобы он (и, возможно, его первые $d - 1$ производные) был непрерывным? Оказывается, что для представления регрессионного сплайна мы можем воспользоваться базисной моделью (7.7). Для некоторого подходящего набора базисных функций $b_1, b_2, \ldots, b_{K+3}$ кубический сплайн с K узлами сочленения можно представить следующим образом:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i. \qquad (7.9)$$

Далее модель (7.9) можно подогнать при помощи метода наименьших квадратов.

Подобно тому, как мы видели несколько способов представления полиномов, существует также множество эквивалентных способов представления кубических сплайнов в (7.9) с использованием различных базисных функций. Наиболее очевидный способ представления кубического сплайна в виде (7.9) заключается в том, чтобы начать с базисных функций для кубического полинома, т. е. x, x^2, x^3, а затем добавить *усеченную степенную базисную функцию*[5] для каждого узла сочленения. Усеченную степенную базисную функцию определяют следующим образом:

усеченная степенная базисная функция

[4] Кубические сплайны популярны, поскольку глаза большинства людей не могут определить отсутствие непрерывности в узлах сочленения.

[5] В оригинале используется термин «truncated power basis function». — *Прим. пер.*

$$h(x,\xi) = (x-\xi)_+^3 = \begin{cases} (x-\xi)^3 & \text{если } x > \xi \\ 0 & \text{в остальных случаях,} \end{cases} \tag{7.10}$$

где ξ — это узел сочленения. Можно показать, что добавление члена $\beta_4 h(x,\xi)$ в модель (7.8) для кубического сплайна приведет к нарушению непрерывности только третьей производной в точке ξ; сама функция, а также первая и вторая производные в каждом узле останутся непрерывными.

Иными словами, для подгонки к некоторым данным кубического сплайна с K узлами сочленения мы строим регрессионную модель по методу наименьших квадратов со свободным членом и $3+K$ предикторами вида $X, X^2, X^3, h(X,\xi_1), h(X,\xi_2), \ldots, h(X,\xi_K)$, где $\xi_1, \ldots, \xi_K$ — это узлы сочленения. Всего оцениваются $K+4$ регрессионных коэффициента, в связи с чем при подгонке кубического сплайна с K узлами затрачивается $K+4$ степеней свободы.

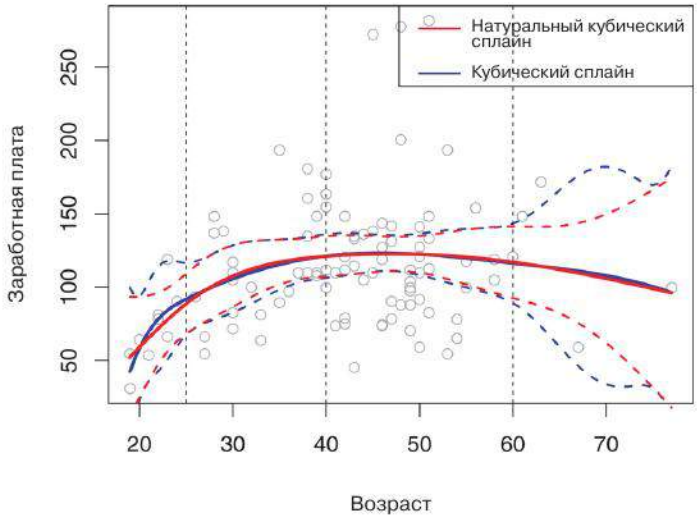

РИСУНОК 7.4. *Кубический сплайн и натуральный кубический сплайн с тремя узлами сочленения, подогнанные к небольшой части данных* `Wage`

К сожалению, сплайны могут проявлять высокую дисперсию на внешних границах интервала значений предиктора, т. е. когда X принимает некоторое очень низкое или очень высокое значение. На рис. 7.4 показана модель с тремя узлами сочленения, подогнанная к данным `Wage`. Мы видим, что кривые доверительного интервала по краям выглядят довольно странно. Натуральный сплайн — это регрессионный сплайн с дополнительными *краевыми ограничениями*[6]: от функции требуется, чтобы она была линейной вблизи к краю (т. е. в области, где X меньше, чем наименьший

натуральный сплайн

[6] В оригинале используется термин «boundary constraints». В русскоязычных источниках применяется также термин «краевые условия». — *Прим. пер.*

узел сочленения, или больше, чем наибольший узел). Это дополнительное ограничение означает, что натуральные сплайны обычно дают более стабильные оценки по краям. На рис. 7.4 в виде красной линии показан также натуральный кубический сплайн. Заметьте, что соответствующие доверительные интервалы в этом случае уже.

7.4.4 Выбор числа и расположения узлов сочленения

Как нам расположить узлы сочленения при подгонке сплайна? Регрессионный сплайн наиболее гибок в областях, содержащих большое число узлов, поскольку в этих областях полиномиальные коэффициенты могут быстро изменяться. Следовательно, одна из возможностей заключается в размещении большего числа узлов в местах, где, как нам кажется, функция может изменяться особенно быстро, и меньшего числа узлов в местах, где она выглядит более стабильной. Хотя этот подход может сработать хорошо, на практике принято располагать узлы равномерно. Один из способов состоит в том, чтобы выбрать желаемое число степеней свободы, а затем воспользоваться статистической программой для автоматического размещения соответствующего числа узлов в точках, равномерно разбивающих данные на отрезки.

На рис. 7.5 приведен пример для набора данных `Wage`. Как и на рис. 7.4, мы подогнали натуральный кубический сплайн с тремя узлами сочленения, однако на этот раз в качестве мест расположения узлов автоматически были выбраны 25–й, 50–й и 75–й процентили переменной `age`. Это было сделано в результате запроса четырех степеней свободы. Аргументация того, почему четыре степени свободы приводят к трем узлам сочленения, носит несколько технический характер[7].

Сколько узлов сочленения нам следует выбрать или, другими словами, сколько степеней свободы должен содержать сплайн? Один из подходов состоит в переборе разного числа узлов, чтобы выяснить, какое из них порождает оптимальную кривую. Несколько более объективный подход заключается в использовании перекрестной проверки, как обсуждалось в главах 5 и 6. При реализации этого метода мы удаляем некоторую часть данных (например, 10%), подгоняем сплайн с определенным числом узлов к остальным данным, а затем используем этот сплайн для получения предсказаний на удержанных данных. Эта процедура повторяется несколько раз до тех пор, пока каждое наблюдение не было удалено из обучающей выборки один раз, а затем вычисляем общую RSS перекрестной проверки. Данную процедуру можно повторить для разных значений числа узлов K. Далее выбирают такое значение K, которое обеспечивает наименьшую RSS.

На рис. 7.6 показаны среднеквадратичные ошибки, полученные в ходе десятикратной перекрестной проверки для сплайнов с разным числом степеней свободы (на примере набора данных `Wage`). График слева соответствует натуральному сплайну, а график справа — кубическому. Эти два

[7] На самом деле имеется пять узлов сочленения, включая два краевых. Кубический сплайн с пятью узлами имел бы девять степеней свободы. Однако натуральный кубический сплайн имеет два дополнительных *натуральных* ограничения по краям для достижения линейности, что приводит к $9 - 4 = 5$ степеням свободы. Поскольку это включает константу, которая входит в свободный член модели, мы засчитываем ее как четвертую степень свободы.

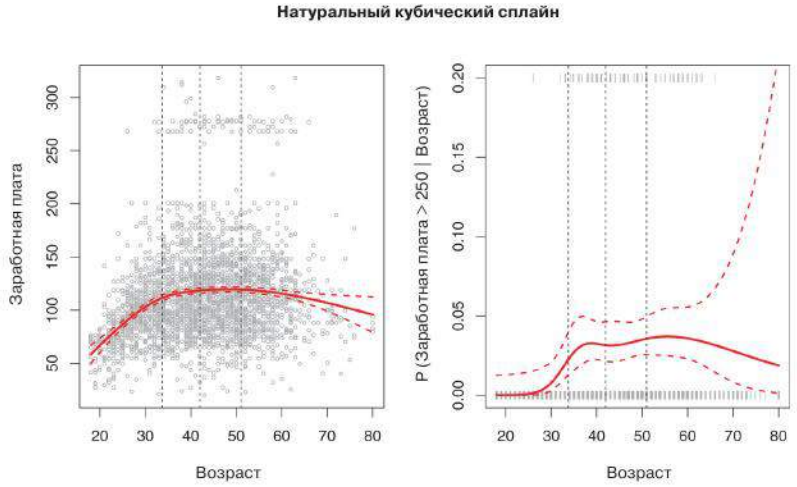

РИСУНОК 7.5. *Функция натурального кубического сплайна с четырьмя степенями свободы подогнана к данным* `Wage`. Слева: *сплайн подогнан к значениям заработной платы (тыс. долларов) в зависимости от возраста.* Справа: *использована логистическая регрессия для моделирования бинарного события* `wage > 250` *в зависимости от возраста. Показана оцененная апостериорная вероятность того, что заработная плата превышает 250 000$*

метода дают почти идентичные результаты, и имеется четкое указание на то, что модель с одной степенью свободы (линейная модель) не является адекватной для этих данных. Обе кривые быстро становятся плоскими, и похоже, что три степени свободы для натурального сплайна и четыре степени свободы для кубического сплайна являются вполне подходящими.

В разделе 7.7 мы рассмотрим создание аддитивных сплайн–моделей на основе нескольких предикторов одновременно. Для этого может потребоваться нахождение числа степеней свободы для каждого предиктора в отдельности. Однако в подобных случаях мы применяем более практичный подход и априори задаем некоторое фиксированное число степеней свободы для всех предикторов (например, четыре).

7.4.5 Сравнение с полиномиальной регрессией

Регрессионные сплайны часто дают более высокое качество предсказаний, чем полиномиальная регрессия. Это обусловлено тем, что, в отличие от полиномов, которые для создания гибких моделей должны использовать некоторое высокое число степеней свободы (т. е. высокий показатель степени, например вплоть до X^{15}), сплайны обеспечивают гибкость за счет увеличения числа узлов сочленения при одновременном удержании числа степеней свободы на фиксированном уровне. Как правило, такой подход дает более стабильные оценки. Сплайны также позволяют нам разместить большее количество узлов, достигая тем самым более высокой гибкости в областях, где функция f изменяется быстро, и меньшее количество уз-

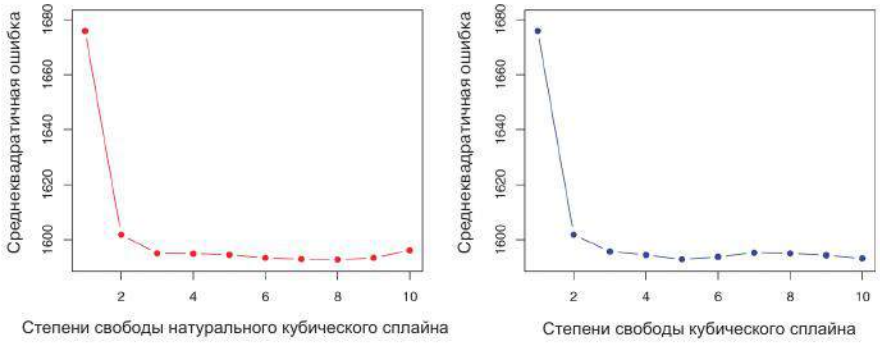

РИСУНОК 7.6. *Среднеквадратичные ошибки, полученные в результате перекрестной проверки и предназначенные для нахождения оптимального числа степеней свободы для подгонки сплайнов к данным* `Wage`. *Отклик представлен переменной* `wage`, *а предиктор — переменной* `age`. Слева: *натуральный кубический сплайн.* Справа: *кубический сплайн*

лов в областях, где функция f выглядит более стабильной. На рис. 7.7 сравниваются натуральный кубический сплайн с 15 степенями свободы и полином 15–й степени, подогнанные к данным `Wage`. Излишняя гибкость полинома порождает нежелательные результаты по краям, тогда как натуральный кубический сплайн обеспечивает вполне подходящее описание этих данных.

7.5 Сглаживающие сплайны

7.5.1 Общее представление о сглаживающих сплайнах

В предыдущем разделе мы обсудили регрессионные сплайны, которые мы создаем путем выбора нескольких узлов сочленения, расчета нескольких базисных функций и последующего применения метода наименьших квадратов для оценивания коэффициентов модели. Теперь мы опишем несколько отличающийся подход для создания сплайнов.

При подгонке гладкой кривой к тому или иному набору данных мы стремимся найти некоторую функцию $g(x)$, которая хорошо описывает эти данные, т. е. мы хотим получить низкую величину $\text{RSS} = \sum_{i=1}^{n}(y_i - g(x))^2$. Однако у этого подхода есть одна проблема. Не накладывая никаких ограничений на $g(x)$, мы всегда можем сделать RSS равной нулю, просто выбрав функцию g, которая *интерполирует* все значения y_i. Подобная функция привела бы к очень сильному эффекту переобучения, т. е. она оказалась бы чрезмерно гибкой. На самом же деле мы хотим, чтобы функция g давала низкую RSS, но при этом была гладкой.

Как же нам сделать g гладкой? Для этого существует несколько способов. Естественным подходом является нахождение такой функции g, которая минимизирует

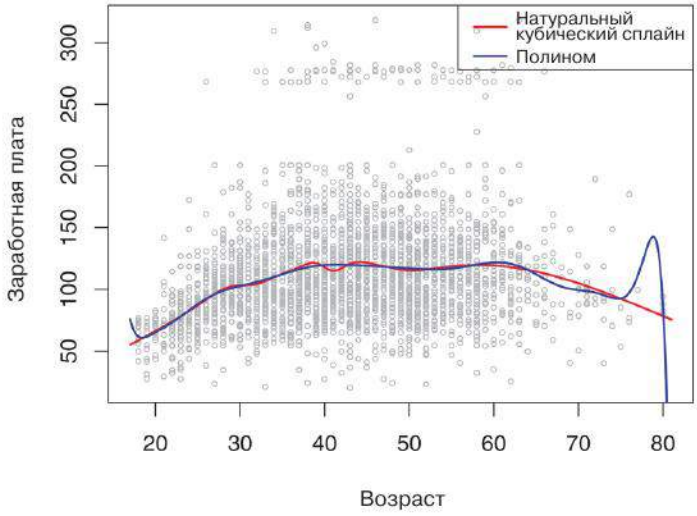

РИСУНОК 7.7. *Сравниваются кубический сплайн с 15 степенями свободы и полином 15-й степени, подогнанные к данным* `Wage`. *Полиномы могут проявлять весьма необоснованное поведение, особенно по краям*

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt, \tag{7.11}$$

где λ — это некоторый неотрицательный *гиперпараметр*. Функция g, минимизирующая (7.11), известна как *сглаживающий сплайн*.

сглаживающий сплайн

Что означает уравнение (7.11)? Это уравнение принимает так называемую форму «потерь и штрафа»[8], с которой мы сталкивались в главе 6 в контексте гребневой регрессии и лассо. Член $\sum_{i=1}^{n}(y_i - g(x_i))^2$ представляет собой *функцию потерь*, которая заставляет g близко соответствовать данным, а член $\lambda \int g''(t)^2 dt$ — это *штрафное слагаемое*, которое ограничивает вариабельность g. Нотация $g''(t)$ обозначает вторую производную функции g. Первая производная $g'(t)$ отражает угол наклона функции в точке t, а вторая производная соответствует скорости, с которой этот угол изменяется. Следовательно, в общих чертах вторая производная некоторой функции — это мера ее *извилистости*: она принимает высокие значения, когда $g(t)$ является очень извилистой, и близка к нулю в остальных случаях. (Вторая производная прямой линии равна нулю; заметьте, что в этом случае линия является идеально гладкой.) Символ $\int$ обозначает *интеграл*, о котором мы можем думать как об операции суммирования в пределах интервала значений t. Иными словами $\int g''(t)^2 dt$ — это просто обобщенная мера изменчивости функции $g'(t)$ на всем интервале ее значений. Если g является очень гладкой, то $g'(t)$ будет почти неизменной и $\int g''(t)^2$ примет некоторое низкое значение. Однако если g очень извилиста и изменчива, то $g'(t)$ тоже будет значительно варьировать и

функция потерь

[8] В оригинале используется термин «loss + penalty formulation». — *Прим. пер.*

$\int g''(t)^2$ примет некоторое высокое значение. Следовательно, в (7.11) слагаемое $\lambda \int g''(t)^2$ заставляет функцию g становиться гладкой. Чем выше значение λ, чем более гладкой будет g.

При $\lambda = 0$ штрафное слагаемое в (7.11) не имеет никакого эффекта, и функция g окажется очень извилистой и будет в точности интерполировать обучающие наблюдения. При $\lambda \to \infty$ g будет идеально гладкой — это будет прямая линия, проходящая максимально близко через все обучающие наблюдения. Более того, в этом случае g будет линией наименьших квадратов, поскольку функция потерь в (7.11) сводится к минимизации суммы квадратов остатков. При некотором промежуточном значении λ g будет аппроксимировать обучающие наблюдения, являясь при этом в определенной степени гладкой функцией. Как видим, λ контролирует соотношение между смещением и дисперсией сглаживающего сплайна.

Можно показать, что функция $g(x)$, минимизирующая (7.11), обладает несколькими особенными свойствами: это кусочный кубический полином с узлами сочленения, приходящимися на каждое значение $x_1, \ldots, x_n$, и с гладкими первой и второй производными в каждом узле. Более того, в областях, находящихся за пределами минимального и максимального узлов, эта функция имеет линейную форму. Другими словами, *функция $g(x)$, которая минимизирует (7.11), является натуральным кубическим сплайном с узлами сочленения в точках $x_1, \ldots, x_n$*! Однако это не тот же сплайн, который можно было бы получить в результате применения описанного в подразделе 7.4.3 подхода, основанного на базисных функциях с узлами в точках $x_1, \ldots, x_n$, — скорее, это *сжатая* версия подобного натурального сплайна, в которой степень сжатия контролируется значением гиперпараметра λ из (7.11).

7.5.2 Нахождение параметра сглаживания λ

Как мы выяснили, сглаживающий сплайн — это не что иное, как натуральный кубический сплайн с узлами сочленения, приходящимися на каждое уникальное значение x_i. Может показаться, что сглаживающий сплайн будет иметь слишком много степеней свободы, поскольку узлы сочленения, приходящиеся на каждое наблюдение, приводят к большому уровню гибкости. Однако гиперпараметр λ контролирует извилистость сглаживающего сплайна, а следовательно, и *эффективное число степеней свободы*[9]. Можно показать, что при увеличении λ от 0 до ∞ эффективное число степеней свободы, которое мы обозначаем как df_λ, возрастает с n до 2.

эффективное число степеней свободы

Почему в контексте сглаживающих сплайнов мы обсуждаем *эффективное* число степеней свободы, а не просто число степеней свободы? Обычно под числом степеней свободы понимают число свободных параметров — таких, например, как коэффициенты, оцененные для полиномиального или кубического сплайна. Хотя сглаживающий сплайн имеет n параметров и, следовательно, n номинальных степеней свободы, эти n параметров существенно ограничены, или сжаты. Следовательно, df_λ является мерой гибкости сглаживающего сплайна — чем она выше, тем гибче сглаживающий сплайн (и ниже смещение, но выше дисперсия). Определение эффективных степеней свободы носит несколько технический характер. Мы можем записать

[9] В оригинале используется термин «effective degrees of freedom». — *Прим. пер.*

$$\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \mathbf{y}, \tag{7.12}$$

где $\hat{\mathbf{g}}$ — это решение уравнения (7.11) для некоторого конкретного значения λ, т. е. это вектор с n значениями, предсказанными сглаживающим сплайном для обучающих наблюдений $x_1, \ldots, x_n$. Уравнение (7.12) показывает, что при использовании сглаживающего сплайна вектор с предсказанными значениями можно записать в виде матрицы $\mathbf{S}_\lambda$ размером $n \times n$, умноженной на вектор значений отклика $\mathbf{y}$. Тогда эффективное число степеней свободы можно определить как сумму диагональных элементов матрицы $\mathbf{S}_\lambda$:

$$df_\lambda = \sum_{i=1}^{n} \{\mathbf{S}_\lambda\}_{ii}. \tag{7.13}$$

При подгонке сглаживающего сплайна нам нет необходимости выбирать расположение узлов сочленения — отдельный узел будет приходиться на каждое обучающее наблюдение $x_1, \ldots, x_n$. Вместо этого у нас возникает другая проблема: нам нужно выбрать значение λ. Неудивительно, что одним из возможных решений этой проблемы является перекрестная проверка. Иными словами, мы можем найти значение λ, которое минимизирует RSS по результатам перекрестной проверки. Оказывается, что ошибку перекрестной проверки по отдельным наблюдениям (LOOCV) для сглаживающих сплайнов можно очень эффективно вычислить при помощи следующей формулы (по сути, с затратой тех же вычислительных ресурсов, которые требуются для подгонки одной модели):

$$\mathrm{RSS}_{cv}(\lambda) = \sum_{i=1}^{n} (y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}} \right]^2.$$

$\hat{g}_\lambda^{(-i)}(x_i)$ — это значение данного сглаживающего сплайна в точке x_i, полученное в результате подгонки модели по всем обучающим наблюдениям, за исключением i–го наблюдения (x_i, y_i). В то же время $\hat{g}_\lambda(x_i)$ — это функция сглаживающего сплайна, подогнанная ко всем обучающим наблюдениям и оцененная в точке x_i. Эта удивительная формула говорит, что мы можем вычислить каждую из этих моделей с *одним исключенным наблюдением*, используя только $\hat{g}_\lambda$ — исходную модель, построенную *по всем* наблюдениям[10]! У нас есть очень похожая формула (5.2) на стр. 197 в главе 5 для регрессии по методу наименьших квадратов. Используя (5.2), мы можем быстро вычислить LOOCV для регрессионных сплайнов, обсуждавшихся ранее в этой главе, а также для регрессии по методу наименьших квадратов с использованием произвольно выбранных базисных функций.

На рис. 7.8 показаны результаты подгонки сглаживающего сплайна к данным `Wage`. Красная кривая соответствует сглаживающему сплайну с 16 априорно заданными эффективными степенями свободы. Синяя кривая представляет собой сглаживающий сплайн, полученный в результате нахождения λ на основе LOOCV; в этом случае выбранное значение λ дает 6.8 эффективной степени свободы (вычислено по (7.13)). Для этих

[10] Точные формулы для вычисления $\hat{g}_\lambda$ и $\mathbf{S}_\lambda$ носят очень технический характер, однако для выполнения этих вычислений имеются эффективные алгоритмы.

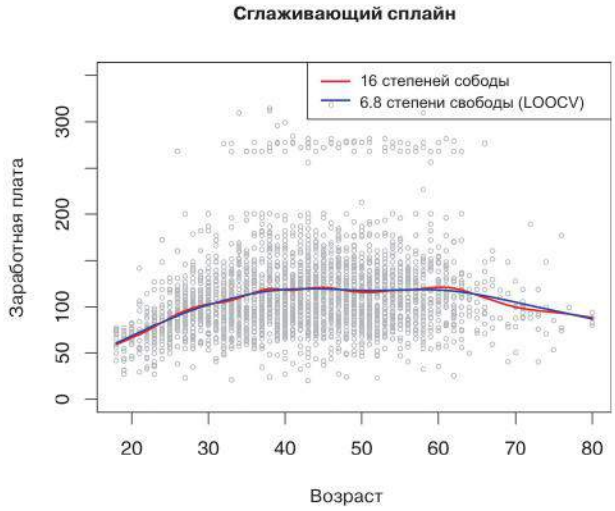

РИСУНОК 7.8. *Сглаживающие сплайны, подогнанные к данным* `Wage`. *Красная кривая получена с использованием 16 априорно заданных эффективных степеней свободы. Для синей кривой параметр* λ *был найден автоматически путем перекрестной проверки по отдельным наблюдениям, которая привела к 6.8 эффективной степени свободы*

данных разница между двумя полученными сглаживающими сплайнами практически неразличима, хотя сплайн с 16 степенями свободы выглядит несколько более извилистым. Поскольку эти две модели почти не различаются, сглаживающий сплайн с 6.8 эффективной степени свободы предпочтителен, т. к. более простые модели обычно обеспечивают более высокое качество (если только данные не свидетельствуют в пользу более сложной модели).

7.6 Локальная регрессия

локальная регрессия

Локальная регрессия — это другой подход для подгонки гибких нелинейных функций, который заключается в вычислении модельного значения в некоторой целевой точке x_0 на основе только близлежащих обучающих наблюдений. Эта идея проиллюстрирована на примере имитированных данных на рис. 7.9, где одна целевая точка находится вблизи значения 0.4, а другая приходится на краевое значение 0.05. Синяя линия на этом рисунке соответствует функции $f(x)$, на основе которой были сгенерированы эти данные, а светло–оранжевая линия показывает локальную оценку функции $\hat{f}(x)$. Реализация метода локальной регрессии описана в алгоритме 7.1.

Заметьте, что на 3–м шаге алгоритма 7.1 весовые коэффициенты K_{i0} будут разными для разных значений x_0. Иными словами, для подгонки локальной регрессии в новой точке нам необходимо рассчитать новую регрессионную кривую по методу взвешенных наименьших квадратов путем

Алгоритм 7.1 Локальная регрессия в точке $X = x_0$

1. Выберите долю $s = k/n$ обучающих наблюдений, чьи значения x_i находятся ближе всего к x_0.

2. Присвойте весовой коэффициент $K_{i0} = K(x_i, x_0)$ каждой точке из этой окрестности таким образом, чтобы вес наиболее удаленной от x_0 точки был равен нулю, а вес наиболее близко расположенной точки был максимальным. Веса всех остальных точек, за исключением этих k ближайших соседей, получают нулевые значения.

3. Используя метод взвешенных наименьших квадратов, постройте регрессионную зависимость y_i от x_i на основе упомянутых выше весов путем нахождения $\hat{\beta}_0$ и $\hat{\beta}_1$, которые минимизируют

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. Прогнозное значение в точке x_0 находят как $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

минимизации (7.14) для нового набора весов. Иногда локальную регрессию называют *процедурой, основанной на запоминании*[11], поскольку, подобно методу ближайших соседей, каждый раз при расчете прогнозных значений нам нужны все обучающие данные. Мы не будем здесь углубляться в технические детали локальной регрессии — по этой теме существуют целые книги.

В ходе построения локальной регрессионной модели необходимо принять ряд решений (например, в отношении типа весовой функции K), а также определиться с тем, следует ли подгонять линейную, кусочно–постоянную или квадратичную регрессию на третьем шаге алгоритма 7.1. (Уравнение (7.14) соответствует линейной регрессии.) Хотя все эти решения играют определенную роль, наиболее важным является выбор *ширины окна* s[12], определение которой дано в описании первого шага алгоритма 7.1. Ширина окна локальной регрессии аналогична гиперпараметру λ в сглаживающих сплайнах: она контролирует гибкость нелинейной модели. Чем меньше значение s, тем более *локальной* и извилистой будет наша модель; в то же время очень большое значение s приведет к глобальной модели, построенной по всей обучающей выборке. Как и раньше, для нахождения s мы можем применить перекрестную проверку или присвоить то или иное значение априори. На рис. 7.10 показаны две локальные регрессионные модели для данных `Wage`, полученные на основе двух значений s — 0.7 и 0.2. Как и следовало ожидать, модель с $s = 0.7$ является более гладкой, чем модель с $s = 0.2$.

Идею локальной регрессии можно обобщить на многие случаи. Одно из полезных обобщений в ситуации с несколькими переменными

[11] В оригинале используется термин «memory–based procedure». — *Прим. пер.*
[12] В оригинале используется термин «span». — *Прим. пер.*

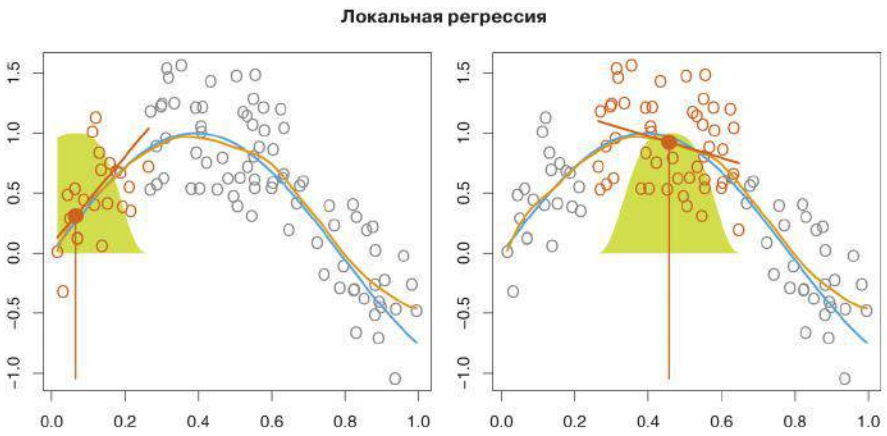

РИСУНОК 7.9. *Локальная регрессия на примере имитированных данных: голубая линия соответствует функции $f(x)$, на основе которой были сгенерированы эти данные, а светло–оранжевая линия — оценке $\hat{f}(x)$, полученной на основе локальной регрессии. Оранжевые полые точки находятся в окрестности целевой точки x_0, показанной в виде оранжевой вертикальной линии. Приведенная на графике желтая колоколообразная фигура изображает присвоенные каждому обучающему наблюдению веса, которые снижаются до нуля по мере удаления от целевой точки. Оценка $\hat{f}(x)$ в точке x_0 получена путем подгонки взвешенной линейной регрессионной модели (оранжевый отрезок) и расчета предсказанного на основе этой модели значения для x_0 (оранжевая сплошная точка)*

$X_1, X_2, \ldots, X_p$ состоит в подгонке множественной линейной регрессионной модели, которая является глобальной по одним переменным, и локальной по какой–то другой переменной (такой, например, как время). Подобные *модели с варьирующими коэффициентами*[13] полезны, когда необходимо адаптировать ту или иную модель к недавно полученным данным. Локальная регрессия естественным образом обобщается также на случаи, когда мы хотим подогнать модели, локальные не по одной, а по двум переменным — X_1 и X_2. Для подгонки линейных регрессионных моделей с двумя предикторами мы можем просто использовать окрестности каждой целевой точки в двумерном пространстве. Теоретически этот подход можно реализовать и для больших размерностей, подгоняя линейные регрессионные модели к точкам из p–мерных окрестностей. Однако при $p > 3$ или $p > 4$ качество локальной регрессии может оказаться низким, поскольку лишь небольшое число обучающих наблюдений будут располагаться вблизи x_0. Описанная в главе 3 регрессия по методу ближайших соседей страдает от похожей проблемы при больших размерностях.

модель с варьирующими коэффициентами

[13] В оригинале используется термин «varying coefficients models». — *Прим. пер.*

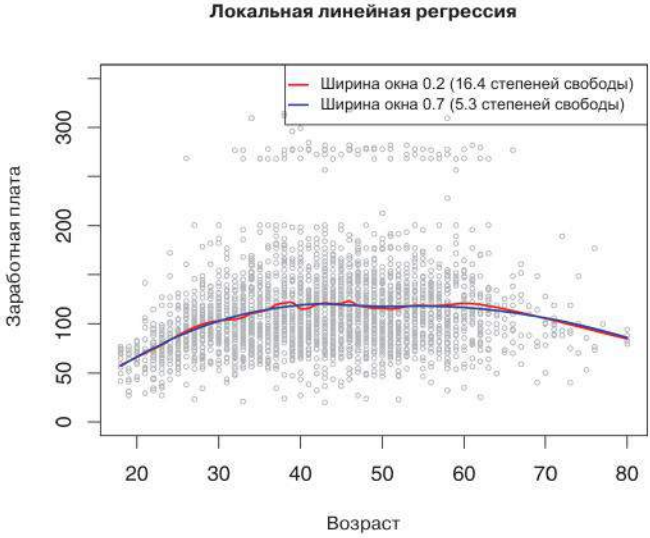

РИСУНОК 7.10. *Локальные регрессионные модели, подогнанные к данным* `Wage`. *Ширина окна определяет долю наблюдений, используемых для расчета модельного значения в каждой целевой точке*

7.7 Обобщенные аддитивные модели

В разделах 7.1–7.6 мы представили несколько гибких методов, позволяющих предсказывать некоторый отклик Y по единственному предиктору X. Эти методы можно рассматривать в качестве расширения простой линейной регрессии. Здесь мы рассмотрим задачу предсказания отклика Y, нелинейно связанного с несколькими предикторами $X_1, \ldots, X_p$. Соответствующий этой задаче метод является расширением множественной линейной регрессии.

обобщенные аддитивные модели

Обобщенные аддитивные модели (GAM)[14] расширяют стандартную линейную модель, допуская использование нелинейных функций каждого предиктора при одновременном сохранении *аддитивности*. Подобно линейным моделям, GAM можно применять как для количественных, так и для качественных откликов. Сначала мы рассмотрим GAM для количественных откликов в подразделе 7.7.1, а затем — GAM для качественных в подразделе 7.7.2.

аддитивность

7.7.1 GAM для регрессионных задач

Естественным способом расширения множественной линейной регрессионной модели

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

для выражения нелинейных зависимостей между каждой переменной и

[14] Аббревиатура от «generalized additive models». — *Прим. пер.*

откликом является замена каждого линейного компонента $\beta_j x_{ij}$ на (гладкую) нелинейную функцию $f_j(x_{ij})$. Тогда мы могли бы записать модель следующим образом:

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\ &= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i. \end{aligned} \tag{7.15}$$

Это — пример GAM. Эта модель называется *аддитивной*, поскольку мы вычисляем отдельные функции f_j для каждой переменной X_j, а затем суммируем их.

В разделах 7.1–7.6 мы обсудили целый ряд методов, позволяющих подгонять модели на основе единственного предиктора. Красота GAM заключается в том, что мы можем использовать эти методы в качестве строительных блоков для создания аддитивных моделей. Более того, для большинства методов, которые мы видели в этой главе до сих пор, это можно сделать довольно просто. Возьмем, например, натуральные сплайны и рассмотрим задачу построения модели

$$\texttt{wage} = \beta_0 + f_1(\texttt{year}) + f_2(\texttt{age}) + f_3(\texttt{education}) + \epsilon \tag{7.16}$$

для данных `Wage`. Здесь `year` и `age` являются количественными переменными, а `education` — качественной переменной с пятью уровнями — `<HS`, `HS`, `<Coll`, `Coll`, `>Coll`, — которые описывают уровень образования, полученного тем или иным человеком. Первые две функции мы подгоняем при помощи натуральных сплайнов. Третью функцию мы подгоняем в виде константы для каждого уровня, используя обычный метод индикаторных переменных из подраздела 3.3.1.

На рис. 7.11 показан результат подгонки модели (7.16) по методу наименьших квадратов. Построить такую модель несложно, поскольку, как было отмечено в разделе 7.4, натуральные сплайны можно сконструировать при помощи нескольких хорошо подобранных базисных функций. Следовательно, вся модель — это просто одна большая регрессия на основе базисных функций и индикаторных переменных, «упакованных» в одну большую регрессионную матрицу.

Интерпретация рис. 7.11 проста. Из графика, приведенного слева, следует, что при некоторых фиксированных уровнях возраста и образования заработная плата со временем обычно возрастает; это может быть связано с инфляцией. График, приведенный в центре, показывает, что для некоторых фиксированных уровней образования и года заработная плата обычно принимает максимальные значения при промежуточных значениях возраста, а минимальные — у очень молодых и очень пожилых людей. Из графика, показанного справа, следует, что для некоторых фиксированных значений года и возраста заработная плата обычно возрастает вслед за уровнем образования, т. е. чем выше уровень образования человека, тем выше (в среднем) его доход. Все эти выводы очевидны.

На рис. 7.12 показаны три похожих графика, однако на этот раз f_1 и f_2 представляют собой сглаживающие сплайны с четырьмя и пятью степенями свободы соответственно. Подгонка GAM со сглаживающими сплайнами сложнее подгонки GAM с натуральными сплайнами, поскольку в случае со сглаживающими сплайнами метод наименьших квадратов

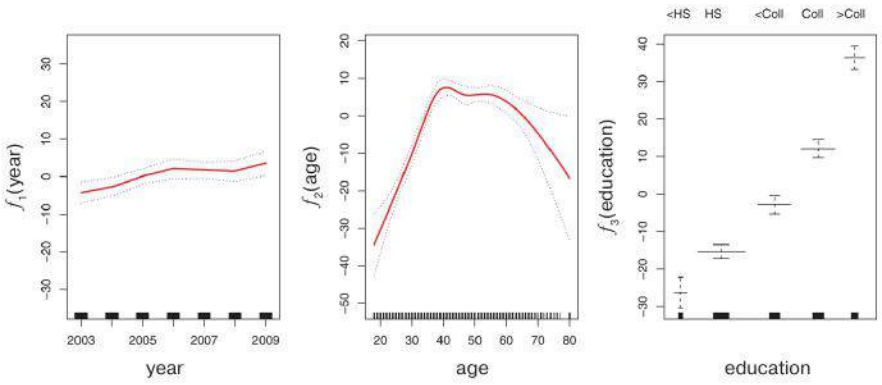

РИСУНОК 7.11. *Графики зависимостей между предикторами из набора данных* `Wage` *и откликом* `wage`*, согласно модели (7.16). На каждом графике показаны подогнанная функция и ее стандартные ошибки в каждой точке. Первые две функции представляют собой натуральные сплайны для* `year` *и* `age` *с четырьмя и пятью степенями свободы соответственно. Третья функция — это ступенчатая функция, подогнанная к качественной переменной* `education`

неприменим. Однако можно использовать стандартное программное обеспечение вроде функции `gam` в `R` для построения GAM–моделей со сглаживающими сплайнами при помощи т. н. «*метода настройки с возвращениями*»[15]. Этот метод позволяет построить модель с несколькими переменными путем многократных обновлений модели для каждого отдельного предиктора, в ходе которых эффекты всех остальных предикторов фиксируются. Элегантность этого метода заключается в том, что во время каждого обновления той или иной функции мы просто подгоняем модель для соответствующей переменной по *частным остаткам*[16].

настройка с возвращениями

Подогнанные функции на рис. 7.11 и 7.12 довольно похожи. В большинстве ситуаций различия между GAM на основе сглаживающих сплайнов и GAM на основе натуральных сплайнов незначительны.

Мы не обязаны использовать сплайны в качестве строительных блоков для GAM: с тем же успехом для создания GAM мы можем применить локальную регрессию, полиномиальную регрессию или любую комбинацию подходов, рассмотренных ранее в этой главе. В лабораторной работе в конце этой главы GAM будут рассмотрены более детально.

Преимущества и недостатки GAM

Прежде чем продолжить, давайте обобщим преимущества и недостатки GAM.

▲ GAM позволяют нам подогнать нелинейную функцию f_j к каждо-

[15] В оригинале используется термин «backfitting». — *Прим. пер.*

[16] Например, частный остаток для X_3 выражается как $r_i = y_i - f_1(x_{i1}) - f_2(x_{i2})$. Если мы знаем f_1 и f_2, то можем подогнать f_3, рассматривая этот остаток как отклик в нелинейной регрессии по X_3.

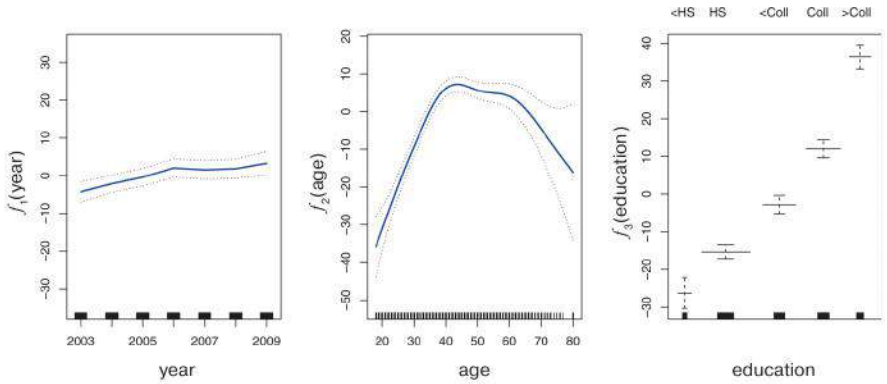

РИСУНОК 7.12. *То же, что и на рис. 7.11, однако здесь f_1 и f_2 представлены сглаживающими сплайнами с четырьмя и пятью степенями свободы соответственно*

му предиктору X_j, в связи с чем мы можем автоматически моделировать нелинейные зависимости, с которыми стандартная линейная регрессия не справится. Это значит, что нам нет необходимости самостоятельно перебирать различные преобразования каждой переменной в отдельности.

▲ Нелинейные модели потенциально могут дать более точные предсказания отклика Y.

▲ Поскольку модель является аддитивной, мы по–прежнему можем установить величину эффекта каждого предиктора X_j на Y, фиксируя при этом эффекты всех остальных переменных. Следовательно, если мы заинтересованы в получении статистических выводов, то GAM обеспечивают для этого удобную форму.

▲ Гладкость функции f_j для переменной X_j можно выразить в виде числа степеней свободы.

◆ Основное ограничение GAM состоит в том, что модель является исключительно аддитивной. При наличии нескольких переменных могут быть упущены важные взаимодействия. Однако, как и в случае с линейной регрессией, мы можем самостоятельно добавить эффекты взаимодействия в соответствующую модель, включив в нее предикторы вида $X_j \times X_k$. Кроме того, мы можем добавить низкоразмерные функции взаимодействий вида $f_{jk}(X_j, X_k)$; их можно подогнать в виде двумерных сглаживающих функций, таких как локальная регрессия или двумерные сплайны (здесь не рассматриваются).

Для построения максимально общих моделей нам необходимо обратиться к еще более гибким методам, таким как случайные леса и бустинг, описанным в главе 8. GAM обеспечивают полезный компромисс между линейными и полностью непараметрическими моделями.

7.7.2 GAM для задач классификации

GAM можно также применять в ситуациях, когда отклик Y является качественным. Для простоты изложения здесь мы предположим, что Y принимает значения 1 или 0, и примем $p(X) = \Pr(Y = 1|X)$ в качестве условной (т. е. определяемой предикторами) вероятности того, что отклик равен единице. Вспомните логистическую регрессионную модель (4.6):

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p. \tag{7.17}$$

Логит представляет собой отношение вероятностей $P(Y = 1|X)$ и $P(Y = 0|X)$, которое в (7.17) представлено в виде линейной функции предикторов. Естественным способом расширения (7.17) на случаи нелинейных зависимостей является модель

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p). \tag{7.18}$$

Уравнение (7.18) — это обобщенная аддитивная логистическая регрессионная модель. Она обладает теми же преимуществами и недостатками, которые обсуждались в предыдущем подразделе для случая количественных откликов.

Построим GAM по данным `Wage` для предсказания вероятности того, что доход того или иного человека превышает 250 000$ в год. Эта GAM принимает следующий вид:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 \times \texttt{year} + f_2(\texttt{age}) + f_2(\texttt{education}), \tag{7.19}$$

где $p(X) = \Pr(\texttt{wage} > 250|\texttt{age}, \texttt{year}, \texttt{education})$.

Как и ранее, функция f_2 подгоняется на основе сглаживающего сплайна с пятью степенями свободы, а f_3 подгоняется как ступенчатая функция для индикаторных переменных, соответствующих разным уровням образования. Полученная модель приведена на рис. 7.13. Последний график выглядит подозрительно, особенно из–за очень широкого доверительного интервала для уровня `<HS`. Оказывается, что у людей из этой категории нет значений отклика, равных 1: никто с уровнем образования ниже средней школы не зарабатывает более 250 000$. Поэтому мы построим новую модель, исключив людей с образованием ниже средней школы. Полученный результат представлен на рис. 7.14. Как и на рис. 7.11 и 7.12, все три графика имеют одинаковые оси ординат. Это позволяет нам визуально оценить удельный вклад каждой переменной. Видно, что по сравнению с `year`, `age` и `education` гораздо больше влияют на вероятность оказаться в группе высокого достатка.

7.8 Лабораторная работа: нелинейные модели

В этой лабораторной работе мы повторно проанализируем данные `Wage`, рассмотренные в примерах на протяжении всей этой главы, и покажем,

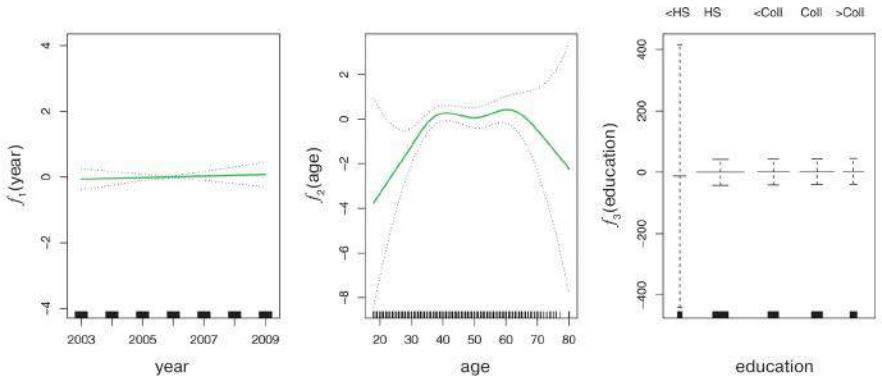

РИСУНОК 7.13. *Логистическая регрессионная GAM из (7.19) для бинарного отклика* `I(wage>250)` *построена по данным из таблицы* `Wage`. *На каждом графике приведены подогнанная функция и ее стандартные ошибки в каждой точке. Первая функция является линейной по* `year`, *вторая функция представляет собой сглаживающий сплайн с пятью степенями свободы для переменной* `age`, *а третья — это ступенчатая функция для* `education`. *Первый уровень переменной* `education` *(*`<HS`*) имеет очень большую стандартную ошибку*

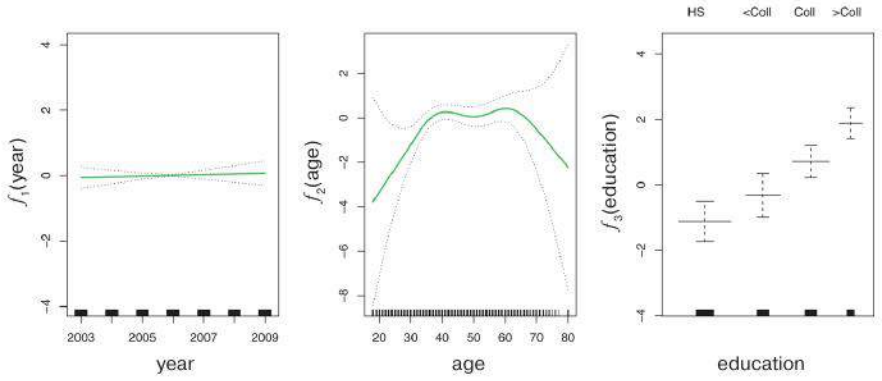

РИСУНОК 7.14. *Подогнана та же модель, что и на рис. 7.13, однако здесь были исключены наблюдения, у которых значение* `education` *равно* `<HS`. *Теперь мы видим, что более высокое образование сопряжено с более высокими уровнями заработной платы*

что многие из описанных сложных процедур подгонки нелинейных моделей можно легко реализовать в `R`. Мы начинаем с подключения библиотеки `ISLR`, содержащей эти данные.

```
> library(ISLR)
> attach(Wage)
```

7.8.1 Полиномиальная регрессия и ступенчатые функции

Теперь мы рассмотрим, как был создан рис. 7.1. Сначала мы подгоняем модель при помощи следующей команды:

```
> fit = lm(wage ~ poly(age, 4), data = Wage)

> coef(summary(fit))
                Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)     111.704       0.729   153.28    <2e-16
poly(age, 4)1   447.068      39.915    11.20    <2e-16
poly(age, 4)2 -478.316       39.915   -11.98    <2e-16
poly(age, 4)3   125.522      39.915     3.14    0.0017
poly(age, 4)4   -77.911      39.915    -1.95    0.0510
```

Этот синтаксис позволяет построить линейную модель при помощи функции `lm()` для предсказания `wage` на основе полинома четвертой степени `age` — `poly(age, 4)`. Функция `poly()` позволяет нам избежать необходимости набирать на клавиатуре длинную формулу со степенями `age`. Эта функция возвращает матрицу, столбцы которой соответствуют базисам *ортогональных полиномов*; по сути, это означает, что каждый столбец является линейной комбинацией переменных `age`, `age^2`, `age^3` и `age^4`.

ортогональные полиномы

Однако при желании мы также можем применить `poly()` для получения исходных значений `age`, `age^2`, `age^3` и `age^4`. Это можно сделать при помощи аргумента `raw = TRUE` функции `poly()`. Позднее мы увидим, что это не оказывает какого–либо существенного влияния на модель: хотя выбор базиса, конечно, изменяет оценки коэффициентов, он не влияет на предсказываемые моделью значения.

```
> fit2 = lm(wage ~ poly(age, 4, raw = TRUE), data = Wage)
> coef(summary(fit2))
                         Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)            -1.84e+02    6.00e+01    -3.07  0.002180
poly(age, 4, raw = T)1  2.12e+01    5.89e+00     3.61  0.000312
poly(age, 4, raw = T)2 -5.64e-01    2.06e-01    -2.74  0.006261
poly(age, 4, raw = T)3  6.81e-03    3.07e-03     2.22  0.026398
poly(age, 4, raw = T)4 -3.20e-05    1.64e-05    -1.95  0.051039
```

Существует несколько эквивалентных способов подгонки этой модели, что подчеркивает гибкость синтаксиса формул `R`. Например:

```
> fit2a = lm(wage ~ age+I(age^2)+I(age^3)+I(age^4), data = Wage)
> coef(fit2a)
(Intercept)         age    I(age^2)    I(age^3)    I(age^4)
  -1.84e+02   2.12e+01   -5.64e-01    6.81e-03   -3.20e-05
```

Эта команда создает полиномиальные базисные функции просто «на лету», защищая при этом элементы вроде `age^2` при помощи «функции–упаковщика» `I()` (дело в том, что символ `age^2` имеет особое значение в формулах).

```
> fit2b = lm(wage ~ cbind(age, age^2, age^3, age^4), data = Wage)
```

Эта команда делает то же самое, но более компактно, используя функцию `cbind()` для построения матрицы из нескольких векторов; вызов любой подобной `cbind()` функции внутри формулы также имеет эффект создания «обертки».

Теперь мы сформируем последовательность значений `age`, для которых необходимо получить предсказания, а затем вызовем функцию общего назначения `predict()`, одновременно инструктируя ее о том, что мы хотим также рассчитать стандартные ошибки.

```
> agelims = range(age)
> age.grid = seq(from = agelims[1], to = agelims[2])
> preds = predict(fit, newdata = list(age = age.grid), se=TRUE)
> se.bands = cbind(preds$fit + 2*preds$se.fit,
                   preds$fit - 2*preds$se.fit)
```

Наконец, мы изображаем данные на графике и добавляем подогнанный полином четвертой степени.

```
> par(mfrow = c(1, 2), mar = c(4.5, 4.5, 1, 1),
      oma = c(0, 0, 4, 0))
> plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")
> title("Degree-4 Polynomial", outer = TRUE)
> lines(age.grid, preds$fit, lwd = 2, col = "blue")
> matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```

Здесь аргументы `mar` и `oma` при вызове функции `par()` позволяют нам контролировать параметры полей вокруг графиков, а функция `title()` создает заголовок рисунка, который простирается на ширину обоих графиков.

`title()`

Ранее мы отметили, что вне зависимости от того, создаются ли ортогональные базисные функции при вызове `poly()`, полученная модель в сущности останется той же. Что мы имеем в виду? Предсказываемые моделью значения в обоих случаях идентичны:

```
> preds2 = predict(fit2, newdata = list(age=age.grid), se = TRUE)
> max(abs(preds$fit - preds2$fit))
[1] 7.39e-13
```

При построении полиномиальной регрессионной модели мы должны определиться со степенью полинома. Один из способов сделать это заключается в проверке статистических гипотез. Сейчас мы построим модели, варьирующие от линейной до полинома пятой степени, и попытаемся определить наиболее простую модель для описания зависимости между `wage` и `age`. Мы применяем функцию `anova()`, которая выполняет *дисперсионный анализ* (ANOVA, на основе F–критерия) для проверки нулевой гипотезы о том, что модель $\mathcal{M}_1$ достаточна для объяснения данных, против альтернативной гипотезы о том, что требуется более сложная модель $\mathcal{M}_2$. Для применения функции `anova()` модели $\mathcal{M}_1$ и $\mathcal{M}_2$ должны быть *вложенными*[17]: предикторы в $\mathcal{M}_1$ должны представлять собой поднабор предикторов из $\mathcal{M}_2$. В нашем случае мы подгоняем пять различных моделей и последовательно сравниваем наиболее простую модель с наиболее сложной.

`anova()`

дисперсионный анализ

[17] В оригинале используется термин «nested». — *Прим. пер.*

```
> fit.1 = lm(wage ~ age, data = Wage)
> fit.2 = lm(wage ~ poly(age, 2), data = Wage)
> fit.3 = lm(wage ~ poly(age, 3), data = Wage)
> fit.4 = lm(wage ~ poly(age, 4), data = Wage)
> fit.5 = lm(wage ~ poly(age, 5), data = Wage)
> anova(fit.1, fit.2, fit.3, fit.4, fit.5)
Analysis of Variance Table
Model 1: wage ~ age
Model 2: wage ~ poly(age, 2)
Model 3: wage ~ poly(age, 3)
Model 4: wage ~ poly(age, 4)
Model 5: wage ~ poly(age, 5)
  Res.Df      RSS  Df  Sum of Sq       F  Pr(>F)
1   2998  5022216
2   2997  4793430   1     228786  143.59  <2e-16 ***
3   2996  4777674   1      15756    9.89  0.0017 **
4   2995  4771604   1       6070    3.81  0.0510 .
5   2994  4770322   1       1283    0.80  0.3697
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

P–значение, по которому линейная модель `Model 1` сравнивается с квадратичной моделью `Model 2`, практически равно нулю ($< 10^{-15}$), указывая на то, что линейной модели недостаточно для описания этих данных. P–значение, по которому квадратичная модель `Model 2` сравнивается с кубической моделью `Model 3`, также очень мало, и поэтому квадратичной модели тоже недостаточно. P–значение, по которому кубическая модель `Model 3` сравнивается с полиномом четвертой степени `Model 4`, примерно равно 5%, тогда как в полиноме пятой степени `Model 5`, по–видимому, нет необходимости, поскольку соответствующее p–значение составляет 0.37. Таким образом, получается, что полиномы третьей или четвертой степени обеспечивают адекватное описание данных, тогда как полиномы более низкой или более высокой степени не оправданы.

В этом примере мы могли бы получить p–значения в более лаконичном виде, применив вместо `anova()` функцию `poly()`, которая создает ортогональные полиномы.

```
> coef(summary(fit.5))
                Estimate  Std. Error   t value   Pr(>|t|)
(Intercept)     111.70      0.7288   153.2780  0.000e+00
poly(age, 5)1   447.07     39.9161    11.2002  1.491e-28
poly(age, 5)2  -478.32     39.9161   -11.9830  2.368e-32
poly(age, 5)3   125.52     39.9161     3.1446  1.679e-03
poly(age, 5)4   -77.91     39.9161    -1.9519  5.105e-02
poly(age, 5)5   -35.81     39.9161    -0.8972  3.697e-01
```

Заметьте, что p–значения идентичны предыдущим и что квадраты значений t–критерия равны соответствующим значениям F–критерия, полученным ранее при помощи `anova()`:

```
> (-11.983)^2
[1] 143.6
```

Однако метод ANOVA работает вне зависимости от того, используем ли мы ортогональные полиномы; он работает также в случаях, когда в нашей модели присутствуют другие переменные. Например, мы можем сравнить при помощи `anova()` следующие три модели:

```
> fit.1 = lm(wage ~ education + age, data = Wage)
> fit.2 = lm(wage ~ education + poly(age, 2), data = Wage)
> fit.3 = lm(wage ~ education + poly(age, 3), data = Wage)
> anova(fit.1, fit.2, fit.3)
```

В качестве альтернативы проверке гипотез мы могли бы выбрать степень полинома при помощи перекрестной проверки (см. главу 5).

Далее мы выясним, как можно предсказать вероятность того, что некоторый человек зарабатывает более 250 000$ в год. В основном мы будем действовать, как и раньше, за тем исключением, что сначала мы создадим подходящий вектор со значениями отклика, а затем применим функцию `glm()` и ее аргумент `family = "binomial"` для подгонки полиномиальной логистической регрессионной модели.

```
> fit = glm(I(wage > 250) ~ poly(age, 4),
            data = Wage, family = binomial)
```

Заметьте, что для создания этого бинарного отклика «на лету» мы опять применяем функцию `I()`. Исполнение выражения `wage > 250` приводит к созданию логической переменной со значениями `TRUE` и `FALSE`, которые `glm()` преобразует в бинарные значения, превращая `TRUE` в 1, а `FALSE` в 0.

Как и раньше, для получения предсказаний мы применяем функцию `predict()`.

```
> preds = predict(fit, newdata = list(age = age.grid), se = TRUE)
```

Однако вычисление доверительных интервалов является несколько более сложной задачей, чем в случае с линейной регрессией. По умолчанию для моделей, полученных при помощи `glm()`, предсказанные значения относятся к типу `type = "link"`, и именно этот тип мы используем здесь. Это означает, что мы получаем предсказания на логит–шкале, т. е. при подгонке модели

$$\log\left(\frac{\Pr(Y=1|X)}{1-\Pr(Y=1|X)}\right) = X\beta$$

предсказанные значения принимают форму $X\hat{\beta}$. Получаемые стандартные ошибки выражаются в тех же единицах измерения. Чтобы получить доверительные интервалы для $\Pr(Y=1|X)$, мы выполняем следующее преобразование:

$$\Pr(Y=1|X) = \frac{\exp X\beta}{1+\exp X\beta}.$$

```
> pfit = exp(preds$fit)/(1 + exp(preds$fit))
> se.bands.logit = cbind(preds$fit + 2 * preds$se.fit,
                         preds$fit - 2 * preds$se.fit)
> se.bands = exp(se.bands.logit)/(1 + exp(se.bands.logit))
```

Заметьте, что мы могли бы вычислить вероятности, сразу выбрав опцию `type = "response"` при вызове функции `predict()`.

```
> preds = predict(fit, newdata = list(age = age.grid),
                  type = "response", se = TRUE)
```

Однако соответствующие доверительные интервалы не имели бы смысла, поскольку некоторые из полученных нами вероятностей оказались бы отрицательными!

```
> plot(age, I(wage > 250), xlim = agelims,
       type = "n", ylim = c(0, .2))
> points(jitter(age), I((wage > 250) / 5), cex = .5,
         pch = "|", col = "darkgrey")
> lines(age.grid, pfit, lwd = 2, col = "blue")
> matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```

Мы изобразили значения `age`, соответствующие наблюдениям с `wage` $>$ 250, в виде серых «насечек» в верхней части графика, а значения, соответствующие наблюдениям с `wage` $<$ 250 — в виде серых «насечек» в нижней части графика. Чтобы немного раздвинуть точки с одинаковыми значениями `age` и избежать их перекрывания, мы применили функцию `jitter()`. Такой график часто называют *«графиком–щеткой»*[18].

`jitter()`

график–щетка

Для подгонки ступенчатых функций, обсуждавшихся в разделе 7.2, мы применяем функцию `cut()`.

`cut()`

```
> table(cut(age, 4))
(17.9, 33.5] (33.5, 49] (49, 64.5] (64.5, 80.1]
         750       1399        779           72
> fit = lm(wage ~ cut(age, 4), data = Wage)
> coef(summary(fit))
                        Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)                94.16        1.48    63.79  0.00e+00
cut(age, 4) (33.5, 49]     24.05        1.83    13.15  1.98e-38
cut(age, 4) (49, 64.5]     23.66        2.07    11.44  1.04e-29
cut(age, 4) (64.5, 80.1]    7.64        4.99     1.53  1.26e-01
```

Здесь функция `cut()` автоматически выбрала точки разрыва, равные 33.5, 49 и 64.5 годам. Мы могли бы также задать свои собственные точки разрыва при помощи опции `breaks`. Функция `cut()` возвращает упорядоченную качественную переменную, после чего функция `lm()` создает несколько индикаторных переменных для регрессионной модели. Категория `age` $<$ `33.5` исключается из анализа, что позволяет интерпретировать свободный член модели, равный 94 160\$, как среднюю заработную плату людей в возрасте менее 33.5 лет, а остальные коэффициенты — как средний прирост заработной платы в соответствующих возрастных группах. Мы можем вычислить предсказанные значения и построить графики тем же образом, как это было сделано для полиномиальной модели.

7.8.2 Сплайны

Для подгонки регрессионных сплайнов в `R` мы применяем библиотеку `splines`. В разделе 7.4 мы видели, что регрессионные сплайны можно

[18] В оригинале используется термин «rug plot». — *Прим. пер.*

создать, сконструировав подходящую случаю матрицу из базисных функ-
bs() ций. bs() генерирует готовую матрицу базисных функций для сплайнов с заданным пользователем набором узлов сочленения. По умолчанию создаются кубические сплайны. Построение сплайн–регрессии wage по age не составляет труда:

```
> library(splines)
> fit = lm(wage ~ bs(age, knots = c(25, 40, 60)), data = Wage)
> pred = predict(fit, newdata = list(age = age.grid), se = TRUE)
> plot(age, wage, col = "gray")
> lines(age.grid, pred$fit, lwd = 2)
> lines(age.grid, pred$fit + 2 * pred$se, lty = "dashed")
> lines(age.grid, pred$fit - 2 * pred$se, lty = "dashed")
```

Мы самостоятельно задали узлы сочленения, приходящиеся на значения возраста 25, 40 и 60 лет. Это приводит к созданию сплайна с шестью базисными функциями. (Вспомните, что кубический сплайн с тремя узлами сочленения имеет 7 степеней свободы, которые расходуются на свободный член и шесть базисных функций.) Мы могли бы также воспользоваться опцией df для получения сплайна с узлами сочленения, приходящимися на равномерно распределенные квантили данных.

```
> dim(bs(age, knots = c(25, 40, 60)))
[1] 3000 6
> dim(bs(age, df = 6))
[1] 3000 6
> attr(bs(age, df = 6), "knots")
 25%   50%   75%
33.8  42.0  51.0
```

В данном случае R выбирает узлы сочленения в точках 33.8, 42.0 и 51.0 лет, которые соответствуют 25–му, 50–му и 75–му процентилям переменной age. Функция bs() имеет также аргумент degree, благодаря которому мы можем построить сплайн любой степени, отличающейся от принятой по умолчанию 3–й степени (кубический сплайн).

Для построения натурального сплайна мы применяем функцию ns().
ns() Ниже мы подгоняем натуральный сплайн с четырьмя степенями свободы.

```
> fit2 = lm(wage ~ ns(age, df = 4), data = Wage)
> pred2 = predict(fit2, newdata = list(age = age.grid), se = T)
> lines(age.grid, pred2$fit, col = "red", lwd = 2)
```

Как и в случае с функцией bs(), мы могли бы самостоятельно задать узлы сочленения при помощи опции knots.

smooth. Для подгонки гладкого сплайна мы применяем функцию
spline() smooth.spline(). Рис. 7.8 был получен при помощи следующего кода:

```
> plot(age, wage, xlim = agelims,
       cex = .5, col = "darkgrey")
> title("Smoothing Spline")
> fit = smooth.spline(age, wage, df = 16)
> fit2 = smooth.spline(age, wage, cv = TRUE)
> fit2$df
```

```
[1] 6.8
> lines(fit, col = "red", lwd = 2)
> lines(fit2, col = "blue", lwd = 2)
> legend("topright", legend = c("16 DF", "6.8 DF"),
        col = c("red", "blue"), lty = 1,
                lwd = 2, cex = .8)
```

Заметьте, что при первом вызове `smooth.spline()` мы указали `df = 16`. Благодаря этому функция находит значение λ, приводящее к 16 степеням свободы. Вызывая `smooth.spline()` второй раз, мы выбираем степень гладкости λ путем перекрестной проверки, что приводит к 6.8 степеням свободы.

Для выполнения локальной регрессии мы применяем функцию `loess()`. `loess()`

```
> plot(age, wage, xlim = agelims, cex = .5, col = "darkgrey")
> title("Local Regression")
> fit = loess(wage ~ age, span = .2, data = Wage)
> fit2 = loess(wage ~ age, span = .5, data = Wage)
> lines(age.grid, predict(fit, data.frame(age = age.grid)),
       col = "red", lwd = 2)
> lines(age.grid, predict(fit2, data.frame(age = age.grid)),
       col = "blue", lwd = 2)
> legend("topright", legend = c("Span = 0.2", "Span = 0.5"),
        col = c("red", "blue"), lty = 1, lwd = 2, cex = .8)
```

Здесь мы построили локальную регрессионную модель, используя значения ширины окна 0.2 и 0.5, т. е. каждая окрестность целевой точки включает 20% и 50% обучающих наблюдений. Чем больше ширина окна, тем более гладкой будет модель. Для подгонки локальных регрессионных моделей в `R` можно также применять библиотеку `locfit`.

7.8.3 GAM

Теперь мы построим GAM для предсказания `wage` на основе натуральных сплайн–функций `age` и `year`, рассматривая `education` как качественную переменную (см. уравнение (7.16)). Поскольку это всего лишь большая линейная регрессионная модель на основе соответствующего набора базисных функций, мы можем сделать это просто при помощи функции `lm()`.

```
> gam1 = lm(wage ~ ns(year, 4)+ns(age, 5)+education, data=Wage)
```

Теперь мы построим модель (7.16), используя сглаживающие, а не натуральные сплайны. Для создания более общих разновидностей GAM на основе сглаживающих сплайнов или других строительных блоков, которые не могут быть выражены в виде базисных функций и рассчитаны по методу наименьших квадратов, нам потребуется библиотека `gam`.

Функция `s()` из библиотеки `gam` служит для построения сглаживающего сплайна. При ее вызове мы указываем, что сглаживающие функции `year` и `age` должны иметь 4 и 5 степеней свободы соответственно. Поскольку `education` является качественной переменной, мы оставляем ее как есть — она будет автоматически преобразована в четыре индикаторные переменные. Для построения GAM на основе этих составляющих частей

мы применяем функцию gam(). Все члены уравнения (7.16) вычисляются одновременно; при этом учитываются их взаимные эффекты на отклик.

```
> library(gam)
> gam.m3 = gam(wage ~ s(year, 4) + s(age, 5) +
                education, data = Wage)
```

Для создания рис. 7.12 мы просто применяем функцию plot():

```
> par(mfrow = c(1, 3))
> plot(gam.m3, se = TRUE, col = "blue")
```

Функция общего назначения plot() распознает, что gam.m3 — это объект класса gam, и вызывает соответствующий метод plot.gam(). Очень удобно, что, несмотря на принадлежность gam1 к классу lm (а не gam), мы *можем* применить к этому объекту функцию plot.gam(). Рисунок 7.11 был получен при помощи следующей команды:

plot.gam()

```
> plot.gam(gam1, se = TRUE, col = "red")
```

Заметьте, что здесь мы воспользовались plot.gam(), а не функцией *общего* назначения plot().

На этих графиках функция year похожа на линейную. Мы можем выполнить серию тестов ANOVA для определения оптимальной модели среди следующих трех кандидатов: GAM без year ($\mathcal{M}_1$), GAM на основе линейной функции year ($\mathcal{M}_2$) и GAM на основе сплайн–функции year ($\mathcal{M}_3$).

```
> gam.m1 = gam(wage ~ s(age, 5) + education, data = Wage)
> gam.m2 = gam(wage ~ year + s(age, 5) + education, data = Wage)
> anova(gam.m1, gam.m2, gam.m3, test = "F")
Analysis of Deviance Table

Model 1: wage ~ s(age, 5) + education
Model 2: wage ~ year + s(age, 5) + education
Model 3: wage ~ s(year, 4) + s(age, 5) + education
  Resid. Df  Resid. Dev Df  Deviance    F    Pr(>F)
1      2990     3711730
2      2989     3693841  1     17889  14.5  0.00014 ***
3      2986     3689770  3      4071  1.1   0.34857
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Как видим, имеется убедительное свидетельство в пользу того, что GAM с линейной функцией year лучше, чем GAM, которая вообще не содержит year (p–значение = 0.00014). Однако нет оснований утверждать, что необходима нелинейная функция year (p–значение = 0.349). Другим словами, согласно результатам этого дисперсионного анализа, предпочтение следует отдать модели $\mathcal{M}_2$.

Функция summary() выдает развернутые результаты по полученной модели.

```
> summary(gam.m3)
Call: gam(formula = wage ~ s(year, 4) + s(age, 5) +
          education, data = Wage)
```

```
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-119.43  -19.70   -3.33   14.17  213.48

(Dispersion Parameter for gaussian family taken to be 1236)

    Null Deviance: 5222086 on 2999 degrees of freedom
Residual Deviance: 3689770 on 2986 degrees of freedom

AIC: 29888

Number of Local Scoring Iterations: 2

DF for Terms and F-values for Nonparametric Effects

                        Df  Npar Df  Npar F   Pr(F)
(Intercept)  1
s(year, 4)   1         3      1.1    0.35
s(age, 5)    1         4     32.4  <2e-16 ***
education    4
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

P–значения для year и age соответствуют нулевой гипотезе о наличии линейной связи (против гипотезы о существовании нелинейной связи). Большое p–значение для year еще раз подтверждает результаты дисперсионного анализа в том, что для этой переменной наиболее подходящей является линейная функция. Однако есть весомое основание для использования нелинейной функции для age.

Мы можем получить предсказания на основе gam–объектов точно так же, как и на основе lm–объектов, воспользовавшись методом predict() для класса gam. Здесь мы вычислим предсказания по обучающим данным.

```
> preds = predict(gam.m2, newdata = Wage)
```

Благодаря функции lo() в качестве строительных блоков GAM мы можем использовать также локальные регрессионные функции. lo()

```
> gam.lo = gam(wage ~ s(year, df = 4) + lo(age, span = 0.7) +
               education, data = Wage)
> plot.gam(gam.lo, se = TRUE, col = "green")
```

Здесь мы применили локальную регрессионную функцию для переменной age с шириной окна 0.7. Функцию lo() можно применить также для добавления эффектов взаимодействий предикторов при вызове функции gam(). Например, команда

```
> gam.lo.i = gam(wage ~ lo(year, age, span = 0.5) +
                 education, data = Wage)
```

строит модель с двумя составляющими, первая из которых описывает взаимодействие между year и age, представленное в виде локальной регрессионной плоскости. Мы можем построить график полученной двумерной плоскости, предварительно установив пакет akima.

```
> library(akima)
> plot(gam.lo.i)
```

При построении логистической GAM с помощью `gam()` мы снова применяем функцию `I()`, а также добавляем аргумент `family = binomial`.

```
> gam.lr = gam(I(wage > 250) ~ year + s(age, df = 5) +
               education, family = binomial, data = Wage)
> par(mfrow = c(1, 3))
> plot(gam.lr, se = TRUE, col = "green")
```

Нетрудно увидеть, что в категории <HS нет никого с высоким достатком:

```
> table(education, I(wage > 250))

education          FALSE  TRUE
1. < HS Grad         268     0
2. HS Grad           966     5
3. Some College      643     7
4. College Grad      663    22
5. Advanced Degree   381    45
```

Поэтому мы подгоняем логистическую GAM, используя все категории `education`, кроме этой. Результаты оказываются более целесообразными.

```
> gam.lr.s = gam(I(wage > 250) ~ year + s(age, df = 5) +
                 education, family = binomial, data = Wage,
                 subset = (education != "1. < HS Grad"))
> plot(gam.lr.s, se = TRUE, col = "green")
```

7.9 Упражнения

Теоретические

1. В этой главе было отмечено, что кубический регрессионный сплайн с одним узлом сочленения ξ можно получить при помощи базисных функций вида $x, x^2, x^3, (x-\xi)^3_+$, где член $(x-\xi)^3_+$ равен $(x-\xi)^3$, если $x > \xi$, и нулю в остальных случаях. Теперь мы покажем, что функция вида

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3_+$$

действительно является кубическим регрессионным сплайном, вне зависимости от значений $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

(a) Найдите такой кубический полином

$$f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3,$$

у которого $f(x) = f_1(x)$ для всех $x \leq \xi$. Выразите a_1, b_1, c_1, d_1 в виде $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$.

(b) Найдите такой кубический полином

$$f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3,$$

у которого $f(x) = f_2(x)$ для всех $x > \xi$. Выразите a_2, b_2, c_2, d_2 в виде $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$. Мы установили, что $f(x)$ — это кусочный полином.

(c) Покажите, что $f_1(\xi) = f_2(\xi)$. Другими словами, покажите, что функция $f(x)$ непрерывна в точке ξ.

(d) Покажите, что $f_1'(\xi) = f_2'(\xi)$. Другими словами, покажите, что функция $f'(x)$ непрерывна в точке ξ.

(e) Покажите, что $f_1''(\xi) = f_2''(\xi)$. Другими словами, покажите, что функция $f''(x)$ непрерывна в точке ξ.

Таким образом, $f(x)$ действительно является кубическим сплайном.

Подсказка: задачи (d) и (e) требуют знания интегрального исчисления для одной переменной. Напомним, что в случае с кубическим полиномом

$$f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3$$

первая производная принимает форму

$$f_1'(x) = b_1 + 2c_1 x + 3d_1 x^2,$$

а вторая производная принимает форму

$$f_1''(x) = 2c_1 + 6d_1 x.$$

2. Допустим, что мы вычисляем кривую $\hat{g}$ для гладкой аппроксимации некоторого набора из n точек по следующей формуле:

$$\hat{g} = \arg\min_g \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int \left[g^{(m)}(x) \right]^2 dx \right),$$

где $g^{(m)}$ — это m–я производная g, а $g^{(0)} = g$. Нарисуйте набросок того, как примерно будет выглядеть $\hat{g}$ в каждом из следующих сценариев:

(a) $\lambda = \infty,\ m = 0.$

(b) $\lambda = \infty,\ m = 1.$

(c) $\lambda = \infty,\ m = 2.$

(d) $\lambda = \infty,\ m = 3.$

(e) $\lambda = 0,\ m = 3.$

3. Предположим, что мы подгоняем кривую с базисными функциями $b_1(X) = X$, $b_2(X) = (X-1)^2 I(X \geq 1)$. (Заметьте, что выражение $I(X \geq 1)$ равно 1, если $X \geq 1$, и нулю в остальных случаях.) Мы подгоняем линейную регрессионную модель

$$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \epsilon$$

и получаем оценки коэффициентов $\hat{\beta}_0 = 1$, $\hat{\beta}_1 = 1$ и $\hat{\beta}_2 = -2$. Нарисуйте набросок того, как оцененная кривая будет выглядеть на интервале значений X от -2 до 2. Учтите свободный член уравнения, угловые коэффициенты и другую важную информацию.

4. Представьте, что мы подгоняем кривую с базисными функциями $b_1(X) = I(0 \leq X \leq 2) - (X-1)I(1 \leq X \leq 2)$ и $b_2(X) = (X-3)I(3 \leq X \leq 4) + I(4 < X \leq 5)$. Мы подгоняем линейную регрессионную модель

$$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \epsilon$$

и получаем оценки коэффициентов $\hat{\beta}_0 = 1$, $\hat{\beta}_1 = 1$ и $\hat{\beta}_2 = 3$. Нарисуйте набросок того, как оцененная кривая будет выглядеть на интервале значений X от -2 до 2. Учтите свободный член уравнения, угловые коэффициенты и другую важную информацию.

5. Рассмотрим две кривые $\hat{g}_1$ и $\hat{g}_2$ вида

$$\hat{g}_1 = \arg\min_g \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int \left[g^{(3)}(x) \right]^2 dx \right),$$

$$\hat{g}_2 = \arg\min_g \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int \left[g^{(4)}(x) \right]^2 dx \right),$$

где $g^{(m)}$ — это m–я производная g.

   (a) При $\lambda \to \infty$ у какой из этих кривых RSS на обучающих данных будет меньше?

   (b) При $\lambda \to \infty$ у какой из этих кривых RSS на контрольных данных будет меньше?

   (c) При $\lambda = 0$ у какой из этих кривых RSS на обучающих и контрольных данных будет меньше?

Практические

6. В этом упражнении мы продолжим анализ набора данных `Wage`, использованного на протяжении всей этой главы.

(a) Постройте полиномиальную регрессионную модель для предсказания `wage` по `age`. Выполните перекрестную проверку для выбора оптимальной степени полинома d. Какая степень была выбрана и насколько этот результат сравним с результатами проверки гипотез при помощи дисперсионного анализа? Постройте график полученной полиномиальной модели.

(b) Подгоните ступенчатую функцию для предсказания `wage` на основе `age` и выполните перекрестную проверку для нахождения оптимального числа интервалов для разбивки данных. Постройте график полученной полиномиальной модели.

7. Таблица `Wage` содержит ряд других, не рассмотренных в этой главе переменных, таких как семейное положение (`maritl`), занимаемая должность (`jobclass`) и др. Исследуйте взаимоотношения между некоторыми из этих дополнительных предикторов и `wage` и примените нелинейные статистические методы для подгонки гибких моделей по этим данным. Постройте графики созданных моделей и напишите краткий отчет о полученных результатах.

8. Постройте некоторые из рассмотренных в этой главе моделей, используя данные из таблицы `Auto`. Есть ли указания на существование нелинейных зависимостей в этих данных? Постройте несколько графиков, подтверждающих ваш ответ.

9. В этой задаче задействованы переменные `dis` (взвешенные средние расстояния до пяти центров по трудоустройству в Бостоне) и `nox` (концентрация оксидов азота, выраженная в частях на 10 миллионов) из таблицы данных `Boston`. Мы будем рассматривать `dis` в качестве предиктора, а `nox` — в качестве отклика.

(a) Воспользуйтесь функцией `poly()` для подгонки кубической полиномиальной регрессии для предсказания `nox` по `dis`. Опишите полученные результаты и изобразите графически исходные данные и построенную модель.

(b) Изобразите полиномиальные модели нескольких различных степеней (например, от 1 до 10) и рассчитайте соответствующие значения сумм квадратов остатков.

(c) Примените перекрестную проверку или какой–либо другой метод для выбора оптимальной степени полинома и объясните свои результаты.

(d) Примените функцию `bs()` для подгонки регрессионного сплайна, предсказывающего `nox` по `dis`. Опишите получаемые результаты для модели с четырьмя степенями свободы. Как вы выбрали узлы сочленения? Изобразите итоговую модель на графике.

(e) Теперь подгоните регрессионный сплайн для нескольких значений числа степеней свободы, а затем изобразите построенные модели на графике и сообщите соответствующие значения RSS. Опишите полученные результаты.

(f) Примените перекрестную проверку или какой–либо другой метод для выбора оптимального числа степеней свободы для регрессионного сплайна, подгоняемого к этим данным. Опишите свои результаты.

10. Эта задача касается набора данных `College`.

(a) Разбейте данные на обучающую и проверочную выборки. Используя размер оплаты за обучение для студентов из других штатов в качестве зависимой переменной, а все остальные переменные в качестве предикторов, выполните отбор с пошаговым включением переменных на обучающих данных и найдите удовлетворительную модель, в которую входят только некоторые из имеющихся предикторов.

(b) Постройте GAM по обучающим данным, используя размер оплаты за обучение для студентов из других штатов в качестве зависимой переменной, а переменные, выбранные на предыдущем шаге, — в качестве предикторов. Изобразите результаты графически и объясните их.

(c) Оцените полученную модель на проверочной выборке и объясните свои результаты.

(d) Для каких переменных (если таковые имеются) есть свидетельство в пользу их нелинейной связи с откликом?

11. В разделе 7.7 было отмечено, что обобщенные аддитивные модели обычно подгоняют при помощи *метода настройки с возвращениями*. Идея, лежащая в основе этого метода, на самом деле очень проста. Сейчас мы разберемся с ним на примере множественной линейной регрессии.

Представьте, что мы хотели бы построить множественную линейную регрессию, но для этого у нас нет подходящего программного обеспечения. Все, что у нас есть, — это программа для вычисления простой линейной регрессии. Поэтому мы применим следующий итеративный подход: несколько раз подряд мы повторим процедуру, в ходе которой будем фиксировать текущие значения всех коэффициентов, за исключением одного, чью оценку мы будем обновлять при помощи простой линейной регрессии. Этот процесс продолжают до *схождения*, т. е. момента, когда оценки коэффициентов перестают изменяться.

Попробуем проделать это на имитированных данных.

(a) Сгенерируйте значения отклика Y и двух предикторов X_1 и X_2 для $n = 100$.

(b) Присвойте $\hat{\beta}_1$ любое начальное значение на ваш выбор. Не важно, какое именно значение вы выберете.

(c) Зафиксировав значение $\hat{\beta}_1$, рассчитайте модель

$$Y - \hat{\beta}_1 X_1 = \beta_0 + \beta_2 X_2 + \epsilon.$$

Это можно сделать следующим образом:

```
> a = y - beta1 * x1
> beta2 = lm(a ~ x2)$coef[2]
```

(d) Зафиксировав значение $\hat{\beta}_2$, рассчитайте модель

$$Y - \hat{\beta}_2 X_1 = \beta_0 + \beta_1 X_2 + \epsilon.$$

Это можно сделать следующим образом:

```
> a = y - beta2 * x2
> beta1 = lm(a ~ x1)$coef[2]
```

(e) Напишите цикл, чтобы повторить шаги (c) и (d) 1000 раз. Сохраните оценки $\hat{\beta}_0$, $\hat{\beta}_1$ и $\hat{\beta}_2$, полученные в ходе каждой итерации цикла. Изобразите эти значения на графике, используя разные цвета для $\hat{\beta}_0$, $\hat{\beta}_1$ и $\hat{\beta}_2$.

(f) Сравните свой ответ из (e) с результатами, которые можно получить путем непосредственной подгонки множественной линейной регрессии Y по X_1 и X_2. Примените функцию `abline()` для добавления оценок коэффициентов множественной линейной регрессии к графику, построенному в задаче (e).

(g) Сколько итераций потребовалось для этих данных, чтобы получить «хорошую» аппроксимацию оценок множественной регрессии?

12. Эта задача является продолжением предыдущего упражнения. На примере данных с $p = 100$ покажите, что оценки коэффициентов множественной линейной регрессии можно аппроксимировать путем многократной подгонки простой линейной регрессии по методу настройки с возвращениями. Сколько итераций требуется для получения «хорошей» аппроксимации оценок коэффициентов множественной регрессии? Постройте график, позволяющий обосновать ваш ответ.

Глава 8

Методы, основанные на деревьях решений

В этой главе мы опишем методы классификации и регрессии, основанные на *деревьях решений*[1]. Эти методы включают *стратификацию*, или *сегментирование*, пространства предикторов на несколько ограниченных областей. Чтобы сделать предсказание для некоторого наблюдения, мы обычно используем среднее значение или моду, рассчитанные по обучающим данным из области, к которой принадлежит это наблюдение. Поскольку набор правил, разбивающих пространство предикторов на сегменты, можно представить в виде древовидной структуры, соответствующие методы известны как *деревья решений*.

деревья решений

Методы, основанные на деревьях решений, просты и полезны с точки зрения их интерпретируемости. Однако обычно они не настолько хороши с точки зрения точности предсказаний, как методы, рассмотренные в главах 6 и 7. Поэтому в этой главе мы вводим также такие подходы, как *бэггинг*, *случайные леса* и *бустинг*[2]. Каждый из этих подходов включает построение большого количества деревьев решений, которые затем объединяются для получения одного согласованного предсказания. Мы увидим, что, несмотря на определенное снижение интерпретируемости, объединение большого числа деревьев часто приводит к существенному улучшению точности предсказаний.

8.1 Деревья решений: основные понятия

Деревья решений можно применять для задач как регрессии, так и классификации. Сначала мы рассмотрим регрессионные проблемы, а затем перейдем к классификации.

[1] В оригинале используется термин «decision trees». В русскоязычной литературе этот термин переводится также как «деревья принятия решений» и «решающие деревья». — *Прим. пер.*

[2] Устоявшегося перевода терминов «bagging» и «boosting» на русский язык не существует, в связи с чем обычно их просто приводят в транслитерации с английского. — *Прим. пер.*

8.1.1 Регрессионные деревья

Мы начнем рассмотрение *регрессионных деревьев*[3] с простого примера.

регрессионные деревья

Предсказание заработной платы бейсбольных игроков при помощи регрессионных деревьев

Мы воспользуемся набором данных `Hitters` для предсказания заработной платы бейсбольных игроков `Salary` на основе `Years` (количество лет, проведенных в крупных лигах) и `Hits` (количество хитов, сделанных в прошлом году). Сначала мы удалим наблюдения с пропущенными значениями `Salary`, а затем прологарифмируем `Salary`, чтобы сделать распределение значений этой переменной более похожим на привычный «колокол». (Напомним, что `Salary` выражается в тысячах долларов.)

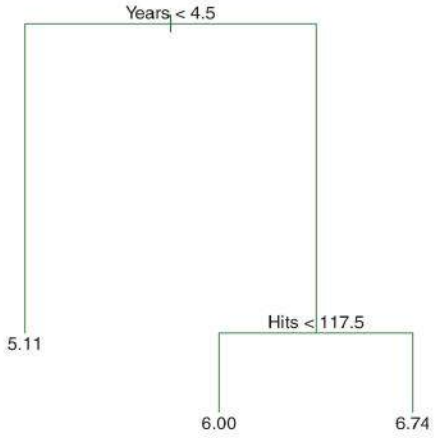

РИСУНОК 8.1. *Регрессионное дерево, построенное по данным* `Hitters` *для предсказания логарифма заработной платы бейсбольного игрока на основе количества лет, проведенного им в крупных лигах, и количества хитов, сделанных в предыдущем году. Для некоторого внутреннего узла дерева правило* $X_j < t_k$ *определяет левую ветвь, исходящую из этого узла, а правая ветвь соответствует правилу* $X_j \geq t_k$*. Так, разбиение на уровне корневого узла приводит к созданию двух крупных ветвей. Левая ветвь соответствует утверждению* `Years < 4.5`*, а правая — утверждению* `Years >= 4.5`*. У дерева есть два внутренних узла и три конечных узла («листья»). Число напротив конечного узла представляет собой среднее значение отклика у соответствующих наблюдений*

На рис. 8.1 показано регрессионное дерево, построенное по этим данным. Оно состоит из нескольких правил разбиений[4], начинающихся с кор-

[3] В оригинале используется термин «regression trees». — *Прим. пер.*

[4] Под правилом здесь понимают утверждение в отношении той или иной переменной, которое может быть истинным или ложным и которое, соответственно, позволяет разбить данные на сегменты (например, для части наблюдений будет справедливо утверждение Years < 4.5, а для другой — утверждение Years ≥ 4.5). В оригинале используется термин «splitting rule». — *Прим. пер.*

невого узла дерева. Корневое правило относит наблюдения со значениями `Years < 4.5` к левой ветви[5]. Предсказанные значения заработной платы у этих игроков соответствуют среднему значению отклика у всех рассматриваемых игроков с `Years < 4.5`. Среднее значение заработной платы у этих игроков на логарифмической шкале составляет 5.107, откуда мы можем вычислить предсказанное значение как $e^{5.107} = 165\,174\$$. Группу игроков, у которых `Years >= 4.5`, относят к правой ветви дерева, а затем разбивают ее еще раз на основе значений `Hits`. В целом это дерево разбивает игроков в пространстве предикторов на три сегмента: на тех, кто играл четыре или менее лет, тех, кто играл пять или более лет и сделал менее 118 хитов в прошлом году, а также тех, кто играл пять или более лет и сделал как минимум 118 хитов. Эти три области можно выразить математически как $R_1 = \{X|\texttt{Years < 4.5}\}$, $R_2 = \{X|\texttt{Years >= 4.5}, \texttt{Hits < 117.5}\}$ и $R_3 = \{X|\texttt{Years >= 4.5}, \texttt{Hits >= 117.5}\}$. На рис. 8.2 данные области показаны в координатах `Years` и `Hits`. Предсказанные значения для этих групп составляют $1000\$ \times e^{5.107} = 165\,174\$$, $1000\$ \times e^{5.999} = 402\,834\$$ и $1000\$ \times e^{6.740} = 845\,346\$$ соответственно.

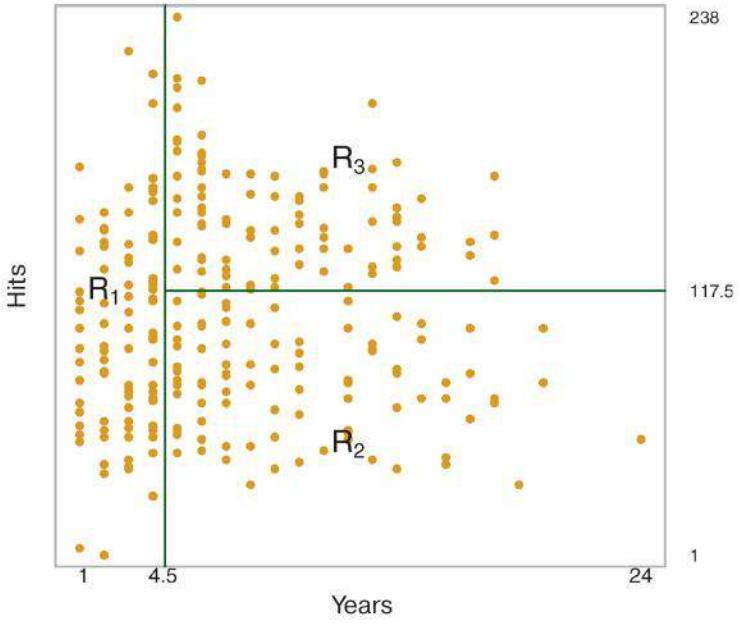

РИСУНОК 8.2. *Разбиение данных* `Hitters` *на три области в соответствии с регрессионным деревом, показанным на рис. 8.1*

конечный узел, лист

В продолжение аналогии с *деревьями* области R_1, R_2 и R_3 называют *конечными узлами*, или *листьями*[6]. Как показано на рис. 8.1, деревья

[5] В этом наборе данных переменные `Years` и `Hits` являются дискретными; функция `tree()` в `R` создает метки разбиений, используя среднее значение из двух соседних целых чисел.

[6] В оригинале используются термины «terminal nodes» и «leaves» соответственно. — *Прим. пер.*

решений обычно изображают в *перевернутом виде*, т. е. листья находятся в основании дерева. Точки вдоль дерева, в которых происходит разбиение пространства предикторов, называют *внутренними узлами*[7]. На рис. 8.1 два внутренних узла отмечены текстовыми метками `Years < 4.5` и `Hits < 117.5`. Отрезки, исходящие из узлов дерева, называют *ветвями*[8].

внутренний узел

ветвь

Мы могли бы интерпретировать регрессионное дерево на рис. 8.1 следующим образом: `Years` — это главный фактор, определяющий `Salary`, причем игроки с небольшим опытом зарабатывают меньше своих более опытных коллег. У игрока с небольшим опытом заработная плата, по–видимому, слабо связана с количеством хитов, сделанных им в прошлом году. Однако среди игроков, которые входили в состав крупных лиг на протяжении пяти или более лет, количество хитов за прошлый год оказывает влияние на заработную плату: чем больше это количество, тем выше заработная плата. Скорее всего, показанное на рис. 8.1 регрессионное дерево является упрощенным представлением зависимости между `Hits`, `Years` и `Salary`. Однако у него есть преимущество в сравнении с другими регрессионными моделями (вроде тех, которые мы видели в главах 3 и 6): его легче интерпретировать, и его можно очень удобно представить графически.

Предсказания, основанные на разбиении пространства предикторов на области

Теперь мы обсудим процедуру построения регрессионного дерева. В целом имеются два основных шага.

1. Мы разбиваем пространство предикторов, т. е. совокупность всех возможных значений $X_1, X_2, \ldots, X_p$, на J отдельных непересекающихся областей $R_1, R_2, \ldots, R_J$.

2. Для всех наблюдений, попадающих в область R_j, мы делаем одинаковое предсказание, которое просто равно среднему значению отклика у обучающих наблюдений из этой области.

Представим, например, что на шаге 1 мы получаем две области — R_1 и R_2 и что среднее значение отклика у обучающих наблюдений в первой области составляет 10, а во второй области — 20. Тогда для некоторого наблюдения $X = x$ мы предсказываем значение 10, если $x \in R_1$, и значение 20, если $x \in R_2$.

Теперь мы остановимся на шаге 1 подробнее. Как мы формируем области $R_1, \ldots, R_J$? Теоретически эти области могут принимать любую форму. Однако мы осознанно выбираем разбиение пространства предикторов на многомерные прямоугольники, или *контейнеры*[9], для простоты и легкости интерпретации итоговой предсказательной модели. Цель заключается в нахождении таких контейнеров, которые минимизируют RSS, рассчитываемую как

[7] В оригинале используется термин «internal nodes». — *Прим. пер.*

[8] В оригинале используется термин «branches». В русскоязычной литературе применяется также термин «ребро». — *Прим. пер.*

[9] В оригинале используется термин «boxes». — *Прим. пер.*

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{8.1}$$

где $\hat{y}_{R_j}$ — это среднее значение отклика у обучающих наблюдений из j–го контейнера. К сожалению, с вычислительной точки зрения невозможно рассмотреть все возможные разбиения пространства признаков на J контейнеров. По этой причине мы применяем *нисходящий жадный* алгоритм, известный как *рекурсивное бинарное разбиение*[10]. Этот алгоритм является *нисходящим*, поскольку он начинается с корневого узла дерева (точки, где все наблюдения принадлежат к одной области), а затем последовательно выполняет разбиение пространства предикторов; каждое разбиение изображается при помощи двух уходящих вниз новых ветвей. Этот алгоритм называется *жадным*, поскольку на каждом этапе процесса построения дерева выполняется разбиение, *оптимальное* для этого конкретного этапа, без «заглядывания вперед» и выбора разбиения, которое приведет к более качественной модели на некотором последующем шаге.

рекурсивное бинарное разбиение

Для выполнения рекурсивного бинарного разбиения мы сначала выбираем такие предиктор X_j и «точку разрыва» s, которые приводят к максимально возможному снижению RSS при разбиении пространства предикторов на области $\{X|X_j < s\}$ и $\{X|X_j \geq s\}$. (Нотация $\{X|X_j < s\}$ означает область пространства предикторов, в котором X_j принимает значение, не превышающее s.) Иными словами, мы перебираем все предикторы $X_1, \ldots, X_p$ и все возможные значения точки s для каждого предиктора, а затем выбираем такие предиктор и точку разрыва, которые приводят к созданию дерева с наименьшей RSS. Если быть точнее, то для любых j и s мы определяем пару полуплоскостей

$$R_1(j,s) = \{X|X_j < s\} \text{ и } R_2(j,s) = \{X|X_j \geq s\}, \tag{8.2}$$

и пытаемся найти значения j и s, которые минимизируют

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2, \tag{8.3}$$

где $\hat{y}_{R_1}$ — это среднее значение отклика у обучающих наблюдений из области $R_1(j,s)$, а $\hat{y}_{R_2}$ — это среднее значение отклика у обучающих наблюдений из области $R_2(j,s)$. Найти значения j и s, минимизирующие (8.3), можно довольно быстро, особенно когда количество предикторов p не очень велико.

Затем мы повторяем этот процесс, пытаясь найти оптимальные предиктор и точку разрыва с целью дальнейшего разбиения данных для минимизации RSS в пределах каждой получаемой области. Однако теперь вместо разбиения всего пространства предикторов мы разбиваем одну из двух областей, найденных на предыдущем этапе. Теперь у нас есть три области. Затем для минимизации RSS мы пытаемся выполнить дальнейшее разбиение уже одной из этих трех областей. Процесс повторяется до тех пор, пока не будет достигнут некоторый стоп–критерий; например, мы

[10] В оригинале используются термины «top–down greedy approach» и «recursive binary splitting» соответственно. — *Прим. пер.*

можем продолжать до тех пор, пока число наблюдений во всех областях составляет не менее пяти.

Получив области $R_1, \ldots, R_J$, мы предсказываем отклик для того или иного контрольного наблюдения, используя среднее значение отклика у обучающих наблюдений из области, к которой принадлежит это контрольное наблюдение.

Пример с пятью областями, полученными при помощи этого подхода, показан на рис. 8.3.

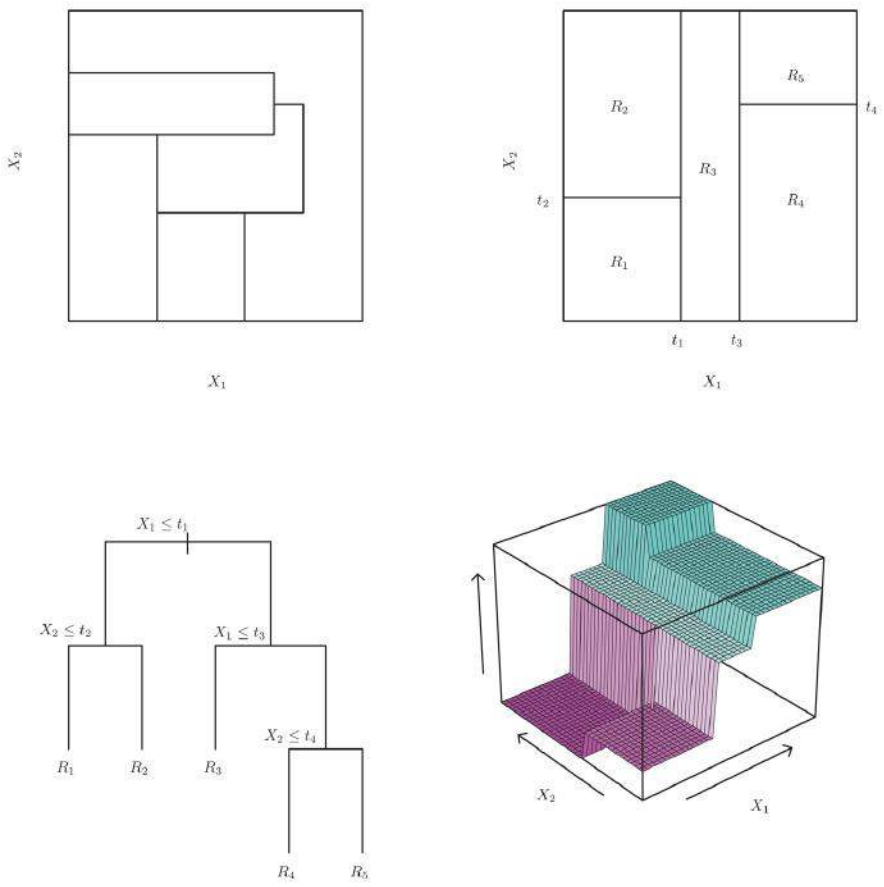

РИСУНОК 8.3. Слева вверху: *разбиение двумерного пространства признаков, которое невозможно было бы получить в результате применения метода рекурсивного бинарного разбиения.* Справа вверху: *результат, полученный при помощи рекурсивного бинарного разбиения на примере двумерного пространства признаков.* Слева внизу: *дерево, соответствующее разбиению, которое представлено на графике справа вверху.* Справа внизу: *трехмерное представление плоскости модельных значений, соответствующей этому дереву*

Обрезка ветвей дерева

Описанный выше процесс может дать хорошие предсказания на обучающей выборке, но с большой вероятностью будет сопровождаться переобучением, приводящим к низкому качеству предсказаний на контрольной выборке. Это обусловлено тем, что полученное дерево может оказаться чрезмерно сложным. Меньшее дерево с небольшим числом разбиений (т. е. меньшим числом областей $R_1, \ldots, R_J$) может снизить дисперсию и улучшить интерпретируемость за счет незначительного увеличения смещения. Одной из альтернатив описанному выше процессу является построение дерева только такой глубины, при которой снижение RSS в каждом узле превосходит некоторое (высокое) пороговое значение. Такая стратегия приведет к менее глубокому дереву, однако она недальновидна, поскольку за кажущимся ненужным разбиением в начале дерева могло бы последовать очень хорошее разбиение, приводящее к существенному снижению RSS.

обрезка

Поэтому более подходящей стратегией является выращивание очень большого дерева T_0 с последующей *обрезкой*[11] для получения его усеченной версии. Как нам найти оптимальный способ обрезки ветвей дерева? Интуитивно понятно, что наша цель заключается в нахождении такого обрезанного дерева, которое обеспечивает минимальную ошибку на контрольных данных. Для некоторого обрезанного дерева мы можем оценить ошибку на контрольной выборке при помощи перекрестной проверки или метода проверочной выборки. Однако вычисление ошибки путем перекрестной проверки для каждой возможной версии обрезанного дерева оказалось бы чрезмерно затруднительным, поскольку существует огромное множество таких возможных версий. Вместо этого нам нужен способ, позволяющий выбрать некоторый ограниченный набор подлежащих рассмотрению обрезанных деревьев.

обрезка с учетом штрафа за сложность

Обрезка с учетом штрафа за сложность, известная также как *обрезка наиболее слабых ветвей*[12], как раз таким способом и является. Вместо перебора всех возможных обрезанных деревьев мы рассматриваем только последовательность деревьев, полученных с учетом значения некоторого неотрицательного гиперпараметра α.

Каждому значению α соответствует такое обрезанное дерево $T \subset T_0$, для которого величина

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \tag{8.4}$$

минимальна. Здесь $|T|$ — это число конечных узлов у дерева T, R_m — контейнер (область в пространстве предикторов), соответствующий m–у конечному узлу, а $\hat{y}_{R_m}$ — предсказанное значение отклика в области R_m, т. е. среднее значение отклика у обучающих наблюдений в R_m. Гиперпараметр α контролирует баланс между сложностью обрезанного дерева и точностью аппроксимации обучающих данных. При $\alpha = 0$ обрезанное дерево T будет просто равно T_0, поскольку в этом случае (8.4) всего лишь

[11] В оригинале используется термин «pruning». — *Прим. пер.*

[12] В оригинале используются термины «cost complexity pruning» и «weakest link pruning» соответственно. — *Прим. пер.*

отражает ошибку на обучающих данных. Однако по мере увеличения α иметь дерево с большим числом конечных узлов становится накладно, в связи с чем величина (8.4) обычно будет достигать минимума у деревьев меньшего размера. Уравнение (8.4) напоминает уравнение (6.7) из главы 6 для лассо–регрессии, в котором параметр, контролирующий сложность линейной модели, был задан похожим образом.

Оказывается, что по мере увеличения α в (8.4) с нуля ветви дерева последовательно обрезаются предсказуемым образом, благодаря чему получение целой серии деревьев разного размера как функции от α не составляет труда. Мы можем выбрать значение α при помощи методов проверочной выборки или перекрестной проверки. После этого мы возвращаемся к полному набору данных и получаем обрезанное дерево, соответствующее найденному значению α. Этот процесс обобщен в алгоритме 8.1.

Алгоритм 8.1 Построение регрессионного дерева

---

1. Примените рекурсивное бинарное разбиение к обучающим данным для построения глубоко ветвящегося дерева; остановите процесс построения дерева в момент, когда количество наблюдений в каждом конечном узле окажется меньше некоторой минимальной пороговой величины.

2. Выполните обрезку ветвей большого дерева с учетом разных значений штрафного параметра α и получите серию оптимальных деревьев меньшего размера.

3. Примените K–кратную перекрестную проверку для выбора оптимального значения α. Иными словами, разбейте обучающие данные на K блоков. Для каждого блока $k = 1, \ldots, K$:

   (a) повторите шаги 1 и 2 для всех блоков обучающих данных, за исключением k–го блока;

   (b) вычислите среднеквадратичную ошибку предсказаний на данных из k–го блока для разных значений α. Усредните результаты для каждого значения α и выберите такое значение этого параметра, которое минимизирует среднюю ошибку.

4. Выберите обрезанное дерево из шага 2, соответствующее оптимальному значению α.

На рис. 8.4 и 8.5 показаны результаты построения и обрезки регрессионного дерева для данных `Hitters` с использованием девяти предикторов. Сначала мы случайным образом разделили данные пополам, что дало 132 наблюдения в обучающей выборке и 131 наблюдение в контрольной выборке. Затем мы построили большое регрессионное дерево по обучающим данным и применили несколько разных значений α для получения обрезанных деревьев с разным числом конечных узлов. Наконец, мы выполнили шестикратную перекрестную проверку для оценивания MSE на контрольных данных в зависимости от α. (Мы решили использовать

шестикратную перекрестную проверку, поскольку число 132 в точности делится на 6.) Необрезанное регрессионное дерево показано на рис. 8.4. Зеленая кривая на рис. 8.5 изображает ошибку перекрестной проверки в зависимости от числа конечных узлов[13], тогда как оранжевая кривая показывает ошибку на контрольных данных. Приведены также стандартные ошибки оцененных ошибок. Для сравнения черным цветом показана кривая ошибок на обучающих данных. Ошибка перекрестной проверки, достигающая минимума при трех конечных узлах, хорошо аппроксимирует ошибку на контрольных данных: последняя также снижается в районе трех конечных узлов (хотя наименьшее ее значение наблюдается при 10 узлах). Обрезанное дерево с тремя конечными узлами показано на рис. 8.1.

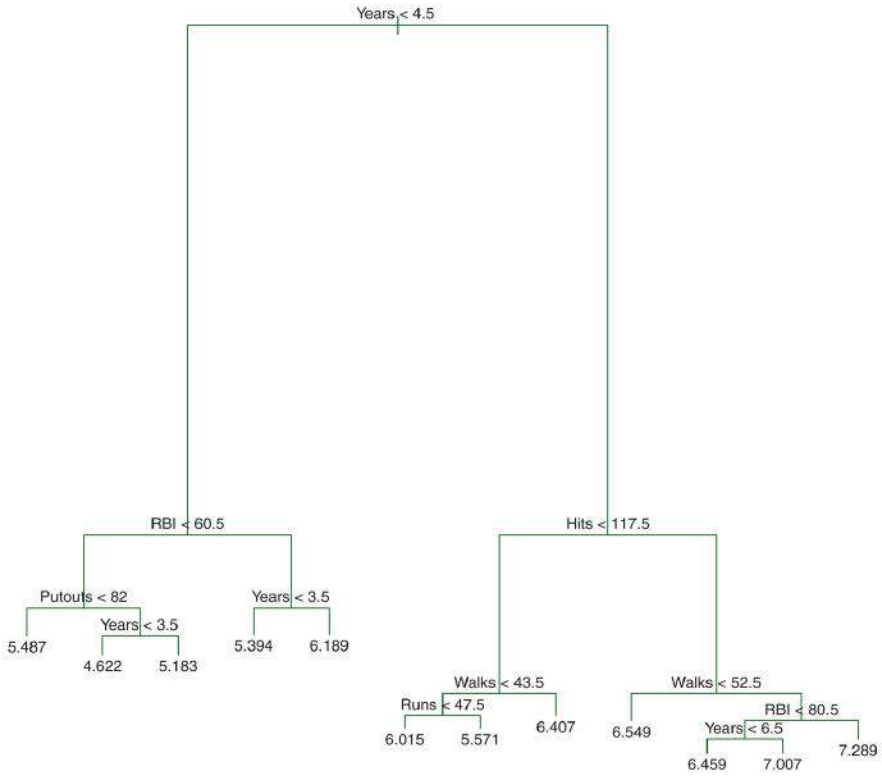

РИСУНОК 8.4. *Регрессионное дерево, построенное по данным* `Hitters`. *Показано необрезанное дерево, полученное в результате применения алгоритма рекурсивного бинарного разбиения к обучающим данным*

[13] Хотя ошибка перекрестной проверки вычисляется как функция от α, результат удобно показывать как функцию от $|T|$ — числа конечных узлов; этот подход возможен благодаря связи между α и $|T|$ в исходном дереве, построенном по всем обучающим данным.

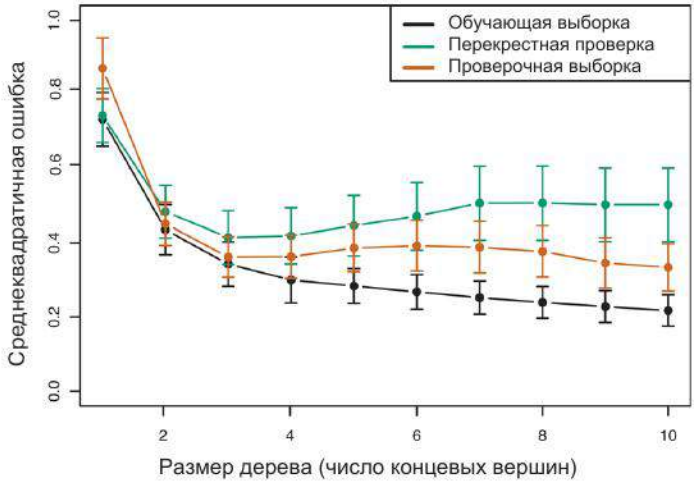

РИСУНОК 8.5. *Регрессионный анализ данных* `Hitters` *с помощью деревьев решений. Оценки MSE на обучающих данных, на контрольных данных, а также оценки, полученные в ходе перекрестной проверки, показаны в зависимости от числа конечных узлов у обрезанного дерева. Приведены также стандартные ошибки. Минимальная ошибка перекрестной проверки наблюдается у дерева с тремя конечными узлами*

8.1.2 Деревья классификации

дерево классификации

Дерево классификации очень похоже на регрессионное дерево, однако применяется для предсказания качественного, а не количественного отклика. Вспомните, что для регрессионного дерева предсказанный отклик представляет собой среднее значение обучающих наблюдений, принадлежащих к соответствующему конечному узлу. В случае же дерева классификации мы относим то или иное наблюдение к *наиболее часто встречающемуся классу* в области, в которую оно попадает. При интерпретации результатов, полученных на основе дерева классификации, нас часто интересует не только предсказание метки класса, но и *частота встречаемости* наблюдений из этого класса в соответствующей области.

Задача по построению дерева классификации довольно похожа на задачу по созданию регрессионного дерева. Как и в случае с регрессионным анализом, для построения дерева классификации мы применяем рекурсивное бинарное разбиение. Однако при классификации объектов бинарные разбиения невозможно выполнять на основе RSS. Естественной альтернативой RSS является *частота ошибок классификации*. Поскольку мы планируем относить то или иное наблюдение в некоторой области к *наиболее часто встречающемуся классу* обучающих наблюдений в этой области, то частота ошибок классификации представляет собой просто долю обучающих наблюдений в соответствующей области, которые не принадлежат к наиболее распространенному классу:

частота ошибок классификации

$$E = 1 - \max_k(\hat{p}_{mk}). \tag{8.5}$$

Здесь $\hat{p}_{mk}$ представляет собой долю обучающих наблюдений в m–й области, принадлежащих к классу k. Оказывается, однако, что ошибка классификации недостаточно чувствительна для построения дерева, и поэтому на практике предпочтительными являются два других критерия.

индекс Джини

Индекс Джини определяют как

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{8.6}$$

что является мерой общей дисперсии во всех K классах. Нетрудно увидеть, что индекс Джини принимает очень низкое значение, когда все $\hat{p}_{mk}$ близки к нулю или единице. По этой причине индекс Джини называют мерой *чистоты* узла[14]: низкое его значение указывает на то, что соответствующий узел преимущественно содержит наблюдения какого–то одного класса.

перекрестная энтропия

Альтернативой индексу Джини является коэффициент *перекрестной энтропии*[15], который рассчитывают как

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}. \tag{8.7}$$

Поскольку $0 \leq \hat{p}_{mk} \leq 1$, то $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$. Можно показать, что коэффициент перекрестной энтропии принимает значение, близкое к нулю, когда все $\hat{p}_{mk}$ близки к нулю или единице. Следовательно, как и индекс Джини, коэффициент перекрестной энтропии будет низким в случае «чистого» m–го узла. Более того, оказывается, что значения индекса Джини и коэффициента перекрестной энтропии довольно похожи.

При построении дерева классификации для оценки качества того или иного разбиения используют либо индекс Джини, либо коэффициент перекрестной энтропии, поскольку оба критерия более чувствительны к чистоте узла, чем частота ошибок классификации. Любой из этих критериев можно было бы использовать и при *обрезке* ветвей дерева, однако если конечной целью является точность предсказаний, то предпочтение следует отдать частоте ошибок классификации.

На рис. 8.6 показан пример для набора данных `Heart`. Эти данные содержат бинарную переменную `HD` для 303 пациентов, испытывающих боль в груди. Значение отклика `Yes` указывает на наличие болезни сердца, выявленное в ходе ангиографического теста, а `No` — на отсутствие болезни. Имеется 13 предикторов, включая `Age` (возраст), `Sex` (пол), `Chol` (уровень холестерина), а также ряд других результатов измерений функций сердца и легких. Перекрестная проверка приводит к дереву с шестью конечными узлами.

До сих пор в наших обсуждениях мы предполагали, что предикторы являются количественными переменными. Однако деревья решений можно построить даже в присутствии качественных предикторов. Например, некоторые переменные в наборе данных `Heart`, такие как `Sex`, `Thal` (таллиевый стресс–тест) и `ChestPain` (боль в груди), являются качественными. Следовательно, разбиение на основе одной из этих переменных сводится

[14] В оригинале используется термин «node purity». — *Прим. пер.*
[15] В оригинале используется термин «cross–entropy». — *Прим. пер.*

к направлению одних значений качественной переменной по одной ветви дерева, а других значений — по другой ветви. На рис. 8.6 некоторые внутренние узлы соответствуют разбиениям качественных переменных. Например, первый внутренний узел соответствует разбиению переменной `Thal`. Текстовая метка `Thal:a` означает, что ветвь, уходящая из этого узла влево, включает наблюдения с первым значением переменной `Thal` (нормальное состояние), а ветвь, уходящая вправо, содержит все остальные наблюдения (хронические и обратимые состояния). Метка `ChestPain:bc`, расположенная слева двумя узлами ниже, показывает, что уходящая влево ветвь содержит наблюдения со вторым и третьим значениями переменной `ChestPain` (возможные значения этой переменной включают типичную ангину, атипичную ангину, не связанную с ангиной боль, а также боли неопределенного происхождения).

Рисунок 8.6 имеет удивительное свойство: некоторые разбиения образуют конечные узлы с *одинаковыми предсказанными значениями*. Рассмотрим, например, разбиение `RestECG<1`, расположенное в нижней правой части необрезанного дерева. Вне зависимости от значения `RestECG` эти наблюдения отнесены к классу `Yes`. Зачем же тогда вообще выполнять это разбиение? Это разбиение выполнено в связи с тем, что оно приводит к повышенной *чистоте узла*. Так, все 9 наблюдений из правого конечного узла принадлежат к классу `Yes`, тогда как это же значение отклика имеют только 7 из 11 наблюдений из левого конечного узла. Почему чистота узла имеет значение? Представьте, что у нас есть некоторый набор контрольных наблюдений, принадлежащих к правому конечному узлу. Тогда мы можем быть почти полностью уверенными в том, что отклик принимает значение `Yes`. Однако если некоторое контрольное наблюдение принадлежит к левому конечному узлу, то его отклик, вероятно, тоже равен `Yes`, хотя мы уверены в этом намного меньше. Несмотря на то что разбиение `RestECG<1` не приводит к возникновению ошибки классификации, оно повышает индекс Джини и перекрестную энтропию, которые намного более чувствительны к чистоте узла.

8.1.3 Сравнение деревьев с линейными моделями

Деревья регрессии и классификации существенно отличаются от более традиционных методов регрессии и классификации, представленных в главах 3 и 4. В частности, линейная регрессия предполагает модель вида

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j, \tag{8.8}$$

тогда как регрессионные деревья предполагают модель вида

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)}, \tag{8.9}$$

где $R_1, \ldots, R_M$ представляют собой некоторые непересекающиеся области пространства предикторов (например, как на рис. 8.3.).

Какая из этих моделей лучше? Это зависит от решаемой проблемы. Если связь между предикторами и откликом хорошо аппроксимируется

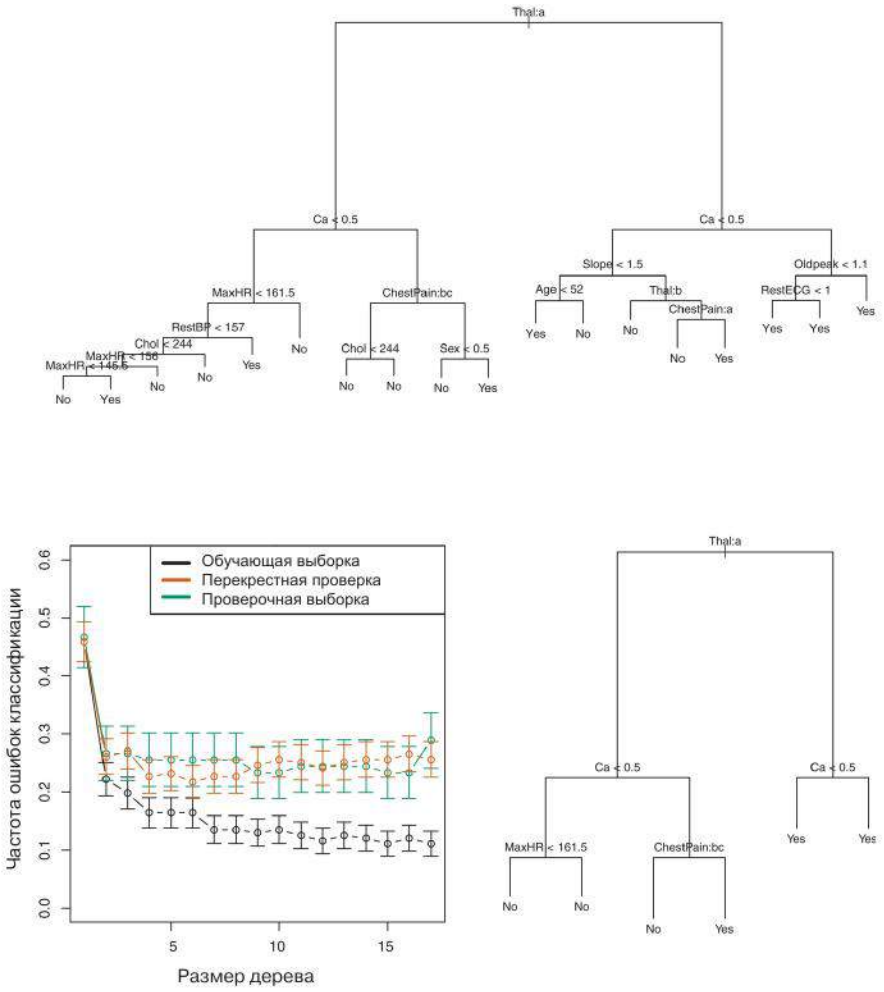

РИСУНОК 8.6. *Данные* `Heart`. Вверху: *необрезанное дерево*. Слева внизу: *оценки ошибок на обучающих данных, на контрольных данных, а также ошибок перекрестной проверки, показанные в зависимости от числа конечных узлов у обрезанного дерева.* Справа внизу: *обрезанное дерево, соответствующее минимальной ошибке перекрестной проверки*

уравнением (8.8), тогда линейная модель, вероятно, сработает хорошо и превзойдет по качеству регрессионное дерево, которое плохо передает эту линейную структуру. Если же между откликом и предикторами имеет место выраженная нелинейная и сложная зависимость, как в (8.9), тогда деревья решений могут превзойти классические методы. Характерный пример приведен на рис. 8.7. Относительное качество классических методов и подходов, основанных на деревьях решений, можно сравнить, оценивая ошибку на независимых данных с помощью перекрестной проверки или метода проверочной выборки (глава 5).

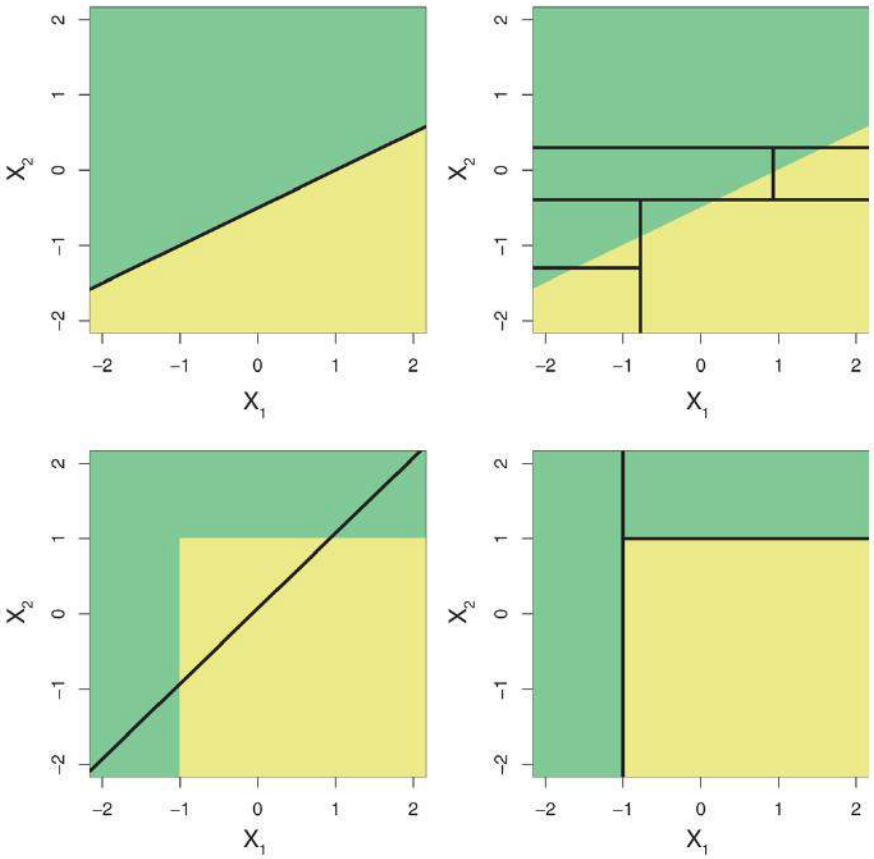

РИСУНОК 8.7. Верхний ряд: *пример классификации по двум переменным, в котором истинная решающая граница является линейной (показана заштрихованными областями). Классический метод, предполагающий линейную границу (слева), превзойдет дерево решений, которое выполняет разбиения параллельно координатным осям (справа).* Нижний ряд: *здесь истинная решающая граница является нелинейной. Линейная модель (слева) в этом случае не способна передать истинную границу, тогда как дерево решений (справа) справляется с этим успешно*

Конечно, помимо ошибки на контрольной выборке, при выборе метода статистического обучения определенную роль могут сыграть и другие соображения; например, в некоторых случаях предсказание с использованием дерева может оказаться предпочтительным в связи с легкостью интерпретации и визуализации модели.

8.1.4 Преимущества и недостатки деревьев решений

Деревья классификации и регрессии имеют ряд преимуществ, по сравнению с более традиционными методами, рассмотренными в главах 3 и 4:

- ▲ Деревья очень легко объяснить другим людям. Более того, их даже легче объяснить, чем линейную регрессию!
- ▲ Бытует убеждение, что деревья решений в большей степени отражают процесс принятия решений людьми, чем рассмотренные в предыдущих главах другие методы регрессии и классификации.
- ▲ Деревья можно показать графически, и их легко может интерпретировать даже неспециалист (особенно если деревья небольшие).
- ▲ Деревья легко справляются с качественными предикторами, без необходимости создания индикаторных переменных.
- ◆ К сожалению, обычно деревья не настолько точны в прогнозах, как другие рассмотренные в этой книге методы классификации и регрессии.

Тем не менее, обобщив большое количество деревьев при помощи таких методов, как *бэггинг*, *случайные леса* и *бустинг*, качество предсказаний деревьев можно существенно улучшить. Эти концепции представлены в следующем разделе.

8.2 Бэггинг, случайные леса, бустинг

Бэггинг, случайные леса и бустинг используют деревья решений в качестве строительных блоков для создания более мощных предсказательных моделей.

8.2.1 Бэггинг

Описанный в главе 5 метод бутстрепа является чрезвычайно сильной идеей. Он применяется во многих ситуациях, когда вычисление стандартного отклонения некоторой статистики затруднено или вообще невозможно. Здесь мы увидим, что бутстреп можно применять в совершенно другом контексте, а именно для улучшения качества предсказаний статистических моделей (в частности, деревьев решений).

Описанные в разделе 8.1 деревья решений страдают от *высокой дисперсии*. Это означает, что если мы случайным образом разобьем обучающие данные на две части и построим дерево решений на основе каждой из них, то полученные результаты могут оказаться довольно разными. В то же время метод с *низкой дисперсией* при многократном применении к разным наборам данных будет давать похожие результаты (так, при умеренно большом отношении n к p линейная регрессия обычно имеет низкую дисперсию). *Бутстреп–агрегирование*, или *бэггинг*[16], является процедурой общего назначения, позволяющей снизить дисперсию статистической модели; мы представляем этот подход здесь, поскольку он особенно полезен и часто применяется в контексте деревьев решений.

бэггинг

[16] В оригинале используются термины «bootstrap aggregation» и «bagging» соответственно. — *Прим. пер.*

Вспомните, что для некоторой совокупности из n наблюдений $Z_1, \ldots, Z_n$, каждое из которых имеет дисперсию σ^2, дисперсия среднего значения $\bar{Z}$ рассчитывается как σ^2/n. Другими словами, *усреднение нескольких наблюдений снижает дисперсию*. Следовательно, естественным способом снижения дисперсии и, как результат, повышения точности предсказаний того или иного метода статистического обучения являются взятие большого количества обучающих выборок из генеральной совокупности, построение предсказательной модели по каждой обучающей выборке и усреднение полученных предсказаний. Иными словами, мы могли бы вычислить $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$ на основе B отдельных обучающих выборок, а затем усреднить их для получения единой статистической модели с низкой дисперсией:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x).$$

Конечно, этот подход неприменим на практике, поскольку обычно у нас нет доступа к большому количеству обучающих выборок. Вместо этого мы можем применить бутстреп, выполнив многократное изъятие выборок из (одного) обучающего набора данных. Так мы формируем B отдельных обучающих бутстреп–выборок. Затем мы обучаем наш метод на b–й бутстреп–выборке для нахождения $\hat{f}^{*b}(x)$ и, наконец, усредняем все предсказания для получения

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

Эта процедура и называется бэггингом.

Хотя бэггинг может улучшить предсказания многих регрессионных методов, он особенно полезен для деревьев решений. Для применения бэггинга к регрессионным деревьям мы просто строим B регрессионных деревьев на основе B обучающих бутстреп–выборок, а затем усредняем получающиеся предсказания. Такие деревья строятся глубокими и не обрезаются. Следовательно, каждое отдельное дерево имеет высокую дисперсию, но низкое смещение. Усреднение этих B деревьев снижает дисперсию. Было показано, что в результате объединения сотен или даже тысяч деревьев в единую процедуру бэггинг дает впечатляющие улучшения точности предсказаний.

До сих пор мы описывали процедуру бэггинга в контексте регрессии, т. е. для предсказания количественного отклика Y. Как можно расширить бэггинг на задачу классификации, где Y является качественной переменной? В такой ситуации есть несколько возможных подходов, однако самый простой из них заключается в следующем. Для некоторого контрольного наблюдения мы учитываем класс, предсказанный каждым из B деревьев, а затем принимаем решение на основе *большинства голосов*[17]: окончательное предсказание представляет собой наиболее часто встречающийся класс среди B предсказаний.

большинство голосов

[17] В оригинале используется термин «majority vote». — *Прим. пер.*

На рис. 8.8 показаны результаты применения бэггинга к деревьям, построенным по данным Heart. Ошибка предсказаний на контрольной выборке показана как функция от B — количества деревьев, построенных на основе обучающих бутстреп–выборок. Как видим, ошибка на контрольной выборке, полученная в результате применения бэггинга, несколько ниже ошибки, полученной при построении одного дерева. Количество деревьев B не является критическим параметром при использовании бэггинга: очень большое значение B не приведет к переобучению. На практике мы используем значение B, достаточно большое для стабилизации ошибки. Значение $B = 100$ является достаточным для достижения хорошего качества предсказаний в этом примере.

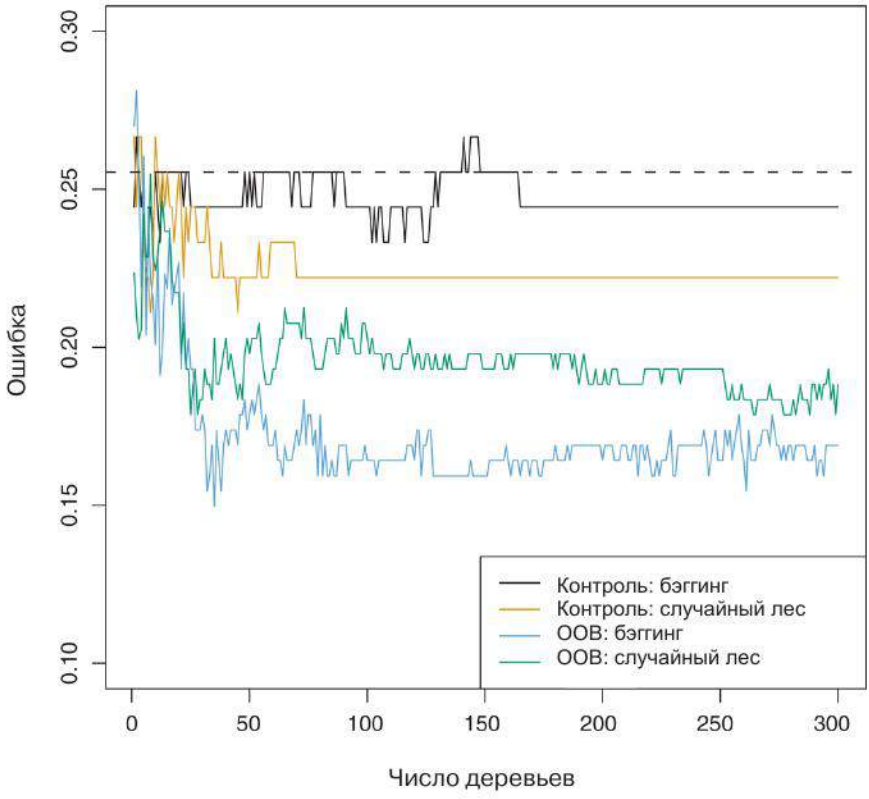

РИСУНОК 8.8. *Результаты применения методов бэггинга и случайного леса к данным* Heart. *Ошибка на контрольных данных (черная и оранжевая линии) показана как функция от B — количества обучающих бутстреп–выборок. Метод случайного леса был применен с использованием $m = \sqrt{p}$. Прерывистая линия показывает ошибку на контрольной выборке, полученную в результате построения только одного дерева. Зеленая и голубая линии показывают ошибку на оставшихся данных (OOB), которая в данном случае является намного более низкой*

Оценивание ошибки по оставшимся данным

Оказывается, что есть очень простой способ оценить ошибку на контрольных данных для бэггинг–модели, который не требует выполнения перекрестной проверки или применения метода проверочной выборки. Вспомните, что ключевая идея бэггинга состоит в многократном построении деревьев по наблюдениям из бутстреп–выборок. Можно показать, что в среднем каждое полученное таким образом дерево использует около двух третей всех наблюдений[18]. Остальная треть наблюдений не используется в подгонке конкретного дерева, и такие наблюдения называют *оставшимися*[19]. Мы можем предсказать отклик для i–го наблюдения с помощью каждого дерева, для которого это наблюдение оказалось оставшимся. Это даст примерно $B/3$ предсказаний для i–го наблюдения. Для получения итогового прогноза для i–го наблюдения мы можем усреднить эти отдельные предсказания отклика (если решается регрессионная задача) или принять решение на основе большинства голосов (если решается задача классификации). Это приведет к предсказанию i–го наблюдения по оставшимся данным. Подобные предсказания по оставшимся данным можно сделать для каждого из n наблюдений, а затем рассчитать общую MSE на оставшихся данных (для задач регрессии) или частоту ошибок (для задач классификации). Для полученной в результате бэггинга модели такая ошибка на оставшихся данных[20] является состоятельной оценкой ошибки на контрольной выборке, поскольку отклик для каждого наблюдения предсказывается на основе только тех деревьев, которые были построены без участия соответствующего наблюдения. На рис. 8.8 приведена ошибка на оставшихся наблюдениях для набора данных `Heart`. Можно показать, что при достаточно большом B ошибка на оставшихся данных фактически эквивалентна ошибке, которую получают в результате перекрестной проверки по отдельным наблюдениям. Нахождение ошибки на контрольной выборке по методу оставшихся данных является особенно удобным при выполнении бэггинга на больших наборах данных, для которых использование перекрестной проверки было бы затруднительным с вычислительной точки зрения.

Показатели важности переменных

Как мы уже отмечали, бэггинг, в отличие от использования одного дерева, обычно приводит к повышенной точности предсказаний. К сожалению, однако, интерпретация получаемой таким образом модели может вызвать затруднения. Вспомните, что одним из преимуществ деревьев решений является привлекательная и легко интерпретируемая итоговая диаграмма вроде той, которая показана на рис. 8.1. При выполнении же бэггинга с большим количеством деревьев получающуюся статистическую модель невозможно представить в виде одного дерева, и становится не понятно, какие из переменных в этой модели являются наиболее важными. Таким образом, бэггинг повышает точность предсказаний за счет снижения интерпретируемости.

[18] См. также упражнение 2 к главе 5.

[19] В оригинале используется термин «out–of–bag observations» (OOB), что дословно можно перевести как «наблюдения, не попавшие в сумку». — *Прим. пер.*

[20] В оригинале используется термин «out–of–bag error». — *Прим. пер.*

Хотя набор полученных в результате бэггинга деревьев гораздо сложнее интерпретировать, чем одно дерево, мы все же можем получить обобщенный показатель важности каждого предиктора с использованием RSS (в случае с регрессионными задачами) или индекса Джини (при решении задач классификации). При выполнении бэггинга регрессионных деревьев мы можем учесть общую величину, на которую RSS из (8.1) снижается после разбиения по некоторому предиктору, и усреднить эту величину по всем B деревьям. Большое значение указывает на важный предиктор. Аналогичным образом при выполнении бэггинга деревьев классификации мы можем получить общую величину, на которую индекс Джини (8.6) снижается при разбиении по некоторому предиктору, и усреднить эту величину по всем B деревьям.

важность переменной

Значения важности переменных из набора данных `Heart` представлены на рис. 8.9. Мы видим среднее снижение индекса Джини для каждой переменной относительно максимального из этих значений. Наибольшее снижение индекса Джини наблюдается у переменных `Thal`, `Ca` и `ChestPain`.

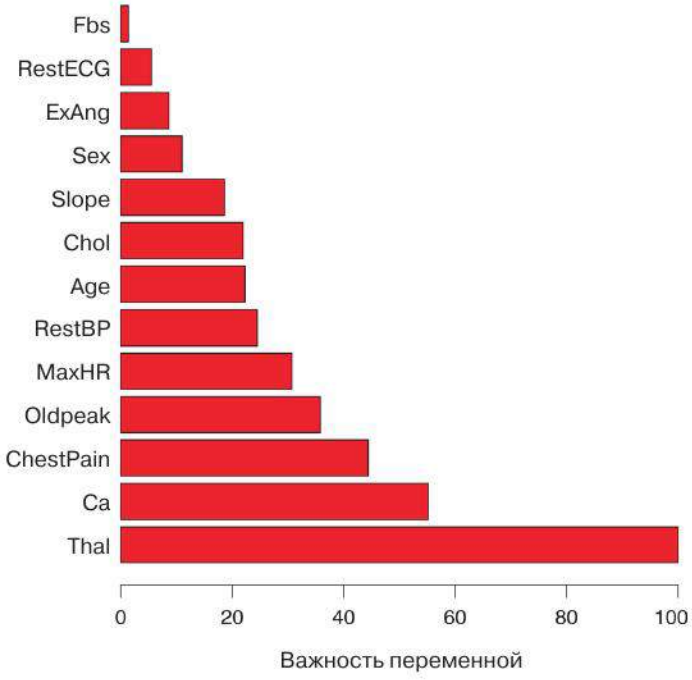

РИСУНОК 8.9. *График важности переменных из набора данных* `Heart`. *Важность переменных вычисляется как среднее снижение индекса Джини и выражается в виде удельной величины, которая рассчитывается относительно максимального значения*

8.2.2 Случайные леса

случайный лес

Метод *случайного леса*[21] представляет собой дальнейшее небольшое улучшение бэггинга деревьев решений, которое заключается в *устранении корреляции* между деревьями. Как и в случае с бэггингом, мы строим несколько деревьев решений по обучающим бутстреп–выборкам. Однако в ходе построения этих деревьев перед выполнением каждого разбиения *случайным образом выбирают только m из p подлежащих рассмотрению предикторов*. Соответствующее разбиение разрешается выполнять только по одному из этих m предикторов. Для каждого разбиения создается новый набор из m предикторов, и обычно мы выбираем $m \approx \sqrt{p}$, т. е. число предикторов, рассматриваемых при каждом разбиении, примерно равно квадратному корню из общего числа предикторов (4 из 13 в случае с данными `Heart`).

Другими словами, в ходе построения случайного леса при выполнении каждого разбиения алгоритму *даже не позволяется рассматривать* большинство из имеющихся предикторов. Это может звучать очень странно, однако данная идея имеет глубокое теоретическое обоснование. Представьте, что в данных есть один очень сильный предиктор и несколько других, умеренно коррелирующих с откликом предикторов. Тогда большинство или даже все полученные в результате бэггинга деревья в своем основании будут использовать этот сильный предиктор. Следовательно, все такие деревья будут очень похожи друг на друга. Получаемые же на их основе предсказания будут сильно коррелировать. К сожалению, усреднение большого числа коррелирующих величин не приводит к тому же значительному снижению дисперсии, которое происходит при усреднении некоррелирующих величин. Это значит, в частности, что в такой ситуации бэггинг не приведет к существенному снижению дисперсии (по сравнению с одним деревом).

Случайные леса решают эту проблему, заставляя алгоритм рассматривать перед каждым разбиением только определенное подмножество предикторов. Следовательно, в среднем $(p - m)/p$ разбиений полностью проигнорируют сильный предиктор, благодаря чему другие предикторы получат свой шанс. Мы можем думать об этой процедуре как об *устранении корреляции* между деревьями, благодаря которому усреднение по итоговым деревьям оказывается менее изменчивым и, следовательно, более надежным.

Основное различие между бэггингом и случайными лесами заключается в выборе размера подмножества предикторов m. Так, если случайный лес строится с использованием $m = p$, то вся процедура сводится к простому бэггингу. В сравнении с бэггингом применение случайного леса с $m = \sqrt{p}$ к данным `Heart` приводит к снижению как ошибки на контрольной выборке, так и ошибки на оставшихся данных (рис. 8.8).

Выбор малого значения m при построении случайного леса обычно будет полезным при наличии большого числа коррелирующих предикторов. Мы применили метод случайного леса к биологическим данным большой размерности, содержащим результаты измерения уровня экспрессии 4718 генов в образцах ткани 349 пациентов. У человека есть примерно 20 000 генов, и отдельные гены проявляют разные уровни активности, или экс-

[21] В оригинале используется термин «random forest». — *Прим. пер.*

прессии, в разных клетках, тканях и биологических условиях. В этом наборе данных каждый образец имеет качественную метку с 15 возможными значениями: нормальная ткань или один из 14 типов рака. Наша цель заключалась в применении метода случайного леса для предсказания типа рака на основе 500 генов, чьи уровни экспрессии характеризуются наибольшей дисперсией в обучающей выборке.

Мы случайным образом разбили наблюдения на обучающую и контрольную выборки и применили метод случайного леса к обучающим данным с тремя разными значениями размера подмножества предикторов m. Результаты представлены на рис. 8.10. Частота ошибки одного дерева составляет 45.7% при фоновой частоте ошибки в 75.4%[22]. Мы видим, что 400 деревьев достаточно для достижения хорошего качества предсказаний и что использование $m = \sqrt{p}$ в данном случае дало незначительное улучшение ошибки на контрольных данных, по сравнению с бэггингом ($m = p$). Как и в случае с бэггингом, случайные леса не приводят к переобучению при увеличении B, и поэтому на практике мы выбираем значение B, достаточно высокое для стабилизации частоты ошибок.

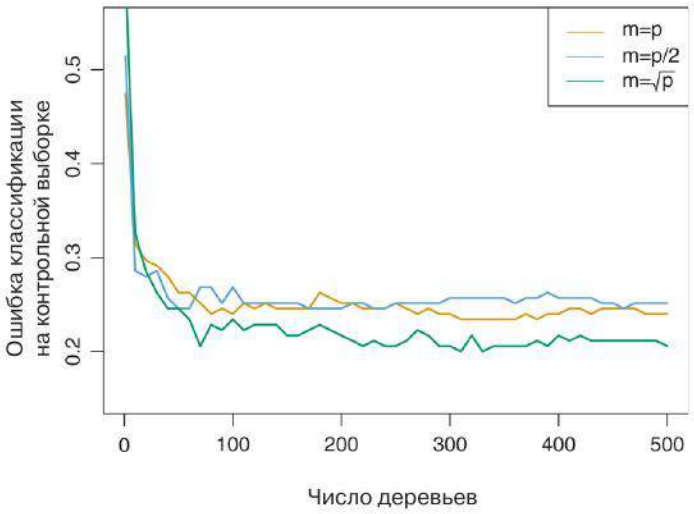

РИСУНОК 8.10. *Результаты применения метода случайного леса к данным с 15 классами и 500 предикторами. Ошибка на контрольных данных показана как функция от числа деревьев. Каждая линия соответствует определенному значению m — числу предикторов, доступных для разбиения данных при формировании каждого внутреннего узла дерева. Случайные леса ($m < p$) привели к некоторому улучшению, по сравнению с бэггингом ($m = p$). Частота ошибки у одного дерева классификации составляет 45.7%*

[22] Фоновая (нулевая) частота ошибки возникает в результате простого отнесения каждого наблюдения к наиболее распространенному классу, который в данном случае представлен нормальным классом.

8.2.3 Бустинг

бустинг

Теперь мы перейдем к обсуждению *бустинга* — еще одного метода для улучшения предсказаний, получаемых на основе одного дерева решений. Подобно бэггингу, бустинг является общим подходом, который можно применять ко многим статистическим методам регрессии и классификации. Здесь мы ограничимся обсуждением бустинга в контексте деревьев решений.

Вспомните, что бэггинг подразумевает создание большого числа копий исходного набора данных путем применения бутстрепа, построение отдельного дерева решений для каждой из этих копий и последующее объединение всех деревьев для создания одной предсказательной модели. Важно, что каждое дерево строится по определенной бутстреп–выборке независимо от других деревьев. Бустинг работает похожим образом, однако деревья строятся *последовательно*: каждое дерево выращивается с использованием информации по ранее выращенным деревьям. Бутстреп–выборки в ходе реализации бустинга не создаются — вместо этого каждое дерево строится по определенным образом модифицированному исходному набору данных.

Рассмотрим для начала регрессионную задачу. Подобно бэггингу, бустинг состоит в объединении большого числа деревьев $\hat{f}^1, \ldots, \hat{f}^B$. Его описание приведено в алгоритме 8.2.

Алгоритм 8.2 Бустинг регрессионных деревьев

1. Присвойте $\hat{f}(x) = 0$ и $r_i = y_i$ для всех i в обучающей выборке.
2. Для $b = 1, 2, \ldots, B$ повторите:
   (a) постройте дерево $\hat{f}^b$ с d внутренними узлами (или $d + 1$ конечными узлами) по обучающим данным (X, r);
   (b) обновите $\hat{f}$, добавив обрезанную версию нового дерева:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x); \tag{8.10}$$
   (c) обновите остатки
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Итоговая модель после бустинга представляет собой
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

В чем состоит идея этой процедуры? В отличие от построения одного большого дерева решений, которое сводится к *тесной аппроксимации данных* и потенциально может привести к переобучению, бустинг выполняет *медленное обучение*. Для текущей модели мы строим дерево решений

по ее остаткам. Другими словами, для построения дерева в качестве отклика мы используем текущие остатки, а не переменную Y. Затем мы добавляем это новое регрессионное дерево к ранее подогнанной модели для обновления остатков. Каждое из этих деревьев может быть довольно небольшим, всего лишь с несколькими конечными узлами, количество которых задается в алгоритме 8.2 параметром d. Благодаря построению неглубоких деревьев по остаткам мы медленно улучшаем $\hat{f}$ в областях, где эта функция работает не очень хорошо. Параметр сжатия λ еще сильнее замедляет этот процесс, позволяя создавать деревья более сложной формы для «атаки» остатков. Обычно *медленно обучаемые* статистические методы обеспечивают более высокое качество предсказаний. Заметьте, что в случае с бустингом (в отличие от бэггинга) построение каждого дерева в значительной мере зависит от ранее построенных деревьев.

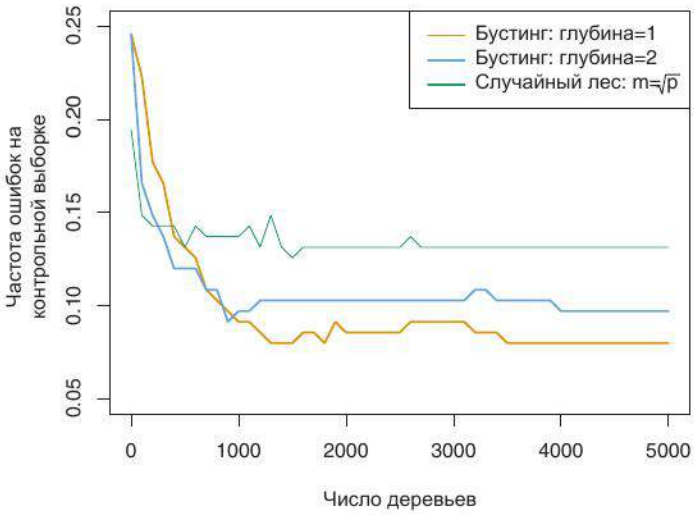

РИСУНОК 8.11. *Результаты применения бустинга и метода случайного леса к данным по уровню экспрессии генов для распознавания 15 классов (раковые и нормальные ткани). Ошибка на контрольных данных показана в зависимости от числа деревьев. У двух моделей, к которым был применен бустинг, $\lambda = 0.01$. Деревья с одним узлом обеспечивают несколько более высокую точность предсказаний, чем деревья с двумя узлами, но при этом оба типа деревьев превосходят случайный лес (правда, стандартные ошибки составляют около 0.02, делая эти различия незначительными). Частота ошибок на контрольных данных для одного дерева составляет 24%*

Мы только что описали процесс бустинга регрессионных деревьев. Бустинг деревьев классификации выполняется похожим, хотя и несколько более сложным образом, и соответствующие детали мы здесь не приводим.

Бустинг имеет три гиперпараметра:

1. Число деревьев B. В отличие от бэггинга и метода случайного леса, бустинг может приводить к переобучению при чрезмерно большом B,

хотя подобное переобучение если и случается, то возникает медленно. Для выбора B мы используем перекрестную проверку.

2. Параметр сжатия λ — некоторое небольшое положительное число. Этот параметр контролирует скорость, с которой происходит обучение модели при реализации бустинга. Типичные значения варьируют от 0.01 до 0.001, и правильный их выбор зависит от решаемой проблемы. Для достижения хорошего качества предсказаний очень низкие значения λ требуют очень большого значения B.

3. Число внутренних узлов d в каждом дереве, которое контролирует сложность получаемого в результате бустинга ансамбля моделей. Часто хорошо работает $d = 1$, когда дерево просто представляет собой «пень»[23], т. е. содержит только один внутренний узел. В таком случае получаемый в результате бустинга ансамбль представляет собой аддитивную модель, поскольку каждый ее член представлен только одной переменной. Говоря более общим языком, параметр d отражает *глубину взаимодействий*[24] и контролирует порядок взаимодействий между предикторами в итоговой модели, поскольку d внутренних узлов могут соответствовать не более чем d переменным.

пень

глубина взаимодействий

На рис. 8.11 мы применили бустинг к данным по уровню экспрессии генов в тканях 15 типов с целью разработки классификатора, способного отличить нормальную ткань от 14 типов раковых тканей. Мы приводим частоту ошибок классификации на контрольной выборке как функцию от общего числа деревьев и глубины взаимодействий d. Как видим, деревья с одним внутренним узлом работают хорошо, если их число достаточно велико. По качеству предсказаний эта модель превосходит модель с двумя внутренними узлами; при этом обе эти модели превосходят случайный лес. Данный результат подчеркивает одно из отличий бустинга от метода случайного леса: поскольку при реализации бустинга каждое дерево строится с учетом ранее построенных деревьев, то деревьев меньшей глубины обычно оказывается достаточно. Использование деревьев меньшей глубины может также помочь с интерпретацией (например, использование «пней» приводит к аддитивной модели).

8.3 Лабораторная работа: деревья решений

8.3.1 Построение деревьев классификации

Для создания деревьев классификации и регрессии служит библиотека `tree`.

```
> library(tree)
```

Сначала мы применим деревья классификации для анализа данных `Carseats`. В этом наборе данных `Sales` представляет собой непрерывную переменную, и поэтому мы начнем с конвертирования ее в бинарную переменную. Мы воспользуемся функцией `ifelse()` для создания переменной

`ifelse()`

[23] В оригинале используется термин «stump». — *Прим. пер.*
[24] В оригинале используется термин «interaction depth». — *Прим. пер.*

High, которая принимает значение Yes, когда переменная Sales превышает 8, и No в остальных случаях.

```
> library(ISLR)
> attach(Carseats)
> High = ifelse(Sales <= 8, "No", "Yes")
```

Наконец, мы применим функцию data.frame() для добавления High к набору данных Carseats.

```
> Carseats = data.frame(Carseats, High)
```

Теперь мы воспользуемся функцией tree() для построения дерева классификации, предсказывающего High по всем переменным, за исключением Sales. Синтаксис функции tree() довольно похож на таковой у функции lm().

```
> tree.carseats = tree(High ~ . -Sales, Carseats)
```

Функция summary() выводит список переменных, использованных при создании внутренних узлов дерева, число конечных узлов, а также частоту ошибок (на обучающей выборке).

```
> summary(tree.carseats)

Classification tree:
tree(formula = High ~ . - Sales, data = Carseats)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"  "CompPrice"
[5] "Population " "Advertising" "Age"     "US"
Number of terminal nodes: 27
Residual mean deviance: 0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

Как видим, частота ошибок на обучающих данных составляет 9%. Для деревьев классификации приведенный выше показатель аномальности («deviance») рассчитывается как

$$-2\sum_m\sum_k n_{mk}\log\hat{p}_{mk},$$

где n_{mk} — это число наблюдений в конечном узле m, принадлежащих к классу k. Небольшое значение показателя аномальности указывает на то, что дерево хорошо описывает (обучающие) данные. Приведенная *остаточная средняя аномальность* («residual mean deviance») — это просто аномальность, деленная на $n - |T_0|$, что в данном случае составляет $400 - 27 = 373$.

Одно из самых больших преимуществ деревьев заключается в том, что их можно изобразить графически. Для изображения структуры дерева мы применяем функцию plot(), а для добавления меток узлов — функцию text(). Аргумент pretty = 0 указывает R, что для любых качественных предикторов необходимо привести названия соответствующих категорий, а не просто буквенные обозначения этих категорий.

```
> plot(tree.carseats)
> text(tree.carseats, pretty = 0)
```

Похоже, что наиболее важным предиктором объема продаж является качество места, в котором расположен выставочный стеллаж, поскольку первая ветвь разделяет хорошие места (Good) и все остальные места (Bad и Medium).

Если мы просто наберем и введем имя объекта, в котором хранится построенное дерево, то R выведет описания каждой ветви этого дерева. R показывает критерий разбиения данных (например, Price < 92.5), число наблюдений, приходящихся на соответствующую ветвь, показатель аномальности, предсказанный класс всей ветви (Yes или No), а также долю приходящихся на эту ветвь наблюдений со значениями Yes и No. Ветви, которые ведут к конечным узлам, отмечены звездочками.

```
> tree.carseats
node), split, n, deviance, yval, (yprob)
      * denotes terminal node
 1) root 400 541.5 No ( 0.590 0.410 )
   2) ShelveLoc: Bad, Medium 315 390.6 No ( 0.689 0.311 )
     4) Price < 92.5 46 56.53 Yes ( 0.304 0.696 )
       8) Income < 57 10 12.22 No ( 0.700 0.300 )
```

Для получения корректной оценки качества предсказаний этого дерева классификации мы должны оценить ошибку на контрольных данных, а не просто вычислить ошибку на обучающих данных. Мы разделяем наблюдения на обучающую и контрольную выборки, строим дерево по обучающей выборке, а затем оцениваем его качество на контрольной выборке. Для этого можно использовать функцию predict(). В случае с деревом классификации аргумент type = "class" говорит R о том, что необходимо возвращать метку предсказанного класса. Этот подход приводит к правильной классификации примерно 71.5% наблюдений из контрольной выборки.

```
> set.seed(2)
> train = sample(1:nrow(Carseats), 200)
> Carseats.test = Carseats[-train, ]
> High.test = High[-train]
> tree.carseats=tree(High ~ . -Sales, Carseats, subset=train)
> tree.pred=predict(tree.carseats, Carseats.test, type="class")
> table(tree.pred, High.test)
          High.test
tree.pred  No Yes
      No   86  27
      Yes  30  57
> (86 + 57) / 200
[1] 0.715
```

Далее мы рассмотрим, может ли обрезка ветвей дерева улучшить эти результаты. Функция cv.tree() выполняет перекрестную проверку для нахождения оптимального уровня сложности дерева; последовательность рассматриваемых деревьев выбирается путем обрезки с учетом штрафа за сложность. Мы применяем аргумент FUN = prune.misclass, чтобы указать, что процесс обрезки и перекрестной проверки должен происходить с учетом частоты ошибок классификации, а не показателя аномальности,

который используется функцией cv.tree() по умолчанию. В качестве результатов функция cv.tree() выводит число конечных узлов у каждого из рассмотренных деревьев (size), а также сопутствующую частоту ошибок и значение использованного штрафа за сложность k, который соответствует параметру α в (8.4).

```
> set.seed(3)
> cv.carseats = cv.tree(tree.carseats, FUN = prune.misclass)
> names(cv.carseats)
[1] "size"   "dev"   "k"   "method"
> cv.carseats
$size
[1] 19  17  14  13  9  7  3  2  1

$dev
[1] 55  55  53  52  50  56  69  65  80

$k
[1] -Inf 0.0000000 0.6666667 1.0000000 1.7500000
[6]  2.0000000 4.2500000 5.0000000 23.0000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"    "tree.sequence"
```

Заметьте, что, несмотря на свое название, dev в данном случае соответствует частоте ошибок, полученной в ходе перекрестной проверки. Согласно результатам перекрестной проверки, дерево с 9 конечными узлами приводит к наименьшей частоте ошибок (50). Изобразим частоту ошибок как функцию от size и k:

```
> par(mfrow = c(1, 2))
> plot(cv.carseats$size, cv.carseats$dev, type = "b")
> plot(cv.carseats$k, cv.carseats$dev, type = "b")
```

Теперь мы применим функцию prune.misclass() для обрезки исходного дерева и получения дерева с девятью конечными узлами.

```
> prune.carseats = prune.misclass(tree.carseats, best = 9)
> plot(prune.carseats); text(prune.carseats, pretty = 0)
```

Какова точность предсказаний обрезанного дерева на контрольной выборке? Мы снова применяем функцию predict().

```
> tree.pred=predict(prune.carseats, Carseats.test, type="class")
> table(tree.pred, High.test)
          High.test
tree.pred  No Yes
       No 94  24
      Yes 22  60
> (94 + 60) / 200
[1] 0.77
```

Теперь правильно предсказаны 77% контрольных наблюдений, т. е. процесс обрезки не только привел к получению более интерпретируемого дерева, но и улучшил точность классификации.

Увеличив значение `best`, мы получим более глубокое обрезанное дерево с более низкой точностью предсказаний:

```
> prune.carseats = prune.misclass(tree.carseats, best = 15)
> plot(prune.carseats)
> text(prune.carseats, pretty = 0)
> tree.pred=predict(prune.carseats, Carseats.test, type="class")
> table(tree.pred, High.test)
          High.test
tree.pred  No Yes
       No 86  22
      Yes 30  62
> (86 + 62) / 200
[1] 0.74
```

8.3.2 Построение регрессионных деревьев

Здесь мы построим регрессионное дерево для набора данных `Boston`. Сначала мы создадим обучающую выборку и построим дерево на ее основе.

```
> library(MASS)
> set.seed(1)
> train = sample(1:nrow(Boston), nrow(Boston)/2)
> tree.boston = tree(medv ~ ., Boston, subset = train)
> summary(tree.boston)
Regression tree:
tree(formula = medv ~ ., data = Boston, subset = train)
Variables actually used in tree construction:
[1]  "lstat"  "rm"  "dis"
Number of terminal nodes: 8
Residual mean deviance:  12.65 = 3099 / 245

Distribution of residuals:
    Min.  1st Qu.   Median    Mean  3rd Qu.      Max.
-14.1000  -2.0420  -0.0536  0.0000   1.9600   12.6000
```

Заметьте, что, согласно выведенным функцией `summary()` результатам, в построении дерева были задействованы только три переменные. В контексте регрессионного дерева показатель аномальности — это не что иное, как сумма квадратов ошибок. Теперь мы изобразим это дерево графически.

```
> plot(tree.boston)
> text(tree.boston, pretty = 0)
```

Переменная `lstat` содержит значения доли людей с низким социоэкономическим статусом. Полученное дерево показывает, что более низкие значения `lstat` соответствуют более дорогим домам. Это дерево предсказывает, что медианная стоимость больших домов в пригородах, где проживают люди с высоким социоэкономическим статусом (`rm >= 7.437` и `lstat < 9.715`), составляет 46 400$.

Теперь мы воспользуемся функцией cv.tree(), чтобы выяснить, поможет ли обрезка дерева улучшить качество предсказаний.

```
> cv.boston = cv.tree(tree.boston)
> plot(cv.boston$size, cv.boston$dev, type = 'b')
```

В данном случае перекрестная проверка выбирает наиболее сложное дерево. Однако при желании мы могли бы обрезать это дерево следующим образом, воспользовавшись функцией prune.tree():

prune.tree()

```
> prune.boston = prune.tree(tree.boston, best = 5)
> plot(prune.boston)
> text(prune.boston, pretty = 0)
```

В соответствии с результатами перекрестной проверки для получения предсказаний мы воспользуемся необрезанным деревом.

```
> yhat = predict(tree.boston, newdata = Boston[-train, ])
> boston.test = Boston[-train, "medv"]
> plot(yhat, boston.test)
> abline(0, 1)
> mean((yhat - boston.test)^2)
[1] 25.05
```

Таким образом, MSE на контрольной выборке у этого регрессионного дерева составляет 25.05. Квадратный корень из этой MSE примерно равен 5.005: это значит, что предсказания данной модели отклоняются от истинной медианной стоимости пригородных домов в среднем на 5 005$.

8.3.3 Бэггинг и случайные леса

Здесь мы применим методы бэггинга и случайного леса к данным Boston, используя пакет randomForest для R. Результаты, которые будут получены в этом разделе, могут зависеть от версий R и пакета randomForest, установленных на вашем компьютере. Напомним, что бэггинг — это просто частный случай метода случайного леса при $m = p$. Следовательно, функцию randomForest() можно применять для выполнения как бэггинга, так и случайного леса. Мы выполняем бэггинг следующим образом:

randomForest()

```
> library(randomForest)
> set.seed(1)
> bag.boston = randomForest(medv ~ ., data = Boston,
       subset = train, mtry = 13, importance = TRUE)
> bag.boston

Call:
randomForest(formula = medv ~ ., data = Boston, mtry = 13,
     importance = TRUE,      subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 13

          Mean of squared residuals: 10.77
                    % Var explained : 86.96
```

Аргумент `mtry = 13` показывает, что для создания каждого узла дерева рассмотрению подлежат все 13 предикторов, т. е. необходимо выполнять бэггинг. Каково качество предсказаний этой бэггинг–модели на контрольных данных?

```
> yhat.bag = predict(bag.boston, newdata = Boston[-train, ])
> plot(yhat.bag, boston.test); abline(0, 1)
> mean((yhat.bag - boston.test)^2)
[1] 13.16
```

MSE на контрольных данных у этой регрессионной модели составляет 13.16 — почти половину от ошибки, полученной для оптимального обрезанного дерева. При помощи аргумента `ntree` мы могли бы изменить число деревьев, создаваемых функцией `randomForest()`:

```
> bag.boston = randomForest(medv ~ ., data = Boston,
              subset = train, mtry = 13, ntree = 25)
> yhat.bag = predict(bag.boston, newdata = Boston[-train, ])
> mean((yhat.bag - boston.test)^2)
[1] 13.31
```

Построение случайного леса выполняется аналогичным образом, однако мы применяем меньшее значение аргумента `mtry`. По умолчанию функция `randomForest()` использует $p/3$ переменных при построении случайного леса регрессионных деревьев и $\sqrt{p}$ переменных при построении случайного леса деревьев классификации. Здесь мы применим `mtry = 6`.

```
> set.seed(1)
> rf.boston = randomForest(medv ~ ., data = Boston,
      subset = train, mtry = 6, importance = TRUE)
> yhat.rf = predict(rf.boston, newdata = Boston[-train, ])
> mean((yhat.rf - boston.test)^2)
[1] 11.31
```

MSE на контрольной выборке составляет 11.31, т. е. случайный лес привел к улучшению точности предсказаний, по сравнению с бэггингом.

Применив функцию `importance()`, мы можем ознакомиться с важностью каждой переменной. `importance()`

```
> importance(rf.boston)
        %IncMSE IncNodePurity
crim     12.384       1051.54
zn        2.103         50.31
indus     8.390       1017.64
chas      2.294         56.32
nox      12.791       1107.31
rm       30.754       5917.26
age      10.334        552.27
dis      14.641       1223.93
rad       3.583         84.30
tax       8.139        435.71
ptratio  11.274        817.33
black     8.097        367.00
lstat    30.962       7713.63
```

Приведены два показателя важности. Первый из них основан на среднем снижении точности предсказаний на оставшихся («out–of–bag») данных при исключении соответствующей переменной из модели. Второй показатель является мерой среднего увеличения чистоты узла дерева («node purity») в результате разбиения данных по соответствующей переменной (см. рис. 8.9). В случае с регрессионными деревьями чистота узла выражается при помощи RSS, а в случае в деревьями классификации — при помощи показателя аномальности. Графики этих показателей важности можно построить при помощи функции `varImpPlot()`.

varImpPlot()

```
> varImpPlot(rf.boston)
```

Полученные результаты показывают, что в образующих случайный лес деревьях наиболее важными переменными являются социально–экономический статус жителей (`lstat`) и размер дома (`rm`).

8.3.4 Бустинг

Здесь мы воспользуемся пакетом `gbm` и входящей в него функцией `gbm()` для выполнения бустинга регрессионных деревьев в приложении к набору данных `Boston`. Мы выполняем команду `gbm()` с опцией `distribution = "gaussian"`, поскольку это регрессионная проблема; если бы это была бинарная классификационная задача, то мы применили бы `distribution = "bernoulli"`. Аргумент `n.trees = 5000` показывает, что мы хотим построить 5000 деревьев, а опция `interaction.depth = 4` ограничивает глубину каждого дерева.

```
> library(gbm)
> set.seed(1)
> boost.boston = gbm(medv ~ ., data = Boston[train, ],
            distribution = "gaussian", n.trees = 5000,
            interaction.depth = 4)
```

Функция `summary()` выводит график относительной важности переменных и соответствующие значения показателя важности.

```
> summary(boost.boston)
        var   rel.inf
1     lstat   45.96
2        rm   31.22
3       dis    6.81
4      crim    4.07
5       nox    2.56
6   ptratio    2.27
7     black    1.80
8       age    1.64
9       tax    1.36
10    indus    1.27
11     chas    0.80
12      rad    0.20
13       zn   0.015
```

Как видим, `lstat` и `rm` превосходят по важности все другие переменные. Мы можем также построить *графики частной зависимости* («partial

dependence plots») для этих двух переменных. Графики этого типа иллюстрируют частные эффекты тех или иных переменных на отклик после *фиксирования* эффектов других переменных на постоянном уровне. В данном случае медианная стоимость домов ожидаемо возрастает при увеличении `rm` и снижается при увеличении `lstat`.

график частной зависимости

```
> par(mfrow = c(1, 2))
> plot(boost.boston, i = "rm")
> plot(boost.boston, i = "lstat")
```

Теперь мы применим бустинг для построения модели, предсказывающей `medv` по контрольным данным:

```
> yhat.boost = predict(boost.boston, newdata = Boston[-train, ],
                       n.trees = 5000)
> mean((yhat.boost - boston.test)^2)
[1] 11.8
```

Полученная MSE на контрольной выборке составляет 11.8; это значение похоже на таковое у случайного леса и намного меньше, чем у модели, построенной методом бэггинга. При желании мы можем выполнить бустинг с другим значением параметра сжатия λ из (8.10). Принятое по умолчанию значение составляет 0.001, однако его легко изменить. Ниже мы применим $\lambda = 0.2$.

```
> boost.boston = gbm(medv ~ ., data = Boston[train, ],
                     distribution = "gaussian", n.trees = 5000,
                     interaction.depth = 4, shrinkage = 0.2,
                     verbose = FALSE)
> yhat.boost = predict(boost.boston, newdata = Boston[-train, ],
                       n.trees = 5000)
> mean((yhat.boost - boston.test)^2)
[1] 11.5
```

В данном случае использование $\lambda = 0.2$ приводит к несколько меньшей MSE на контрольной выборке, чем при $\lambda = 0.001$.

8.4 Упражнения

Теоретические

1. Нарисуйте выбранный на ваше усмотрение пример разбиения двумерного пространства признаков, которое могло бы получиться в результате применения бинарного рекурсивного разбиения. Ваш пример должен содержать не менее шести областей. Изобразите дерево решений, которое соответствует этому разбиению. Убедитесь, что вы снабдили метками все элементы вашего рисунка, включая области $R_1, R_2, \ldots$, узлы $t_1, t_2, \ldots$ и т. д.
   Подсказка: ваш результат должен напоминать рис. 8.1 и 8.2.

2. В подразделе 8.2.3 было отмечено, что бустинг на основе деревьев с одним узлом («пней») приводит к *аддитивной* модели, т. е. модели вида

$$f(X) = \sum_{j=1}^{p} f_j(X_j).$$

Объясните, почему это так. Вы можете начать с уравнения (8.12) в алгоритме 8.2.

3. Рассмотрим индекс Джини, ошибку классификации и перекрестную энтропию для простого случая классификации с двумя признаками. Постройте график, демонстрирующий как каждая из этих величин изменяется в зависимости от $\hat{p}_{m1}$. Ось X должна показывать $\hat{p}_{m1}$ для интервала значений от 0 до 1, а ось Y должна показывать значения индекса Джини, ошибку классификации и энтропию.
Подсказка: в случае с двумя классами $\hat{p}_{m1} = 1 - \hat{p}_{m2}$. *Вы могли бы нарисовать этот график от руки, однако будет намного проще сделать это в* R.

4. Этот вопрос имеет отношении к графикам, представленным на рис. 8.12.

   (a) Нарисуйте набросок дерева, соответствующего разбиению пространства предикторов, которое приведено слева на рис. 8.12. Числа внутри каждой области показывают средние значения Y.

   (b) Постройте диаграмму, подобную той, которая приведена на рис. 8.12 слева, используя дерево, показанное на том же рисунке справа. Вы должны правильно разбить пространство предикторов и указать среднее значение отклика для каждой области.

5. Представьте, что мы создаем десять бутстреп–выборок для набора данных с «красным» и «зеленым» классами. Затем мы строим дерево классификации по каждой из этих выборок и для некоторого значения X получаем 10 предсказаний вероятности P(«красный» класс | X): 0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7 и 0.75.

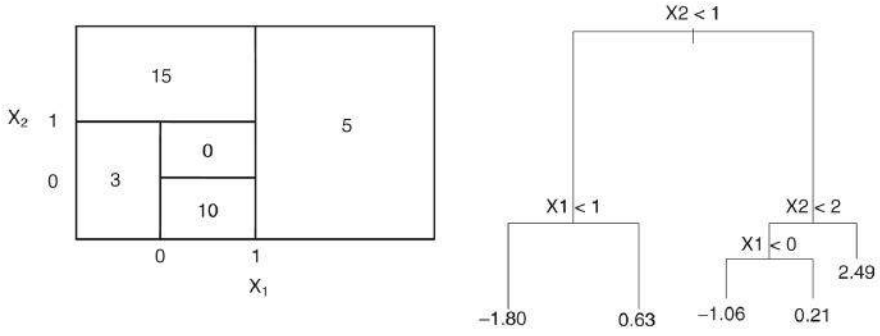

РИСУНОК 8.12. Слева: *разбиение пространства предикторов, соответствующее упражнению 4a.* Справа: *дерево, соответствующее упражнению 4b*

Есть два распространенных способа объединения этих результатов в одно число для предсказания класса. Один из них — это предсказание на основе большинства голосов, описанное в этой главе. Второй подход основан на усредненной вероятности. Какое предсказание дает в данном примере каждый из этих способов?

6. Приведите подробное объяснение алгоритма построения регрессионного дерева.

Практические

7. В лабораторной работе мы применили метод случайного леса к данным `Boston`, используя `mtry = 6`, `ntree = 25` и `ntree = 500`. Постройте кривую ошибки на контрольных данных, возникающей в результате применения случайного леса к этим данным с использованием более широкого диапазона значений `mtry` и `ntree`. Вы можете сделать свой график похожим на рис. 8.10. Опишите полученные результаты.

8. В лабораторной работе было построено дерево классификации по данным `Carseats` после преобразования `Sales` в качественную зависимую переменную. Теперь мы попробуем предсказать `Sales` при помощи регрессионного дерева и родственных ему методов, рассматривая отклик как количественную переменную.

   (a) Разбейте данные на обучающую и контрольную выборки.

   (b) Постройте регрессионное дерево по обучающей выборке. Изобразите дерево и интерпретируйте его. Чему равна полученная вами MSE на контрольной выборке?

   (c) Примените перекрестную проверку для нахождения оптимального уровня сложности дерева. Приводит ли обрезка дерева к улучшению MSE на контрольной выборке?

   (d) Проанализируйте эти данные с помощью метода бэггинга. Какую MSE на обучающей выборке вы получаете? Воспользуйтесь функцией `importance()` для определения наиболее важных переменных.

   (e) Проанализируйте эти данные с помощью метода случайного леса. Какую MSE на обучающей выборке вы получаете? Воспользуйтесь функцией `importance()` для определения наиболее важных переменных. Опишите эффект m (числа переменных, рассматриваемых при создании каждого узла дерева) на получаемую частоту ошибок.

9. Эта задача касается набора данных `OJ` из пакета `ISLR`.

   (a) Создайте обучающую выборку с 800 случайно отобранными наблюдениями и контрольную выборку, содержащую все остальные наблюдения.

(b) Постройте дерево по обучающей выборке, используя `Purchase` в качестве отклика, а все остальные переменные — в качестве предикторов. Воспользуйтесь функцией `summary()` для получения итоговых статистик для этого дерева и опишите свои результаты. Чему равна частота ошибок на обучающей выборке? Сколько у этого дерева конечных узлов?

(c) Наберите и введите имя объекта, содержащего это дерево, для получения подробной текстовой информации. Выберите один из конечных узлов и интерпретируйте выведенную информацию.

(d) Изобразите дерево графически и интерпретируйте его.

(e) Выполните предсказания отклика по контрольным данным и приведите матрицу неточностей, сравнивающую истинные метки классов контрольных наблюдений с предсказанными метками. Чему равна частота ошибок на контрольных данных?

(f) Примените функцию `cv.tree()` к обучающим данным для определения оптимальной глубины дерева.

(g) Постройте график, на котором по оси X показана глубина дерева, а по оси Y — частота ошибок классификации, полученная в ходе перекрестной проверки.

(h) Какая глубина дерева соответствует наименьшей частоте ошибок, полученной в результате перекрестной проверки?

(i) Постройте обрезанное дерево с оптимальной глубиной, найденной в ходе перекрестной проверки. Если перекрестная проверка не привела к выбору обрезанного дерева, то постройте обрезанное дерево с пятью конечными узлами.

(j) Сравните частоты ошибок на обучающих данных у обрезанного и необрезанного деревьев. Какая из них выше?

(k) Сравните частоты ошибок на контрольных данных у обрезанного и необрезанного деревьев. Какая из них выше?

10. Теперь мы применим метод бустинга для предсказания `Salary` из набора данных `Hitters`.

(a) Удалите наблюдения с отсутствующими значениями заработной платы, а затем прологарифмируйте оставшиеся значения этой переменной.

(b) Сформируйте обучающую выборку, содержащую первые 200 наблюдений, и контрольную выборку, содержащую остальные наблюдения.

(c) Выполните бустинг на обучающих данных, используя 1000 деревьев и некоторый интервал значений параметра сжатия λ. Постройте график со значениями параметра сжатия на оси X и соответствующими значениями MSE на обучающей выборке на оси Y.

(d) Постройте график со значениями параметра сжатия на оси X и соответствующими значениями MSE на контрольной выборке на оси Y.

(e) Сравните значение MSE на контрольной выборке, полученное при помощи бустинга, со значениями MSE на контрольной выборке, полученными в результате применения каких–либо двух других методов регрессии из глав 3 и 6.

(f) Какие предикторы являются наиболее важными в модели, полученной методом бустинга?

(g) Теперь примените бэггинг к обучающим данным. Чему равна MSE на контрольных данных, полученная этим методом?

11. В этой задаче используется набор данных `Caravan`.

(a) Создайте обучающую выборку, состоящую из первых 1000 наблюдений, и контрольную выборку, состоящую из всех остальных наблюдений.

(b) Постройте модель по методу бустинга, используя `Purchase` из обучающей выборки в качестве отклика, а остальные переменные — в качестве предикторов. Постройте 1000 деревьев со значением параметра сжатия, равным 0.01. Какие предикторы оказываются наиболее важными?

(c) Используйте полученную модель для предсказания отклика по контрольным данным. Примените следующее правило: если оцененная вероятность покупки страховки превышает 20%, то клиент совершит покупку. Постройте матрицу неточностей. Среди всех клиентов, для которых модель предсказывает совершение покупки, какова доля клиентов, действительно ее совершивших? Сравните полученные результаты с результатами применения к этим данным метода KNN и логистической регрессии.

12. Примените методы бустинга, бэггинга и случайного леса к любому выбранному вами набору данных. Убедитесь, что вы выполняете построение моделей по обучающим данным, а проверку их качества — по контрольным данным. Какова точность предсказаний этих методов в сравнении с простыми методами, такими как линейная или логистическая регрессия? Какой из этих методов обеспечивает наиболее точные предсказания?

Глава 9

Машины опорных векторов

В этой главе мы обсудим *машину опорных векторов* (SVM)[1] — метод классификации, который был разработан сообществом специалистов по компьютерным наукам в 1990–е годы и с тех пор приобрел широкую популярность. Было показано, что основанные на SVM модели хорошо работают в самых разных ситуациях, и часто SVM считается одним из лучших «готовых к употреблению» классификаторов.

Машина опорных векторов представляет собой обобщение простого и интуитивно понятного *классификатора с максимальным зазором*, который мы описываем в разделе 9.1. Несмотря на свою элегантность и простоту, этот классификатор, к сожалению, неприменим для большинства наборов данных, поскольку он требует, чтобы классы были разделимы линейной границей. В разделе 9.2 мы введем понятие *классификатора на опорных векторах*, который является расширением классификатора с максимальным зазором и применим для более широкого круга задач. Раздел 9.3 описывает *машину опорных векторов* — метод, который является дальнейшим расширением классификатора на опорных векторах и охватывает случаи с нелинейными границами между классами. Машины опорных векторов предназначены для сценария бинарной классификации, когда есть два класса; в разделе 9.4 мы обсудим расширения этого метода на случай с большим числом классов. В разделе 9.5 мы обсуждаем тесные сходства между машинами опорных векторов и другими статистическими методами, в частности логистической регрессией.

Часто под термином «машины опорных векторов» необоснованно обобщают классификатор с максимальным зазором, классификатор на опорных векторах и собственно машину опорных векторов. Во избежание путаницы в данной главе мы будем четко разделять эти три метода.

9.1 Классификатор с максимальным зазором

В этом разделе мы дадим определение гиперплоскости и введем понятие оптимальной разделяющей гиперплоскости.

[1] Аббревиатура от «support vector machine». — *Прим. пер.*

9.1.1 Что такое гиперплоскость?

гиперплоскость

В контексте p–мерного пространства гиперплоскость представляет собой плоское аффинное подпространство[2] с размерностью $p-1$. Например, в двумерном пространстве гиперплоскость — это плоское одномерное подпространство, т. е. линия. В трехмерном пространстве гиперплоскость представлена двумерным подпространством, т. е. плоскостью. При наличии $p > 3$ размерностей гиперплоскость сложно изобразить графически, однако принцип $(p-1)$–мерного плоского подпространства по–прежнему применим.

Математическое определение гиперплоскости довольно простое. В двумерном пространстве гиперплоскость с параметрами β_0, β_1 и β_2 задается уравнением

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0. \tag{9.1}$$

Когда мы говорим, что (9.1) «задает» гиперплоскость, мы имеем в виду, что любая величина $X = (X_1, X_2)^T$, в отношении которой уравнение (9.1) является справедливым, представляет собой точку на этой гиперплоскости. Заметьте, что (9.1) — это просто уравнение прямой, поскольку в двумерном пространстве гиперплоскость действительно является линией.

Уравнение (9.1) можно легко расширить на случай p–мерного пространства:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0. \tag{9.2}$$

Это уравнение задает p–мерную гиперплоскость в том смысле, что если точка $X = (X_1, X_2, \ldots, X_p)^T$ в p–мерном пространстве удовлетворяет условию (9.2), то эта точка лежит на гиперплоскости.

Теперь представьте, что X не удовлетворяет условию (9.2), а именно:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0. \tag{9.3}$$

Это говорит нам о том, что X находится на определенной стороне относительно гиперплоскости. В то же время если

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0, \tag{9.4}$$

то X находится на противоположной стороне от гиперплоскости. Таким образом, мы можем думать о гиперплоскости как о чем–то, что разделяет p–мерное пространство на две половины. Местоположение той или иной точки относительно гиперплоскости можно легко определить, вычислив знак выражения, находящегося в уравнении (9.2) слева от знака «равно». Пример гиперплоскости в двумерном пространстве показан на рис. 9.1.

9.1.2 Классификация с использованием гиперплоскости

Представим теперь, что у нас есть матрица с данными $\mathbf{X}$ размером $n \times p$, содержащая n обучающих наблюдений в p–мерном пространстве

[2] Термин «аффинный» указывает на то, что подпространство не обязательно проходит через начало координат.

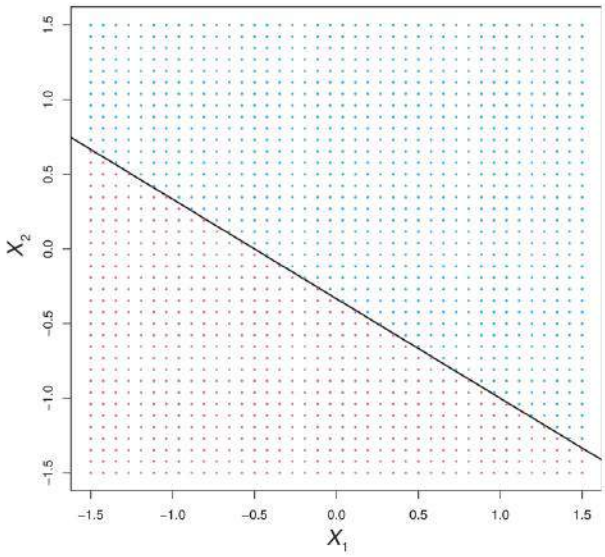

РИСУНОК 9.1. *Показана гиперплоскость* $1 + 2X_1 + 3X_2 = 0$. *Голубая область представляет собой множество точек, для которых* $1 + 2X_1 + 3X_2 > 0$, *а фиолетовая область — множество точек, для которых* $1 + 2X_1 + 3X_2 < 0$

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}, \tag{9.5}$$

и что эти наблюдения относятся к двум классам, т. е. $y_1, \ldots, y_n \in \{-1, 1\}$, где -1 обозначает один класс, а 1 — другой класс. У нас есть также контрольное наблюдение — p–мерный вектор со значениями признаков $x^* = (x_1^* \ \ldots \ x_p^*)^T$. Наша цель заключается в создании классификатора на основе обучающих данных, который позволит правильно предсказать контрольное наблюдение по значениям его признаков. Мы уже познакомились с целым рядом методов для решения этой задачи, таких как линейный дискриминантный анализ и логистическая регрессия в главе 4 и деревья классификации, бэггинг и бустинг в главе 8. Теперь мы рассмотрим новый подход, который основан на понятии *разделяющей гиперплоскости*[3].

разделяющая гиперплоскость

Представьте, что имеется возможность сконструировать такую гиперплоскость, которая разделяет обучающие наблюдения в точном соответствии с метками их классов. Примеры подобных *разделяющих гиперплоскостей* показаны на слева рис. 9.2. Мы можем обозначить наблюдения из «голубого класса» как $y_i = 1$, а наблюдения из «фиолетового класса» — как $y_i = -1$. Тогда разделяющая гиперплоскость будет иметь следующие свойства:

[3] В оригинале используется термин «separating hyperplane». — *Прим. пер.*

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_p x_{ip} > 0 \text{ если } y_i = 1 \tag{9.6}$$

и

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_p x_{ip} < 0 \text{ если } y_i = -1. \tag{9.7}$$

Можно также сказать, что разделяющей является гиперплоскость со следующим свойством:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_p x_{ip}) > 0 \tag{9.8}$$

для всех $i = 1, \ldots, n$.

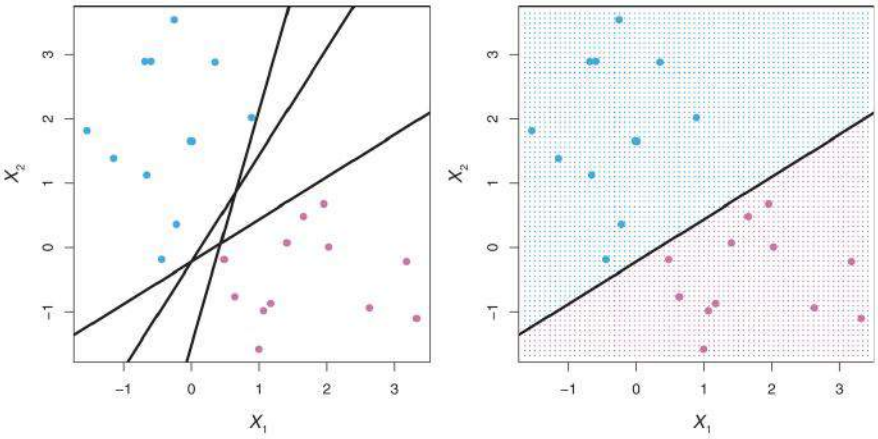

РИСУНОК 9.2. Слева: *есть два класса наблюдений (показаны голубым и фиолетовым цветом), для каждого из которых были измерены две переменные. Черным цветом показаны три (из многих возможных) разделяющие гиперплоскости.* Справа: *разделяющая гиперплоскость показана черным цветом. Голубая и фиолетовая области соответствуют правилу принятия решений, созданному классификатором на основе разделяющей гиперплоскости: контрольное наблюдение, попадающее в голубую область, будет отнесено к «голубому классу», а контрольное наблюдение, попадающее в фиолетовую область, будет отнесено к «фиолетовому классу»*

Если разделяющая плоскость существует, то мы можем использовать ее для построения совершенно естественного классификатора: контрольное наблюдение классифицируют в зависимости от того, по какую сторону от разделяющей гиперплоскости оно находится. Справа на рис. 9.2 приведен пример подобного классификатора. Другими словами, мы классифицируем контрольное наблюдение x^* в зависимости от знака функции $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$. Если функция $f(x^*)$ положительна, то мы относим контрольное наблюдение к классу 1, а если она отрицательна — то к классу -1. Мы можем также принять во внимание конкретное *значение* $f(x^*)$. Если значение $f(x^*)$ существенно отличается от нуля, то это означает, что x^* находится далеко от гиперплоскости и, соответственно,

мы можем быть уверены в правильной классификации x^*. С другой стороны, если значение $f(x^*)$ близко к нулю, то x^* находится близко к гиперплоскости, в связи с чем мы менее уверены в правильной классификации x^*. Неудивительно, что классификатор, основанный на разделяющей гиперплоскости, приводит к линейной решающей границе (см. рис. 9.2).

9.1.3 Классификатор с максимальным зазором

Если наши данные можно идеально разделить при помощи гиперплоскости, то обычно таких гиперплоскостей будет бесконечное множество. Это обусловлено тем, что некоторую гиперплоскость можно сдвинуть на очень небольшое расстояние вверх или вниз, не задев при этом ни одного из наблюдений. Слева на рис. 9.2 показаны три возможные гиперплоскости. Для создания классификатора на основе разделяющей гиперплоскости нам необходим какой–то приемлемый способ принятия решения в отношении того, какую из бесконечно большого числа возможных гиперплоскостей выбрать.

гиперплоскость с максимальным зазором

Естественным выбором является *гиперплоскость с максимальным зазором* (также *«оптимальная разделяющая гиперплоскость»*)[4], представляющая собой гиперплоскость, максимально удаленную от обучающих наблюдений. Другими словами, мы можем вычислить (перпендикулярное) расстояние от каждого наблюдения до некоторой разделяющей гиперплоскости; минимальное из этих расстояний называют *зазором*. Гиперплоскость с максимальным зазором представляет собой разделяющую гиперплоскость, которая удалена от ближайших к ней обучающих наблюдений на наибольшее расстояние. Далее мы можем классифицировать некоторое контрольное наблюдение с учетом того, по какую сторону от гиперплоскости с максимальным зазором оно находится. Такой классификатор называют *классификатором с максимальным зазором*[5]. Мы надеемся, что классификатор, обладающий максимальным зазором для обучающих данных, будет иметь широкий зазор и для контрольных данных, и, следовательно, мы будем правильно классифицировать контрольные наблюдения. Хотя классификатор с максимальным зазором часто дает хорошие результаты, при больших p он может приводить к переобучению.

зазор

классификатор с максимальным зазором

Если $\beta_0, \beta_1, \ldots, \beta_p$ — это коэффициенты гиперплоскости с максимальным зазором, тогда классификатор с максимальным зазором отнесет некоторое контрольное наблюдение x^* к тому или иному классу в зависимости от знака $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$.

На рис. 9.3 показана гиперплоскость, обеспечивающая максимальный зазор для данных из рис. 9.2. Сравнив рис. 9.3 с графиком, представленным справа на рис. 9.2, мы увидим, что гиперплоскость на рис. 9.3 действительно обеспечивает максимальный зазор. В каком–то смысле эта гиперплоскость представляет собой срединную линию наиболее широкого «бруска», который мы можем вставить между двумя классами.

На рис. 9.3 мы видим, что три обучающих наблюдения находятся на одинаковом удалении от гиперплоскости с максимальным зазором и лежат на прерывистых линиях, обозначающих ширину зазора. Эти три на-

[4] В оригинале используются термины «maximal margin hyperplane» и «optimal separating hyperplane» соответственно. — *Прим. пер.*

[5] В оригинале используется термин «maximal margin classifier». — *Прим. пер.*

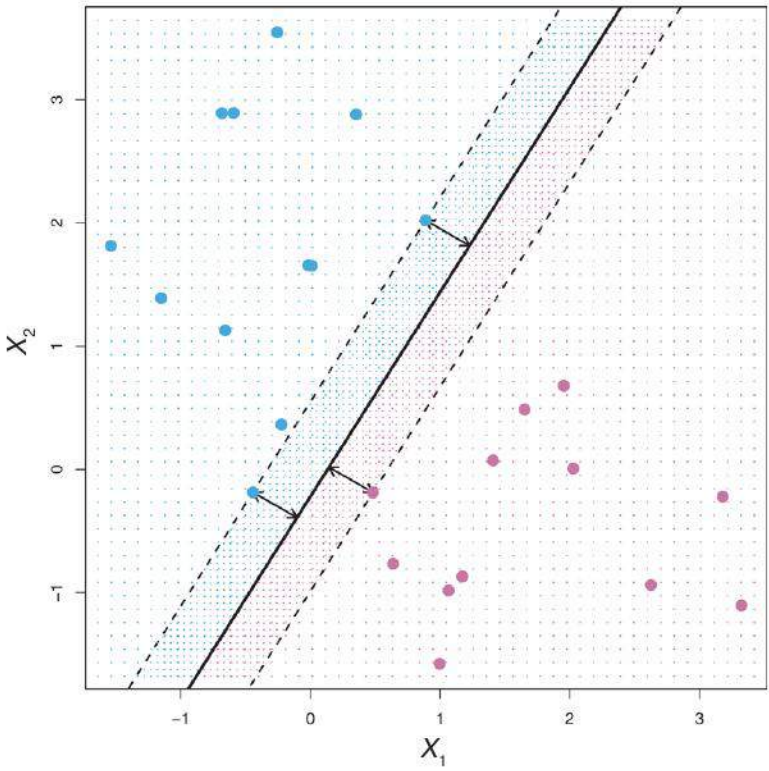

РИСУНОК 9.3. *Имеются наблюдения из двух классов, показанные в виде голубых и фиолетовых точек. Гиперплоскость, обеспечивающая максимальный зазор, показана в виде сплошной линии. Зазор — это расстояние от сплошной линии до любой из прерывистых линий. Две голубые точки и фиолетовая точка, лежащие на прерывистых линиях, являются опорными векторами; расстояние от этих точек до гиперплоскости показано при помощи стрелок. Фиолетовая и голубая области соответствуют решающему правилу, полученному при помощи классификатора на основе этой разделяющей гиперплоскости*

опорные векторы

блюдения известны как *опорные векторы*, поскольку они являются векторами в p–мерном пространстве (на рис. 9.3 $p = 2$) и «поддерживают» гиперплоскость в том смысле, что если эти точки незначительно сдвинуть, то сдвинется и сама гиперплоскость. Интересно, что гиперплоскость с максимальным зазором определяется исключительно опорными векторами: сдвиг любого из остальных наблюдений не повлиял бы на нее, если только в результате этого сдвига точка не пересечет границу зазора. Тот факт, что гиперплоскость с максимальным зазором зависит только от небольшого подмножества наблюдений, является важным свойством, к которому мы вернемся в этой главе при обсуждении классификатора на опорных векторах и машин опорных векторов.

9.1.4 Построение классификатора с максимальным зазором

Теперь мы рассмотрим задачу нахождения гиперплоскости с максимальным зазором по выборке из n обучающих наблюдений $x_1, \ldots, x_n \in \mathbb{R}^p$ и соответствующих им меток классов $y_1, \ldots, y_n \in \{-1, 1\}$. Вкратце, такая гиперплоскость представляет собой решение следующей оптимизационной проблемы:

$$\underset{\beta_0, \beta_1, \ldots, \beta_p}{\text{максимизировать}}\ M \tag{9.9}$$

$$\text{при условии, что } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.10}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \ \forall\ i = 1, \ldots, n. \tag{9.11}$$

В действительности оптимизационная проблема (9.9)–(9.11) проще, чем она выглядит. Прежде всего ограничение (9.11)

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \ \forall\ i = 1, \ldots, n$$

гарантирует, что каждое наблюдение будет лежать по правильную сторону от гиперплоскости при условии, что M принимает положительное значение. (На самом деле для этого нам достаточно было бы иметь $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0$; таким образом, ограничение (9.11) задано с некоторым запасом, т. е. требуется, чтобы значение M было также положительным.)

Во–вторых, обратите внимание на то, что (9.10) не является строгим ограничением, поскольку если гиперплоскость задается выражением $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} = 0$, то с таким же успехом она задается и выражением $k(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) = 0$ для любых $k \neq 0$. Однако (9.10) наполняет смыслом (9.11): можно показать, что в присутствии этого ограничения перпендикулярное расстояние от i–го наблюдения до гиперплоскости определяется как

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}).$$

Следовательно, ограничения (9.10) и (9.11) позволяют убедиться в том, что каждое наблюдение лежит по правильную сторону от гиперплоскости, и как минимум на расстоянии M от нее. Таким образом, M представляет собой зазор нашей гиперплоскости и приведенная выше оптимизационная проблема заключается в нахождении $\beta_0, \beta_1, \ldots, \beta_p$, которые максимизируют M. Это, в свою очередь, в точности является определением гиперплоскости с максимальным зазором! Задачу (9.9)–(9.11) можно решить эффективным способом, однако его детали лежат за рамками данной книги.

9.1.5 Случай, когда разделяющая гиперплоскость не существует

Классификатор с максимальным зазором представляет собой естественный инструмент для выполнения классификации *при условии, что раз-*

деляющая гиперплоскость существует. Однако, как мы уже намекали ранее, во многих случаях такая гиперплоскость, а следовательно, и классификатор с максимальным зазором не существуют. При таком сценарии оптимизационная проблема (9.9)–(9.11) для $M > 0$ не имеет решения. На рис. 9.4 приведен пример. Здесь мы не можем *четко* разделить два класса. Однако, как мы увидим в следующем разделе, мы можем развить идею разделяющей гиперплоскости и сконструировать такую гиперплоскость, которая почти полностью разделяет классы при помощи т. н. *мягкого зазора*[6]. Обобщение классификатора с максимальным зазором на случай неразделимых классов дает метод, известный как *классификатор на опорных векторах*[7].

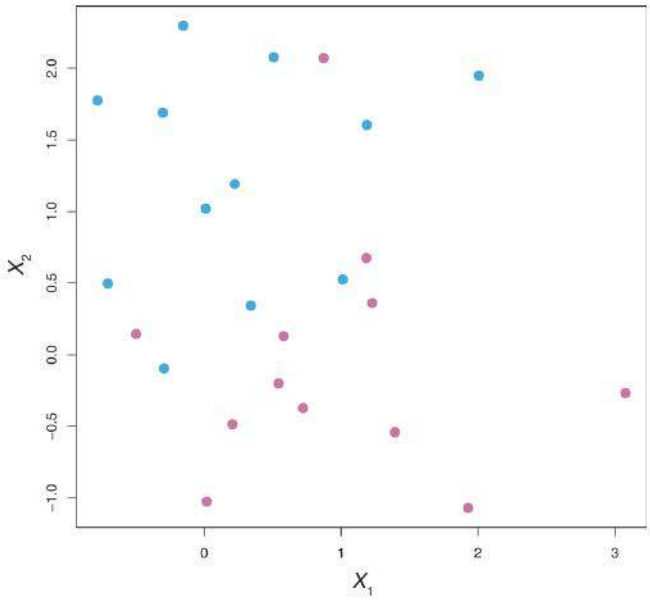

РИСУНОК 9.4. *Имеются наблюдения из двух классов, показанные в виде голубых и фиолетовых точек. В данном случае разделить два класса при помощи гиперплоскости невозможно, и поэтому классификатор с максимальным зазором неприменим*

9.2 Классификаторы на опорных векторах

9.2.1 Общие представления о классификаторах на опорных векторах

Как показано на рис. 9.4, наблюдения, принадлежащие к двум классам, не всегда можно разделить при помощи гиперплоскости. Более того, даже

[6] В оригинале используется термин «soft margin». — *Прим. пер.*
[7] В оригинале используется термин «support vector classifier». — *Прим. пер.*

если разделяющая гиперплоскость существует, встречаются случаи, когда классификатор, основанный на такой гиперплоскости, может оказаться нежелательным. Классификатор на основе разделяющей гиперплоскости всегда будет идеально классифицировать все обучающие наблюдения, а это может привести к повышенной чувствительности в отношении отдельных наблюдений. На рис. 9.5 приведен пример. Справа на этом рисунке добавление всего лишь одного наблюдения вызывает существенное изменение гиперплоскости с максимальным зазором. Получающаяся гиперплоскость неудовлетворительна: прежде всего она обладает очень узким зазором. Это проблематично, поскольку, как мы обсуждали ранее, расстояние от того или иного наблюдения до гиперплоскости можно рассматривать как меру нашей уверенности в том, что это наблюдение было классифицировано правильно. Кроме того, чрезвычайно высокая чувствительность гиперплоскости с максимальным зазором к изменению одного наблюдения предполагает, что мы, возможно, переобучаем модель по обучающим данным.

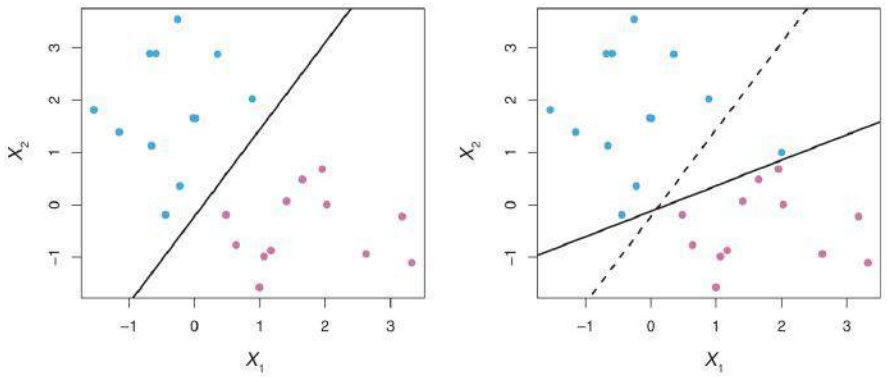

РИСУНОК 9.5. Слева: *показаны наблюдения из двух классов в виде голубых и фиолетовых точек, а также гиперплоскость с максимальным зазором.* Справа: *добавлена еще одна голубая точка, вызывающая существенный сдвиг гиперплоскости с максимальным зазором (показана в виде сплошной линии). Прерывистая линия соответствует гиперплоскости с максимальным зазором, полученной в отсутствие этой дополнительной точки*

Возможно, в данном случае нам стоит рассмотреть классификатор на основе гиперплоскости, *не выполняющей* четкого разделения двух классов, что позволит обеспечить:

- более высокую устойчивость к отдельным наблюдениям и
- более высокое качество классификации *большинства* обучающих наблюдений.

Иными словами, неверная классификация нескольких обучающих наблюдений может оказаться полезной для более качественной классификации остальных наблюдений.

классификатор на опорных векторах

Классификатор на опорных векторах, который иногда называют также *классификатором с мягким зазором*, именно это и делает. Вместо нахождения максимально возможного зазора, при котором каждое наблюдение лежит не только на правильной стороне относительно гиперплоскости, но еще и на правильной стороне относительно соответствующей границы зазора, мы позволяем некоторым наблюдениям находиться на неправильной стороне относительно границы зазора или даже на неправильной стороне относительно гиперплоскости. (Зазор называют *мягким*, поскольку некоторые наблюдения могут нарушать его границы.) Пример приведен слева на рис. 9.6. Большинство наблюдений лежат по правильную сторону относительно границы зазора. Однако небольшая часть наблюдений лежит на неправильной стороне относительно этой границы.

Некоторое наблюдение может находиться не только на неправильной стороне относительно границы зазора, но и на неправильной стороне относительно гиперплоскости. Более того, когда разделяющая гиперплоскость не существует, такая ситуация неизбежна. Наблюдения, лежащие по неправильную сторону относительно гиперплоскости, соответствуют обучающим наблюдениям, которые неверно предсказаны классификатором. Такой сценарий показан справа на рис. 9.6.

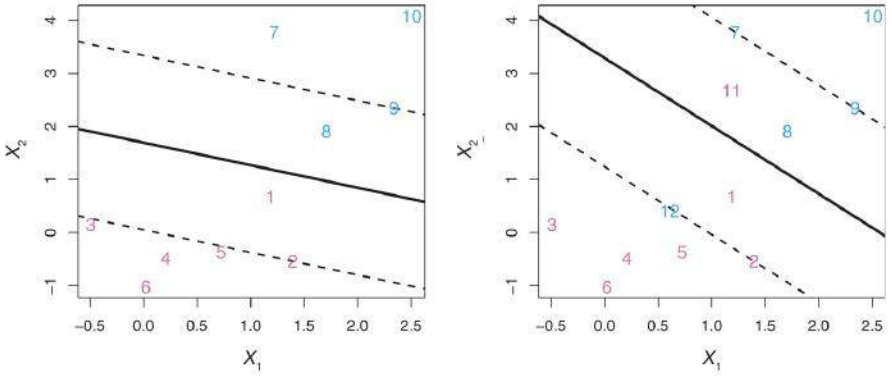

РИСУНОК 9.6. Слева: *Классификатор на опорных векторах построен с использованием небольшого набора данных. Гиперплоскость показана в виде сплошной линии, а границы зазора — в виде прерывистых линий.* Фиолетовые значения: *наблюдения 3, 4, 5 и 6 находятся на правильной стороне относительно границы зазора, наблюдение 2 — на границе зазора, а наблюдение 1 — на неправильной стороне относительно границы зазора.* Голубые значения: *Наблюдения 7 и 10 находятся на правильной стороне относительно границы зазора, наблюдение 9 — на границе зазора, а наблюдение 8 — на неправильной стороне относительно границы зазора. Ни одно из наблюдений не находится на неправильной стороне относительно гиперплоскости.* Справа: *То же, но с двумя дополнительными наблюдениями — 11 и 12. Эти два наблюдения лежат на неправильной стороне относительно как гиперплоскости, так и границы зазора*

9.2.2 Более подробное описание классификатора на опорных векторах

Классификатор на опорных векторах относит контрольное наблюдение к тому или иному классу в зависимости от того, по какую сторону относительно гиперплоскости оно находится. Гиперплоскость выбирается так, чтобы правильно разделять на два класса большинство обучающих наблюдений, но при этом разрешается неверно классифицировать некоторую небольшую группу наблюдений. Это является решением следующей оптимизационной проблемы:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{максимизировать}}\ M \tag{9.12}$$

$$\text{при условии, что } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.13}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)\ \forall\ i = 1, \ldots, n, \tag{9.14}$$

$$\epsilon_i \geq 0,\ \sum_{i=1}^{n} \epsilon_i \leq C, \tag{9.15}$$

где C — это некоторый неотрицательный гиперпараметр. Как и в (9.11), M — это ширина зазора, которую мы пытаемся сделать максимально возможной. В (9.14) $\epsilon_1, \ldots, \epsilon_n$ представляют собой *фиктивные переменные*, которые позволяют отдельным наблюдениям находиться на неправильной стороне относительно границы зазора или относительно гиперплоскости; ниже мы обсудим их подробнее. Решив (9.12)–(9.15), мы выполняем классификацию контрольного наблюдения x^* обычным способом, т. е. просто выясняя, по какую сторону от гиперплоскости оно лежит. Иными словами, мы классифицируем контрольное наблюдение в зависимости от знака $f(x^*) = \beta_0 + \beta_1 x_1^* + \cdots + \beta_p x_p^*$.

фиктивные переменные

Проблема (9.12)–(9.15) выглядит сложной, однако ее можно понять, сделав несколько простых описанных ниже наблюдений. Прежде всего фиктивная переменная ϵ_i говорит нам о том, где находится i–е наблюдение относительно гиперплоскости и границы зазора. Если $\epsilon_i = 0$, то i–е наблюдение лежит на правильной стороне относительно границы зазора (мы уже видели такую ситуацию в подразделе 9.1.4). Если $\epsilon_i > 0$, то i–е наблюдение лежит на неправильной стороне относительно границы зазора, в связи с чем мы говорим, что i–е наблюдение *нарушило* эту границу. Если $\epsilon_i > 1$, то данное наблюдение лежит на неправильной стороне относительно гиперплоскости.

Теперь мы рассмотрим роль гиперпараметра C. В (9.14) C ограничивает сумму значений ϵ_i и тем самым задает допустимое число нарушений границы зазора (или гиперплоскости) и их выраженность. Мы можем думать о C как о *бюджете* некоторой величины, отражающей степень нарушений границы зазора n наблюдениями. Если $C = 0$, то возможности для нарушений границы зазора нет, в связи с чем $\epsilon_1 = \ldots \epsilon_n = 0$, и проблема (9.12)–(9.15) просто сводится к оптимизационной проблеме (9.9)–(9.11) для гиперплоскости с максимальным зазором. (Конечно, гиперплоскость с максимальным зазором существует только если два класса разделимы.)

При $C > 0$ на неправильной стороне относительно гиперплоскости могут находиться не более C наблюдений, поскольку если некоторое наблюдение лежит на неправильной стороне относительно гиперплоскости, то $\epsilon_i > 1$, а (9.14) требует, чтобы $\sum_{i=1}^{n} \epsilon_i \leq C$. При увеличении бюджета C допустимая степень нарушений границ зазора возрастает, и поэтому зазор расширяется. И наоборот: при уменьшении C допустимая степень нарушений границ зазора снижается, и поэтому он сужается. На рис. 9.7 приведен пример.

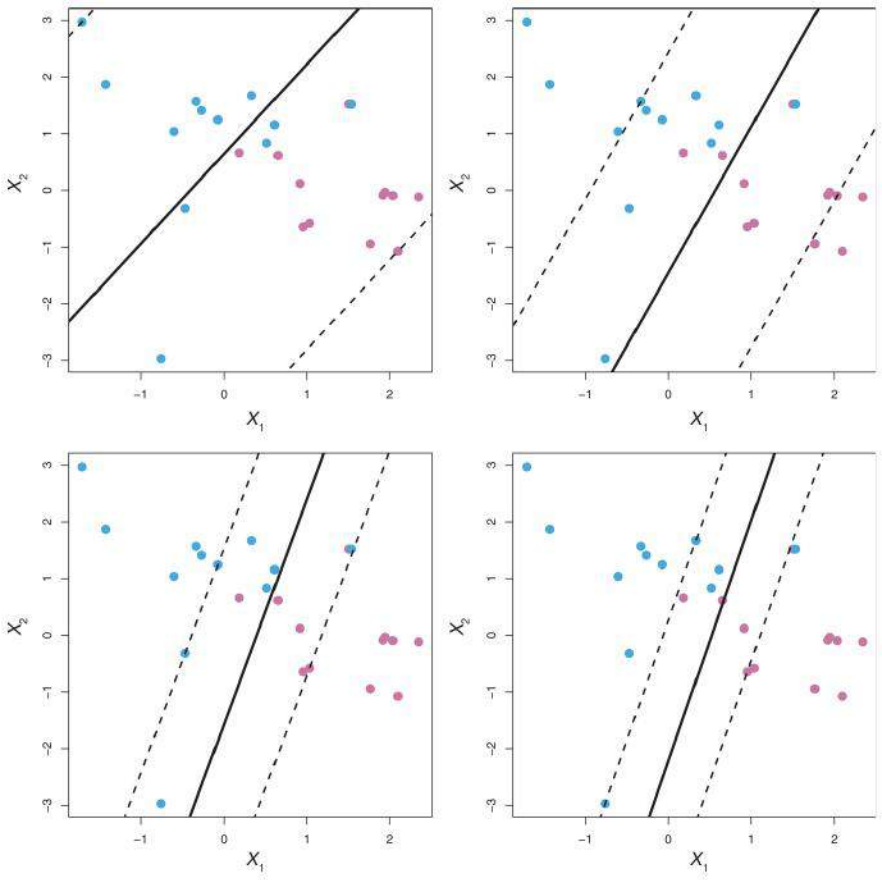

РИСУНОК 9.7. *Классификатор на опорных векторах построен с использованием четырех различных значений гиперпараметра C из (9.12)–(9.15). Максимальное значение C было применено на графике, представленном слева вверху, а на остальных графиках были использованы меньшие значения. При большом C допустимая степень нарушений границ зазора высока, в связи с чем зазор будет широким. По мере уменьшения C эта степень снижается, и зазор сужается*

На практике C используется как гиперпараметр, значение которого обычно выбирают при помощи перекрестной проверки. Как и в случае с другими гиперпараметрами, которые мы встречали в этой книге, C контролирует степень компромисса между смещением и дисперсией модели.

При небольшом C мы пытаемся найти узкий зазор, границы которого нарушаются редко; эта ситуация соответствует классификатору, который очень хорошо аппроксимирует данные и характеризуется небольшим смещением, но при этом обладает высокой дисперсией. С другой стороны, при большом C зазор является широким, и мы разрешаем большему числу наблюдений нарушать его границы; это соответствует менее качественной аппроксимации данных и построению классификатора с потенциально большим смещением, но низкой дисперсией.

У оптимизационной проблемы (9.12)–(9.15) есть одно очень интересное свойство: получается, что влияние на гиперплоскость, а следовательно, и на полученный классификатор будут оказывать только те наблюдения, которые лежат на границах зазора или нарушают их. Другими словами, наблюдение, которое находится строго на правильной стороне относительно границы зазора, не влияет на классификатор на опорных векторах. Изменение положения этого наблюдения никак не изменило бы классификатор, при условии что это наблюдение по–прежнему лежит на правильной стороне относительно границы зазора. Наблюдения, лежащие непосредственно на границах зазора, или на неправильной (для своего класса) стороне относительно этих границ, называются *опорными векторами*. Эти наблюдения оказывают влияние на классификатор на основе опорных векторов.

Тот факт, что на классификатор влияют только опорные векторы, согласуется со сделанным нами ранее утверждением о том, что C контролирует достигаемую классификатором степень компромисса между смещением и дисперсией. При большом параметре C зазор получается широким, многие наблюдения нарушают его границы, и поэтому имеется много опорных векторов. В этом случае гиперплоскость задается большим числом наблюдений. Данный сценарий показан слева вверху на рис. 9.7: этот классификатор обладает низкой дисперсией (поскольку многие наблюдения являются опорными векторами), но потенциально высоким смещением. При небольшом же C опорных векторов будет меньше, и поэтому получаемый классификатор будет обладать низким смещением и высокой дисперсией. Этот сценарий показан на рис. 9.7 справа внизу (есть только восемь опорных векторов).

Тот факт, что решающее правило классификатора на опорных векторах задается потенциально небольшим подмножеством обучающих наблюдений, означает, что оно довольно устойчиво к поведению наблюдений, расположенных далеко от гиперплоскости. Это свойство отличается от свойств некоторых других методов классификации, с которыми мы познакомились в предыдущих разделах (например, линейного дискриминантного анализа). Вспомните, что решающее правило LDA зависит от среднего значения *всех* наблюдений в каждом классе, а также от ковариационной матрицы, которая вычисляется по *всем* имеющимся наблюдениям. В то же время логистическая регрессия, в отличие от LDA, обладает очень низкой чувствительностью к наблюдениям, находящимся далеко от решающей границы. Более того, в разделе 9.5 мы увидим, что классификатор на опорных векторах и логистическая регрессия являются близкими родственниками.

9.3 Машины опорных векторов

Сначала мы опишем общий механизм, позволяющий преобразовать линейный классификатор в классификатор с нелинейными решающими границами. Затем мы представим машину опорных векторов — метод, который делает это автоматически.

9.3.1 Классификация с использованием нелинейных решающих границ

Классификатор на опорных векторах представляет собой естественный подход для распознавания наблюдений из двух классов, когда решающая граница между этими классами является линейной. Однако на практике мы в ряде случаев сталкиваемся с нелинейными границами между классами. Рассмотрим, например, данные, изображенные слева на рис. 9.8. Четко видно, что классификатор на опорных векторах или любой другой линейный классификатор сработает на этих данных плохо. И действительно, как показано справа на рис. 9.8, классификатор на опорных векторах здесь бесполезен.

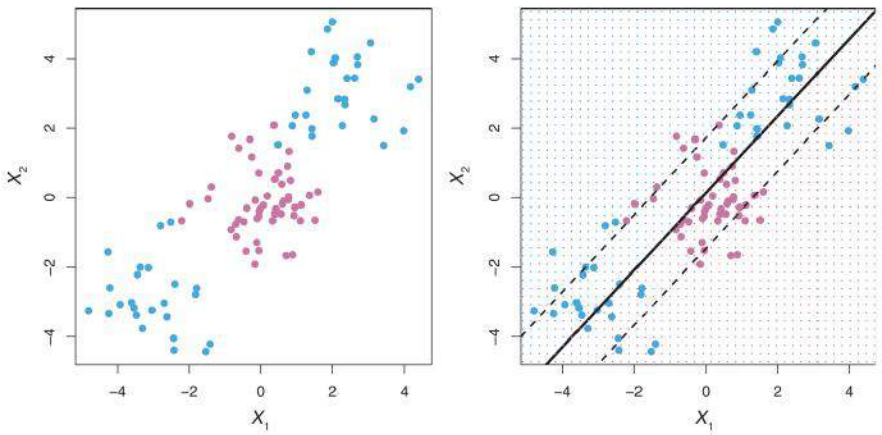

РИСУНОК 9.8. Слева: *наблюдения образуют два класса, разделенных нелинейной границей.* Справа: *классификатор на опорных векторах дает линейную разделяющую границу и, следовательно, обладает очень низким качеством*

В главе 7 мы столкнулись с аналогичной ситуацией. Там мы выяснили, что при наличии нелинейной связи между предикторами и откликом качество линейной регрессии может быть неудовлетворительным. Для учета нелинейности мы расширяем пространство предикторов, используя функции от исходных предикторов (в частности, квадратичные и кубические). В случае с классификатором на опорных векторах мы могли бы учесть возможные нелинейные границы между классами похожим образом, т. е. расширив пространство предикторов при помощи их полиномов второй,

третьей и более высоких степеней. Например, вместо построения классификатора на опорных векторах с использованием p признаков

$$X_1, X_2, \ldots, X_p$$

мы могли бы построить классификатор с использованием $2p$ признаков

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2.$$

Тогда (9.12)–(9.15) превратились бы в следующую задачу:

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n}{\text{максимизировать}} \quad M \tag{9.16}$$

$$\text{при условии, что } y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i),$$

$$\sum_{i=1}^{n} \epsilon_i \leq C,\ \epsilon_i \geq 0,\ \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1.$$

Почему это приводит к нелинейной решающей границе? В расширенном пространстве предикторов решающая граница, вытекающая из (9.16), в действительности является линейной. Однако в исходном пространстве предикторов решающая граница имеет вид $q(x) = 0$, где q — это полином второй степени, и ее решения являются нелинейными. Мы могли бы расширить пространство предикторов также при помощи полиномов более высокой степени и с использованием переменных вида $X_j X_{j'}$ для $j \neq j'$, отражающих взаимодействие между предикторами. В качестве альтернативы можно было бы рассмотреть и другие функции предикторов, отличные от полиномов. Нетрудно увидеть, что существует множество способов расширить пространство предикторов и что без соблюдения должной осторожности мы в итоге могли бы получить огромное число переменных. Это сделало бы вычисления невозможными. Описанная ниже машина опорных векторов позволяет нам расширить пространство предикторов для классификатора на опорных векторах таким образом, чтобы вычисления оставались эффективными.

9.3.2 Машина опорных векторов

машина опорных векторов

ядерная функция

Машина опорных векторов (SVM)[8] представляет собой более общую версию классификатора на опорных векторах, которую получают путем расширения пространства предикторов особым образом с помощью *ядерных функций*. Мы обсудим этот метод ниже, хотя его технические детали довольно сложны и выходят за рамки данной книги. Тем не менее основная идея этого метода описана в подразделе 9.3.1: мы можем расширить наше пространство предикторов для учета нелинейной формы границы между классами. Использование рассматриваемых здесь ядерных функций есть не что иное, как эффективный способ реализации этой идеи.

[8] Аббревиатура от «support vector machine». — *Прим. пер.*

До сих пор мы не обсуждали, как именно вычисляют классификатор на основе опорных векторов, поскольку соответствующие детали приобретают довольно технический характер. Оказывается, однако, что решение проблемы (9.12)–(9.15) для классификатора на основе опорных векторов базируется на использовании только *скалярных произведений* наблюдений, а не самих исходных наблюдений. Скалярное произведение двух r–векторов a и b вычисляется как $\langle a, b\rangle = \sum_{i=1}^{r} a_i b_i$. Тогда скалярное произведение двух наблюдений $x_i, x_{i'}$ определяется как

$$\langle x_i, x_{i'}\rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}. \tag{9.17}$$

Можно показать следующее:

- Линейный классификатор на опорных векторах можно представить как

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i\rangle, \tag{9.18}$$

  где имеется n параметров α_i, $i = 1, \ldots, n$, по одному на каждое обучающее наблюдение.

- Все, что нам требуется для получения оценок параметров $\alpha_1, \ldots, \alpha_n$ и β_0, — это $\binom{n}{2}$ скалярных произведений $\langle x_i, x_{i'}\rangle$ между всеми парами обучающих наблюдений. (Нотация $\binom{n}{2}$ означает $n(n-1)/2$ и показывает число пар в наборе из n элементов.)

Заметьте, что в (9.18) для нахождения функции $f(x)$ нам необходимо вычислить скалярное произведение между новым наблюдением x и каждым из обучающих наблюдений x_i. Оказывается, однако, что при решении этой проблемы α_i принимают ненулевые значения только для опорных векторов, т. е. если обучающее наблюдение не является опорным вектором, то его параметр α_i равен нулю. Таким образом, если $\mathcal{S}$ представляет собой множество индексных номеров этих опорных точек, то мы можем представить любое решение для функции (9.18) в виде уравнения

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i\rangle, \tag{9.19}$$

которое обычно содержит намного меньше членов, чем (9.18)[9].

Таким образом, для представления линейного классификатора $f(x)$ и вычисления его коэффициентов нам не нужно ничего, кроме скалярных произведений.

Теперь представьте, что каждый раз, когда скалярное произведение (9.17) появляется в выражении (9.18), т. е. при нахождении классификатора на опорных векторах, мы заменяем его на следующую обобщенную форму скалярного произведения:

[9] Расписав подробнее каждое скалярное произведение в (9.19), легко увидеть, что $f(x)$ является линейной функцией координат x. Мы также увидим связь между α_i и исходными параметрами β_j.

$$K(x_i, x_{i'}), \tag{9.20}$$

где K — это некоторая функция, которую мы будем называть *ядром*. Ядро представляет собой функцию, которая количественно выражает сходство между двумя наблюдениями. Например, мы бы могли просто взять функцию

ядро

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}, \tag{9.21}$$

которая дала бы нам обычный классификатор на опорных векторах. Уравнение (9.21) известно как *линейное* ядро, поскольку классификатор на опорных векторах является линейным по предикторам; по сути, линейное ядро количественно выражает сходство между парой наблюдений в виде стандартного коэффициента корреляции Пирсона. Однако можно было бы выбрать и другую форму для (9.20). Например, вместо $\sum_{j=1}^{p} x_{ij} x_{i'j}$ можно было бы применить

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d. \tag{9.22}$$

Данное уравнение известно как *полиномиальное ядро* степени d, где d — это положительное целое число. Использование такого ядра с $d > 1$ вместо стандартного линейного ядра (9.21) в алгоритме классификатора на опорных векторах приводит к гораздо более гибкой решающей границе. В сущности, происходит построение классификатора на опорных векторах в пространстве большей размерности, которое, в отличие от пространства исходных переменных, включает также полиномы степени d. При объединении классификатора на опорных векторах с нелинейным ядром, подобным (9.22), получают классификатор, который называется машиной опорных векторов. Заметьте, что в этом случае (нелинейная) функция принимает форму

полиномиальное ядро

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i). \tag{9.23}$$

На рис. 9.9 слева показан пример SVM–классификатора с полиномиальным ядром для нелинейных данных из рис. 9.8. Эта модель существенно лучше линейного классификатора на опорных векторах. При $d = 1$ SVM сводится к линейному классификатору на опорных векторах, который мы видели ранее в этой главе.

Полиномиальное ядро (9.22) — это один из возможных примеров нелинейных ядер, однако существует и множество других альтернатив. Популярным является также *радиальное ядро* следующего вида:

радиальное ядро

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2). \tag{9.24}$$

В уравнении (9.24) γ — это некоторая положительная константа. Справа на рис. 9.9 приведен пример SVM–классификатора с радиальным ядром для этих нелинейных данных; он также хорошо разделяет два класса.

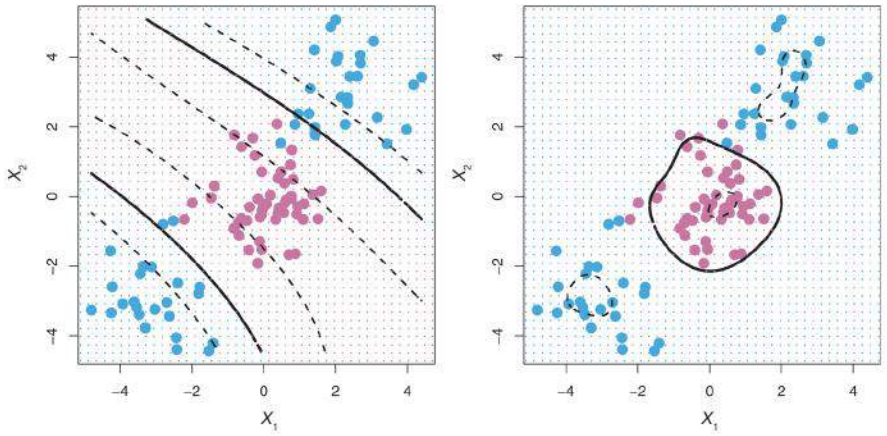

РИСУНОК 9.9. Слева: *SVM–классификатор на основе полиномиального ядра 3-й степени применен к нелинейным данным из рис. 9.8, что дает более адекватную решающую границу.* Справа: *применен SVM–классификатор с радиальным ядром. В данном примере решающая граница хорошо описывается любым из этих ядер*

Как именно работает радиальное ядро (9.24)? Если некоторое контрольное наблюдение $x^* = (x_1^* \dots x_p^*)^T$ находится далеко от обучающего наблюдения x_i с точки зрения евклидова расстояния, то величина $\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2$ будет большой, в связи с чем значение $K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p}(x_{ij} - x_{i'j})^2)$ будет очень низким. Это значит, что x_i не будет играть практически никакой роли в определении $f(x^*)$ в (9.23). Напомним, что метка класса, предсказываемая для контрольного наблюдения x^*, зависит от знака $f(x^*)$. Другими словами, обучающие наблюдения, находящиеся далеко от x^*, практически не будут играть никакой роли в предсказании класса x^*. Это значит также, что радиальное ядро проявляет очень *локальное* поведение в том смысле, что на метку класса контрольного наблюдения влияют только близко расположенные к нему обучающие наблюдения.

В чем заключается преимущество использования ядра в сравнении с простым расширением пространства признаков при помощи функций исходных признаков подобно тому, как это сделано в (9.16)? Одно из преимуществ имеет вычислительный характер и состоит в том, что при использовании ядер требуется вычислить $K(x_i, x_i')$ только для всех $\binom{n}{2}$ уникальных пар i, i'. Это можно сделать без непосредственной работы с расширенным пространством признаков. Данное обстоятельство является важным, поскольку во многих случаях применения SVM расширенное пространство признаков настолько велико, что вычисления становятся практически невозможными. Для некоторых ядер, таких как радиальное ядро (9.24), пространство признаков является *неявным* и обладает бесконечной размерностью, в связи с чем мы в любом случае не смогли бы выполнить в нем никаких вычислений!

9.3.3 Применение к данным по нарушению сердечной функции

В главе 8 мы применили деревья решений и родственные им методы к данным `Heart`. Цель состоит в использовании 13 предикторов, таких как `Age`, `Sex` и `Chol`, чтобы предсказать наличие заболевания сердца у того или иного человека. Теперь мы выясним, насколько хорошо метод SVM работает на этих данных, по сравнению с LDA. После удаления 6 пропущенных наблюдений мы получаем данные для 297 человек, которые мы далее разбиваем случайным образом на 207 обучающих и 90 контрольных наблюдений.

Сначала мы строим LDA–модель и классификатор на опорных векторах по обучающим данным. Заметьте, что классификатор на опорных векторах является эквивалентным SVM–классификатору с полиномиальным ядром 1–й степени. Слева на рис. 9.10 приведены ROC–кривые (см. подраздел 4.4.3) для предсказаний на обучающих данных, полученных при помощи LDA и классификатора на опорных векторах. Оба классификатора вычисляют $\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p$ для каждого наблюдения. Используя некоторое пороговое значение t, мы относим наблюдения к категориям «болезнь есть» или «болезни нет» в зависимости от того, какое из следующих двух условий выполняется: $\hat{f}(X) < t$ или $\hat{f}(X) \geq t$. ROC–кривую получают, формируя такие предсказания и вычисляя частоты встречаемости истинно положительных и ложноположительных случаев для некоторого интервала значений t. В случае с оптимальным классификатором ROC–кривая будет близко подходить к верхнему левому углу графика. В нашем примере и LDA, и классификатор на опорных векторах обеспечивают высокое качество предсказаний, хотя есть указания на то, что классификатор на опорных векторах работает несколько лучше.

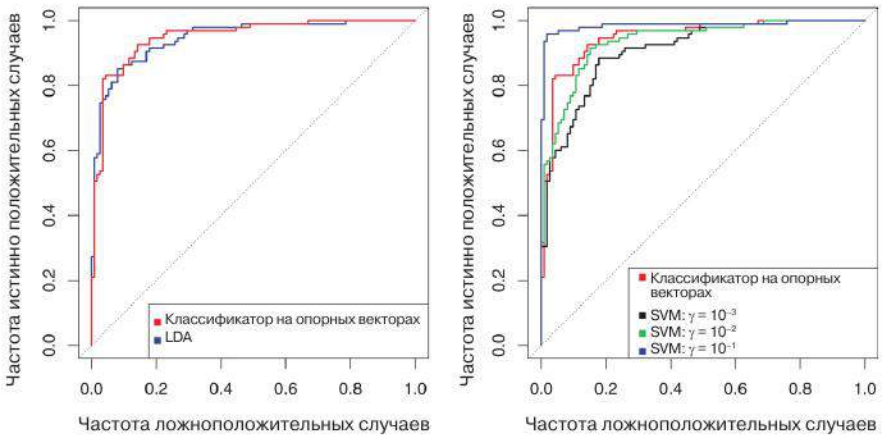

РИСУНОК 9.10. *ROC–кривые для набора обучающих данных* `Heart`. Слева: *сравниваются классификатор на опорных векторах и LDA–классификатор.* Справа: *классификатор на опорных векторах сравнивается с SVM с радиальным ядром и значениями* $\gamma = 10^{-3}$, 10^{-2} *и* 10^{-1}

Справа на рис. 9.10 показаны ROC–кривые SVM–классификаторов с радиальным ядром для ряда значений γ. По мере увеличения γ и степени гибкости модели, наблюдается улучшение ROC–кривых. При $\gamma = 10^{-1}$ ROC–кривая принимает почти идеальный вид. Однако эти кривые описывают частоты ошибок на обучающей выборке, которые могут создавать ложное представление о качестве предсказаний на новых, контрольных данных. На рис. 9.11 показаны ROC–кривые, вычисленные по 90 контрольным наблюдениям. Мы видим некоторые отличия от ROC–кривых, полученных по обучающим данным. Слева на рис. 9.11 классификатор на опорных векторах имеет небольшое преимущество, по сравнению с LDA (хотя это отличие не является статистически значимым). На графике справа SVM–классификатор с $\gamma = 10^{-1}$, который на обучающих данных сработал лучше всего, дает самые плохие оценки на контрольных данных. Этот пример еще раз показывает, что хотя более гибкие методы часто дают более низкие частоты ошибок на обучающей выборке, это не обязательно приводит к такому же высокому качеству на контрольной выборке. Качество SVM–классификаторов с $\gamma = 10^{-2}$ и $\gamma = 10^{-3}$ близко к качеству классификатора на опорных векторах, и при этом все эти три модели превосходят SVM с $\gamma = 10^{-1}$.

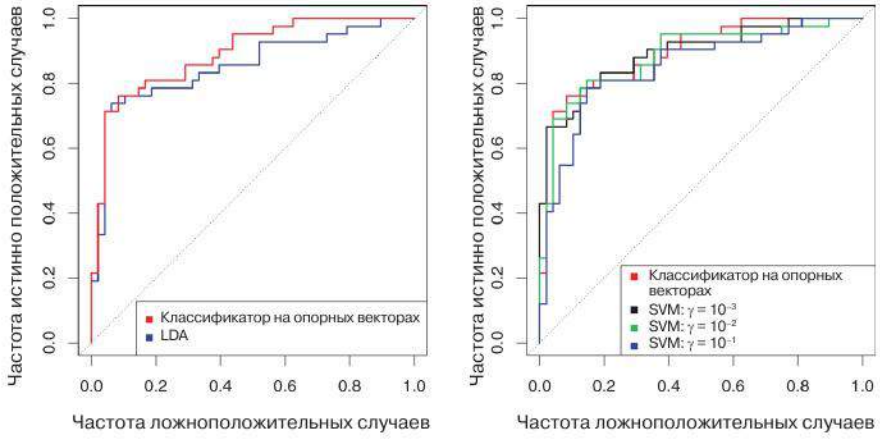

РИСУНОК 9.11. *ROC–кривые для набора контрольных данных* `Heart`. Слева: *сравниваются классификатор на опорных векторах и LDA–классификатор.* Справа: *классификатор на опорных векторах сравнивается с SVM с радиальным ядром и значениями* $\gamma = 10^{-3}$, 10^{-2} *и* 10^{-1}

9.4 Машины опорных векторов для случаев с несколькими классами

До сих пор мы обсуждали только случай бинарной классификации, т. е. сценарий, в котором есть два класса. Как можно расширить метод SVM на более общий случай, когда у нас есть некоторое произвольное число классов? Оказывается, что концепция разделяющих гиперплоскостей,

на которой основаны машины опорных векторов, естественным образом на случай с несколькими классами не переносится. Хотя было предложено несколько способов расширения метода SVM на случай с K классами, наиболее популярными являются подходы «один против одного» и «один против всех». Здесь мы кратко обсудим эти два подхода.

9.4.1 Классификация типа «один против одного»

один против одного

Представим, что мы хотели бы выполнить классификацию при помощи машин опорных векторов и у нас есть $K > 2$ классов. Метод «один против одного», или метод «всех пар», создает $\binom{K}{2}$ таких машин, каждая из которых сравнивает определенную пару классов. Например, одна такая машина опорных векторов могла бы сравнить k–й класс, закодированный как $+1$, с k'–м классом, закодированным как -1. Мы классифицируем то или иное контрольное наблюдение при помощи каждого из $\binom{K}{2}$ классификаторов, а затем подсчитываем, сколько раз это наблюдение было отнесено к каждому из K классов. Итоговая классификация выполняется путем отнесения контрольного наблюдения к классу, который чаще всего присваивался в этих $\binom{K}{2}$ парных сравнениях.

9.4.2 Классификация типа «один против всех»

один против всех

«Один против всех» является альтернативной процедурой для применения SVM к случаю с $K > 2$ классами. Мы строим K машин опорных векторов, каждый раз сравнивая один из K классов с остальными $K - 1$ классами. Пусть $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ обозначают параметры, получаемые в результате подгонки SVM–модели, которая сравнивает k–й класс (закодирован как $+1$) с остальными классами (закодированы как -1). Пусть также x^* обозначает контрольное наблюдение. Мы относим это наблюдение к классу, для которого величина $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \cdots + \beta_{pk}x_p^*$ является максимальной, поскольку это означает большую уверенность в том, что это контрольное наблюдение принадлежит к k–му, а не какому–то другому классу.

9.5 Связь с логистической регрессией

Когда машины опорных векторов впервые были представлены в середине 1990–х годов, они наделали довольно много шума в сообществах статистиков и специалистов по машинному обучению. Отчасти это было связано с хорошим качеством этого метода, его хорошей рекламой, а также с тем, что лежащий в его основе подход казался одновременно новым и таинственным. Идея нахождения гиперплоскости, которая разделяет данные максимально хорошо, позволяя при этом незначительные нарушения этого разделения, выглядела отличной от классических методов классификации, таких как логистическая регрессия и линейный дискриминантный анализ. Более того, идея использования ядерной функции для расширения пространства признаков с целью учета нелинейного характера разделяющих границ выглядела уникальной и ценной особенностью.

Однако со временем обнаружились тесные связи между SVM и другими, более традиционными методами статистики. Оказалось, что критерий (9.12)–(9.15) для подгонки классификатора на опорных векторах $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ можно переписать следующим образом:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{минимизировать}} \left\{ \sum_{i=1}^{n} \max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}, \tag{9.25}$$

где λ — это некоторый неотрицательный гиперпараметр. При большом λ коэффициенты $\beta_1, \ldots, \beta_p$ невелики, допускается большее количество нарушений границ зазора, и классификатор обладает низкой дисперсией и высоким смещением. При малом значении λ количество нарушений границ зазора будет небольшим, что приведет к высокой дисперсии и низкому смещению классификатора. Таким образом, небольшое значение λ в (9.25) аналогично небольшому значению C в (9.15). Заметьте, что $\lambda \sum_{j=1}^{p} \beta_j^2$ в (9.25) представляет собой штрафной член гребневой регрессии из подраздела 6.2.1 и выполняет похожую роль, т. е. контролирует баланс между дисперсией и смещением классификатора на опорных векторах.

А теперь (9.25) принимает форму «функция потерь + штраф», с которой мы уже неоднократно сталкивались в этой книге:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{минимизировать}} \left\{ L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta) \right\}. \tag{9.26}$$

$L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta)$ в (9.26) — это некоторая функция потерь, количественно выражающая степень соответствия модели с параметрами β данным $(\mathbf{X}, \mathbf{y})$, а $P(\beta)$ — это штрафная функция для вектора параметров β, чьи значения контролируются некоторым неотрицательным гиперпараметром λ. Например, и гребневая регрессия, и лассо–регрессия принимают эту форму с функцией потерь

$$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

и $P(\beta) = \sum_{j=1}^{p} \beta_j^2$ в случае гребневой регрессии и $P(\beta) = \sum_{j=1}^{p} |\beta_j|$ в случае лассо. В то же время в (9.25) функция потерь принимает форму

$$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^{n} \max[0, 1 - y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})].$$

Эта функция (изображена на рис. 9.12) известна как *кусочно–линейная функция потерь*[10]. Оказывается, однако, что кусочно–линейная функция потерь очень близка к функции потерь, используемой в логистической регрессии (также показана на рис. 9.12).

кусочно-линейная функция потерь

[10] В оригинале используется термин «hinge loss». — *Прим. пер.*

Интересным свойством классификатора на опорных векторах является то, что он полностью определяется опорными векторами, т. е. наблюдения, лежащие на правильной стороне относительно границы зазора, никакого влияния не оказывают. Это связано с тем, что изображенная на рис. 9.12 функция строго равна нулю для наблюдений, у которых $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq 1$, т. е. наблюдений, расположенных на правильной стороне относительно границы зазора[11]. В то же время показанная на рис. 9.12 функция потерь логистической регрессии нигде нулю не равна. Тем не менее она принимает очень низкие значения для наблюдений, лежащих далеко от решающей границы. Благодаря сходству между их функциями потерь логистическая регрессия и классификатор на опорных векторах часто дают очень похожие результаты. При хорошей разделимости классов машины опорных векторов обычно ведут себя лучше логистической регрессии, однако при более выраженном перекрытии классов часто более предпочтительной является логистическая регрессия.

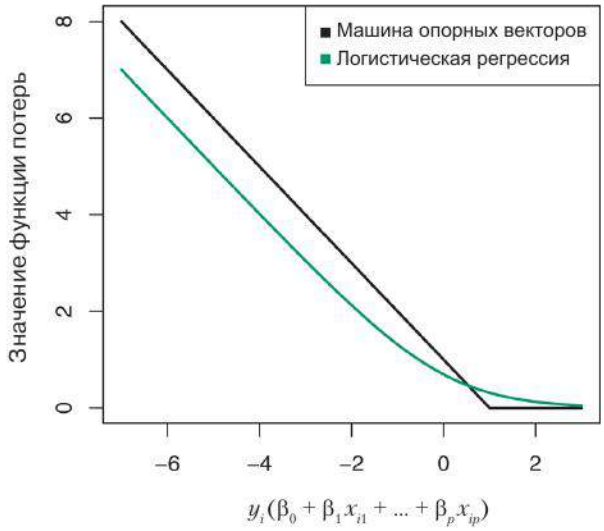

РИСУНОК 9.12. *Сравнение функций потерь SVM и логистической регрессии в зависимости от* $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})$. *При* $y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) > 1$ *функция потерь SVM равна нулю, что соответствует наблюдению, расположенному на правильной стороне относительно границы зазора. В целом обе функции потерь демонстрируют довольно похожее поведение*

Когда классификатор на опорных векторах и SVM были представлены впервые, считалось, что гиперпараметр C в (9.15) был неважным «мешающим» параметром, которому можно присвоить какое–нибудь произвольное значение по умолчанию, например 1. Однако формулировка классификатора на опорных векторах в (9.25) в виде «функция потерь

[11] Когда используется представление классификатора в виде «кусочно–линейная функция потерь + штраф», границе зазора соответствует значение 1, а ширина зазора равна $\sum \beta_j^2$.

+ штраф» показывает, что это мнение ошибочно. Выбор значения гиперпараметра очень важен и определяет степень переобучения модели на данных (см., например, рис. 9.7).

Мы установили, что классификатор на опорных векторах очень похож на логистическую регрессию и другие существовавшие ранее статистические методы. Является ли SVM уникальным методом в том, что он использует ядра для расширения пространства признаков для учета нелинейных границ между классами? Ответ на этот вопрос — «нет». Мы могли бы с тем же успехом применить логистическую регрессию или многие другие методы классификации, описанные в этой книге, используя нелинейные ядра; это сильно напоминало бы некоторые нелинейные методы из главы 7. Однако по историческим причинам использование нелинейных ядер гораздо более распространено в контексте SVM, а не логистической регрессии или других методов.

Хотя мы его здесь не обсуждали, но существует расширение метода SVM для задач регрессии (т. е. для случаев количественного, а не качественно отклика), которое называют *регрессией на основе опорных векторов*[12]. В главе 3 мы видели, что регрессия по методу наименьших квадратов пытается найти такие коэффициенты $\beta_0, \beta_1, \ldots, \beta_p$, которые минимизируют сумму квадратов остатков. (Вспомните из главы 3, что остатки определяются как $y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip}$.) Регрессия же на опорных векторах пытается минимизировать другую функцию потерь, значение которой определяется только такими остатками, чьи абсолютные значения превышают некоторую положительную константу. Это представляет собой расширение идеи зазора из классификатора на опорных векторах на случай регрессии.

регрессия на основе опорных векторов

9.6 Лабораторная работа: машины опорных векторов

Для демонстрации классификатора на опорных векторах и метода SVM мы воспользуемся библиотекой `e1071` для `R`. Альтернативой является также библиотека `LiblineaR`, которая полезна для построения линейных моделей по очень большим объемам данных.

9.6.1 Классификатор на опорных векторах

Библиотека `e1071` позволяет реализовать целый ряд методов статистического обучения. В частности, можно воспользоваться функцией `svm()` в сочетании с аргументом `kernel = "linear"` для построения классификатора на опорных векторах. Эта функция использует определение классификатора, несколько отличающееся от (9.14) и (9.25). Аргумент `cost` позволяет нам задать штраф за нарушение границ зазора. При малом значении аргумента `cost` зазор будет широким, и многие опорные векторы будут лежать на границах зазора или выходить за них. При большом значении аргумента `cost` зазор будет узким, и лишь немногие опорные векторы будут лежать на его границах или выходить за них.

[12] В оригинале используется термин «support vector regression». — *Прим. пер.*

Сейчас мы воспользуемся функцией `svm()` для построения классификатора на опорных векторах для некоторого фиксированного значения параметра `cost`. Мы продемонстрируем применение этой функции на примере двумерных данных, чтобы можно было графически изобразить итоговую решающую границу. Начнем с генерации наблюдений, принадлежащих к двум классам.

```
> set.seed(1)
> x = matrix(rnorm(20 * 2), ncol = 2)
> y = c(rep(-1, 10), rep(1, 10))
> x[y==1, ] = x[y==1, ] + 1
```

Сначала проверим, являются ли классы линейно разделимыми.

```
> plot(x, col = (3 - y))
```

Нет, это не так. Далее мы построим классификатор на опорных векторах. Обратите внимание, что для выполнения классификации (а не SVM–регрессии) при помощи функции `svm()` мы должны закодировать отклик как факторную переменную. Сейчас мы создадим таблицу с данными, в которой отклик представлен в виде фактора.

```
> dat = data.frame(x = x, y = as.factor(y))
> library(e1071)
> svmfit = svm(y ~ ., data = dat, kernel = "linear",
               cost = 10, scale = FALSE)
```

Аргумент `scale = FALSE` говорит функции `svm()` о том, что мы не хотим выполнять стандартизацию, в результате которой среднее значение каждой переменной станет равным 0, а стандартное отклонение — 1; однако в некоторых задачах мы могли бы предпочесть `scale = TRUE`.

Теперь мы изобразим полученный классификатор:

```
> plot(svmfit, dat)
```

Заметьте, что поданные на функцию `plot.svm()` два аргумента — это объект, полученный в результате применения `svm()`, а также таблица с данными, использованными при создании этого объекта. Область пространства признаков, которая будет отнесена к классу -1, показана светло–голубым цветом, а область, которая будет отнесена к классу $+1$, показана фиолетовым цветом. Хотя решающая граница между этими двумя классами является линейной (поскольку мы применили аргумент `kernel = "linear"`), на графике она выглядит зигзагообразно, что связано с особенностями реализации метода для построения графиков в этой библиотеке. Как видим, в этом случае неправильно классифицировано только одно наблюдение. (Заметьте, что, в отличие от поведения обычной функции `plot()` в `R`, вторая переменная здесь показана на оси X, а первая — на оси Y.) Опорные векторы изображены в виде крестиков, а остальные наблюдения показаны в виде кружков; как видим, имеется семь опорных векторов. Мы можем выяснить, какие это наблюдения, следующим образом:

```
> svmfit$index
[1]  1  2  5  7  14 16 17
```

При помощи команды `summary()` мы можем получить определенную общую информацию о построенном классификаторе на опорных векторах:

```
> summary(svmfit)
Call:
svm (formula = y ~ ., data = dat, kernel = "linear",
     cost = 10, scale = FALSE)
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.5
Number of Support Vectors:  7
( 4 3 )
Number of Classes:  2
Levels:
 -1 1
```

Это говорит нам, например, о том, что линейное ядро было использовано в сочетании с `cost = 10` и что имеются семь опорных векторов, четыре из которых принадлежат к одному классу, а три — к другому.

Что произойдет, если мы воспользуемся меньшим значением штрафного параметра?

```
> svmfit = svm(y ~ ., data = dat, kernel = "linear",
               cost =0.1, scale = FALSE)
> plot(svmfit, dat); svmfit$index
[1] 1  2  3  4  5  7  9  10 12 13 14 15 16 17 18 20
```

Применение меньшего значения штрафного параметра приводит к большему числу опорных векторов, поскольку зазор стал более широким. К сожалению, функция `svm()` не выдает ни самих оценок коэффициентов линейной решающей границы, полученной в результате построения классификатора, ни значения ширины зазора.

В состав библиотеки `e1071` входит встроенная функция `tune()`, предназначенная для выполнения перекрестной проверки. По умолчанию `tune()` выполняет десятикратную перекрестную проверку для набора из нескольких интересующих нас моделей. На вход этой функции подается соответствующая информация о рассматриваемых моделях. Следующая команда означает, что мы хотим сравнить машины опорных векторов с линейным ядром на определенном интервале значений параметра `cost`:

```
> set.seed(1)
> tune.out = tune(svm, y ~ ., data = dat, kernel = "linear",
  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
```

При помощи команды `summary()` мы можем легко получить значения ошибок, рассчитанные для каждой из этих моделей в ходе перекрестной проверки:

```
> summary(tune.out)
Parameter tuning of 'svm':
- sampling method: 10- fold cross validation
- best parameters:
 cost
  0.1
```

```
- best performance: 0.1
- Detailed performance results:
   cost error dispersion
1 1e-03  0.70      0.422
2 1e-02  0.70      0.422
3 1e-01  0.10      0.211
4 1e+00  0.15      0.242
5 5e+00  0.15      0.242
6 1e+01  0.15      0.242
7 1e+02  0.15      0.242
```

Как видим, `cost = 0.1` приводит к наименьшей частоте ошибок по результатам перекрестной проверки. Функция `tune()` сохраняет полученную оптимальную модель, доступ к которой можно получить следующим образом:

```
> bestmod = tune.out$best.model
> summary(bestmod)
```

Функция `predict()` позволяет предсказать класс контрольных наблюдений для любого значения штрафного параметра. Сначала сгенерируем контрольные данные.

```
> xtest = matrix(rnorm(20 * 2), ncol = 2)
> ytest = sample(c(-1, 1), 20, rep = TRUE)
> xtest[ytest == 1, ] = xtest[ytest == 1, ] + 1
> testdat = data.frame(x = xtest, y = as.factor(ytest))
```

Теперь мы предскажем метки классов для этих контрольных наблюдений. Для этого мы воспользуемся оптимальной моделью, полученной в ходе перекрестной проверки.

```
> ypred = predict (bestmod, testdat )
> table(predict = ypred, truth = testdat$y)
       truth
predict -1  1
     -1 11  1
      1  0  8
```

Таким образом, при этом значении `cost` правильно классифицированы 19 контрольных наблюдений. Что, если вместо этого мы применим `cost = 0.01`?

```
> svmfit = svm(y ~ ., data = dat, kernel = "linear",
               cost = .01, scale = FALSE)
> ypred = predict(svmfit, testdat)
> table(predict = ypred, truth = testdat$y)
       truth
predict -1  1
     -1 11  2
      1  0  7
```

В этом случае неверно классифицированным оказалось еще одно наблюдение.

Теперь мы рассмотрим ситуацию, в которой два класса являются линейно разделимыми. Далее мы сможем найти разделяющую гиперплоскость при помощи функции `svm()`. Сначала мы еще больше раздвинем два класса в нашем наборе имитированных данных таким образом, чтобы они стали линейно разделимыми:

```
> x[y==1, ] = x[y==1, ] + 0.5
> plot(x, col = (y + 5)/2, pch = 19)
```

Теперь наблюдения стали едва разделимыми с помощью линейной решающей границы. Построим классификатор на опорных векторах с очень большим значением `cost`, позволяющим правильно классифицировать все наблюдения, и изобразим полученную гиперплоскость.

```
> dat = data.frame(x = x, y = as.factor(y))
> svmfit = svm(y ~., data = dat, kernel = "linear",
              cost = 1e5)
> summary(svmfit)
Call:
svm(formula = y ~ ., data = dat, kernel = "linear",
    cost = 1e+05)
Parameters:
   SVM-Type: C-classification
 SVM-Kernel: linear
       cost: 1e+05
      gamma: 0.5
Number of Support Vectors:  3
( 1 2 )
Number of Classes:  2
Levels:
 -1 1
> plot(svmfit, dat)
```

Все обучающие наблюдения классифицированы правильно, и при этом были использованы только три опорных вектора. Однако на рисунке видно, что зазор очень узок (поскольку показанные в виде кружков наблюдения, не являющиеся опорными векторами, находятся очень близко к решающей границе). С большой вероятностью эта модель плохо сработает на контрольных данных. Попробуем теперь воспользоваться меньшим значением `cost`:

```
> svmfit = svm(y ~ ., data = dat, kernel = "linear", cost = 1)
> summary(svmfit)
> plot(svmfit, dat)
```

С `cost = 1` мы неверно классифицируем одно обучающее наблюдение, но при этом получаем намного более широкий зазор и используем семь опорных векторов. Скорее всего, эта модель сработает на контрольных данных лучше, чем модель с `cost = 10e5`.

9.6.2 Машина опорных векторов

Для построения SVM с нелинейным ядром мы снова воспользуемся функцией `svm()`. Однако теперь мы присвоим другое значение параметру

kernel. Для построения SVM с полиномиальным ядром мы применяем kernel = "polynomial", а для SVM с радиальным ядром — kernel = "radial". В первом случае мы применяем также аргумент degree для указания степени полиномиального ядра (d в (9.22)), а во втором случае мы применяем аргумент gamma для указания параметра γ ядра на основе радиальной базисной функции (9.24).

Сначала мы сгенерируем данные с нелинейной границей между классами:

```
> set.seed(1)
> x = matrix(rnorm(200 * 2), ncol = 2)
> x[1:100, ] = x[1:100, ] + 2
> x[101:150, ] = x[101:150, ] - 2
> y = c(rep(1, 150), rep(2, 50))
> dat = data.frame(x = x, y = as.factor(y))
```

Изображение данных четко показывает, что граница между классами действительно является нелинейной:

```
> plot(x, col = y)
```

Данные случайным образом разделяются на обучающую и контрольную выборки. Далее при помощи функции svm() мы строим модель с радиальным ядром и $\gamma = 1$ по обучающим данным:

```
> train = sample(200, 100)
> svmfit = svm(y ~ ., data = dat[train, ], kernel = "radial",
               gamma = 1, cost = 1)
> plot(svmfit, dat[train, ])
```

Как видно на графике, полученная машина опорных векторов имеет выраженную нелинейную решающую границу. Можно воспользоваться функцией summary() и получить некоторую информацию об этой модели:

```
> summary(svmfit)
Call:
svm(formula = y ~ ., data = dat, kernel = "radial",
    gamma = 1, cost = 1)
Parameters:
   SVM-Type: C-classification
 SVM-Kernel: radial
       cost: 1
      gamma: 1
Number of Support Vectors: 37
 ( 17 20 )
Number of Classes:  2
Levels:
 1 2
```

Как видим, эта SVM–модель совершает довольно заметное число ошибок на обучающих данных. Увеличив значение cost(), мы можем снизить это число. Однако из–за этого будет получена более извилистая решающая граница, которая, по всей видимости, приведет к переобучению.

```
> svmfit = svm(y ~., data = dat[train, ], kernel = "radial",
               gamma = 1, cost = 1e5)
> plot(svmfit, dat[train, ])
```

При помощи `tune()` мы можем выполнить перекрестную проверку для нахождения оптимальных значений γ и `cost` для SVM с радиальным ядром:

```
> set.seed(1)
> tune.out = tune(svm, y ~., data=dat[train, ], kernel="radial",
                  ranges = list(cost = c(0.1, 1, 10, 100, 1000),
                                gamma = c(0.5, 1, 2, 3, 4)))
> summary(tune.out)
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 cost gamma
    1     2
- best performance: 0.12
- Detailed performance results :
    cost gamma error dispersion
1  1e-01   0.5  0.27     0.1160
2  1e+00   0.5  0.13     0.0823
3  1e+01   0.5  0.15     0.0707
4  1e+02   0.5  0.17     0.0823
5  1e+03   0.5  0.21     0.0994
6  1e-01   1.0  0.25     0.1354
7  1e+00   1.0  0.13     0.0823
. . .
```

Таким образом, оптимальные параметры составляют `cost = 1` и `gamma = 2`. Применив функцию `predict()`, мы можем ознакомиться с предсказаниями этой модели для контрольных данных. Заметьте, что для этого мы извлекаем часть таблицы `dat`, используя `-train` в качестве вектора с соответствующими индексными номерами:

```
> table(true = dat[-train, "y"],
        pred = predict(tune.out$best.model,
        newdata = dat[-train, ]))
```

Эта машина опорных векторов неверно классифицирует 10% контрольных наблюдений.

9.6.3 ROC–кривые

Для построения ROC–кривых, подобных приведенным на рис. 9.10 и 9.11, можно воспользоваться пакетом `ROCR`. Сначала мы напишем небольшую функцию для расчета ROC–кривой на основе вектора `pred` со значениями классификационной функции для всех наблюдений, а также вектора `truth` с истинными метками классов.

```
> library(ROCR)
> rocplot = function(pred, truth, ...) {
+ predob = prediction(pred, truth)
```

```
+ perf = performance(predob, "tpr", "fpr")
+ plot(perf, ...)
+ }
```

Для каждого наблюдения машины опорных векторов и классификаторы на опорных векторах выдают соответствующую метку класса. Однако имеется возможность получить также *модельные значения* для каждого наблюдения, которые представляют собой количественные величины, используемые для присвоения классовых меток. Например, в случае с классификатором на основе опорных векторов такое модельное значение для наблюдения $X = (X_1, X_2, \ldots, X_p)^T$ принимает форму $\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p$. Для SVM с нелинейным ядром модельное значение получают по формуле (9.23). По сути, знак этой величины определяет то, по какую сторону относительно решающей границы лежит соответствующее наблюдение. Следовательно, связь между модельным значением и предсказанным классом того или иного наблюдения проста: если модельное значение превышает 0, то это наблюдение относят к одному классу, а если оно оказывается меньше 0, то наблюдение относят к другому классу. Для получения модельных значений машины опорных векторов в ходе ее построения с помощью `svm()` мы указываем `decision.values = TRUE`.

```
> svmfit.opt = svm(y ~ ., data=dat[train, ], kernel="radial",
                   gamma=2, cost=1, decision.values = TRUE)
> fitted = attributes(predict(svmfit.opt, dat[train, ],
                      decision.values = TRUE))$decision.values
```

Теперь мы можем построить график ROC–кривой.

```
> par(mfrow = c(1, 2))
> rocplot(fitted, dat[train, "y"], main = "Training Data")
```

Похоже, что SVM дает точные предсказания. Увеличивая γ, мы можем построить более гибкую модель и повысить точность предсказаний еще больше.

```
> svmfit.flex = svm(y ~ ., data=dat[train, ], kernel="radial",
                    gamma=50, cost=1, decision.values=TRUE)
> fitted = attributes(predict(svmfit.flex, dat[train, ],
                      decision.values = TRUE))$decision.values
> rocplot(fitted, dat[train, "y"], add = TRUE, col = "red")
```

Однако все эти ROC–кривые относятся к обучающей выборке. В действительности же мы более заинтересованы в точных предсказаниях на контрольной выборке. Результаты вычисления ROC–кривых на контрольной выборке указывают на то, что наиболее точные результаты дает модель с $\gamma = 2$.

```
> fitted = attributes(predict(svmfit.opt, dat[-train, ],
                      decision.values = TRUE))$decision.values
> rocplot(fitted, dat[-train, "y"], main = "Test Data")
> fitted = attributes(predict (svmfit.flex, dat[-train, ],
                      decision.values = TRUE))$decision.values
> rocplot(fitted, dat[-train, "y"], add = TRUE, col = "red")
```

9.6.4 SVM с несколькими классами

Если отклик представлен фактором, включающим более двух уровней, то функция `svm()` выполнит многоклассовую классификацию с использованием подхода «один против одного». Мы рассмотрим такой сценарий, создав третий класс наблюдений.

```
> set.seed(1)
> x = rbind(x, matrix(rnorm(50 * 2), ncol = 2))
> y = c(y, rep(0, 50))
> x[y==0, 2] = x[y==0, 2] + 2
> dat = data.frame(x = x, y = as.factor(y))
> par(mfrow = c(1, 1))
> plot(x, col = (y + 1))
```

Теперь мы подгоним SVM–модель к этим данным:

```
> svmfit = svm(y ~ ., data = dat, kernel = "radial",
               cost = 10, gamma = 1)
> plot(svmfit, dat)
```

Когда подаваемый на функцию `svm()` вектор со значениями отклика является количественной, а не качественной переменной, библиотеку `e1071` можно использовать также для выполнения регрессии на опорных векторах.

9.6.5 Применение к данным по экспрессии генов

Теперь мы рассмотрим набор данных `Khan`, который включает целый ряд образцов тканей четырех отдельных типов саркомы[13]. Для каждого образца тканей доступны результаты измерения экспрессии генов. Этот набор включает обучающие данные — `xtrain` и `ytrain` и контрольные данные — `xtest` и `ytest`.

Выясним размерность данных:

```
> library(ISLR)
> names(Khan)
[1] "xtrain" "xtest" "ytrain" "ytest"
> dim(Khan$xtrain)
[1] 63 2308
> dim(Khan$xtest)
[1] 20 2308
> length(Khan$ytrain)
[1] 63
> length(Khan$ytest)
[1] 20
```

Этот набор данных состоит из измерений уровня экспрессии 2308 генов. Обучающая и контрольная выборки содержат 63 и 20 наблюдений соответственно.

```
> table(Khan$ytrain)
1  2  3  4
```

[13] Раковое заболевание. — *Прим. пер.*

```
8 23 12 20
> table(Khan$ytest)
1 2 3 4
3 6 6 5
```

Мы применим классификатор на опорных векторах для предсказания типа рака по уровню экспрессии генов. Количество переменных здесь намного превышает количество наблюдений. Это предполагает, что нам следует воспользоваться линейным ядром, поскольку здесь нет необходимости в гибкости, добавляемой классификатору за счет полиномиальных или радиальных ядер.

```
> dat = data.frame(x = Khan$xtrain, y = as.factor(Khan$ytrain))
> out = svm(y ~ ., data = dat, kernel = "linear", cost = 10)
> summary(out)

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10)
Parameters:
   SVM-Type: C-classification
 SVM-Kernel: linear
       cost: 10
      gamma: 0.000433
Number of Support Vectors: 58
 ( 20 20 11 7 )
Number of Classes: 4
Levels:
 1 2 3 4
> table(out$fitted, dat$y)
    1  2  3  4
 1  8  0  0  0
 2  0 23  0  0
 3  0  0 12  0
 4  0  0  0 20
```

Мы видим полное *отсутствие* неправильно классифицированных обучающих наблюдений. На самом деле это неудивительно, поскольку в случае, когда число переменных значительно превышает число наблюдений, можно легко найти гиперплоскость, разделяющую классы идеально. Однако больше всего мы заинтересованы в качестве классификатора не на обучающих, а на контрольных данных.

```
> dat.te = data.frame(x = Khan$xtest, y = as.factor(Khan$ytest))
> pred.te = predict(out, newdata = dat.te)
> table(pred.te, dat.te$y)
pred.te  1  2  3  4
      1  3  0  0  0
      2  0  6  2  0
      3  0  0  4  0
      4  0  0  0  5
```

Как видим, использование `cost = 10` приводит к возникновению двух ошибок на контрольных данных.

9.7 Упражнения

Теоретические

1. Эта задача посвящена гиперплоскостям в двумерном пространстве.

   (a) Изобразите набросок гиперплоскости $1 + 3X_1 - X_2 = 0$. Укажите подмножество точек, для которых $1 + 3X_1 - X_2 > 0$, а также подмножество точек, для которых $1 + 3X_1 - X_2 < 0$.

   (b) На том же графике изобразите набросок гиперплоскости $-2 + X_1 + 2X_2 = 0$. Укажите подмножество точек, для которых $-2 + X_1 + 2X_2 > 0$, а также подмножество точек, для которых $-2 + X_1 + 2X_2 < 0$.

2. Мы видели, что в двумерном пространстве предикторов линейная решающая граница принимает форму $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$. Теперь мы исследуем нелинейную решающую границу.

   (a) Изобразите набросок кривой

   $$(1 + X_1)^2 + (2 - X_2)^2 = 4.$$

   (b) Укажите на своем наброске подмножество точек, для которых

   $$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$

   а также подмножество точек, для которых

   $$(1 + X_1)^2 + (2 - X_2)^2 \leq 4.$$

   (c) Допустим, что классификатор относит некоторое наблюдение к «голубому классу», если

   $$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$

   и к «красному классу» во всех остальных случаях. К какому классу будут отнесены наблюдения (0, 0), (2, 2), (3, 8)?

   (d) Приведите аргументы в пользу того, что, несмотря на нелинейный характер решающей границы из (c) по X_1 и X_2, она линейна по X_1, X_1^2, X_2 и X_2^2.

3. Здесь мы исследуем классификатор с максимальным зазором на небольшом гипотетическом наборе данных.

   (a) У нас есть $n = 7$ наблюдений в двумерном пространстве. У каждого наблюдения есть метка класса, к которому оно принадлежит.

Наблюдение	X_1	X_2	Y
1	3	4	красный
2	2	2	красный
3	4	4	красный
4	1	4	красный
5	2	1	голубой
6	4	3	голубой
7	4	1	голубой

Нарисуйте набросок графика с этими наблюдениями.

(b) Изобразите оптимальную разделяющую гиперплоскость и приведите ее уравнение, используя (9.1) в качестве примера.

(c) Опишите решающее правило для этого классификатора с максимальным зазором. Оно должно быть похоже на следующий пример: «Отнести к красному классу, если $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$, и к голубому классу в остальных случаях». Приведите значения β_0, β_1 и β_2.

(d) Покажите на своем наброске границы зазора разделяющей гиперплоскости.

(e) Покажите опорные векторы этого классификатора с максимальным зазором.

(f) Приведите аргументы в пользу того, что незначительный сдвиг седьмого наблюдения не оказал бы влияния на разделяющую плоскость этого классификатора.

(g) Нарисуйте набросок гиперплоскости, которая не является оптимальной разделяющей гиперплоскостью, и приведите ее уравнение.

(h) Нарисуйте на графике такое дополнительное наблюдение, которое не позволило бы разделить два класса с помощью гиперплоскости.

Практические

4. Создайте искусственный набор данных с двумя классами, содержащий 100 наблюдений и два признака, в котором имеет место выраженная нелинейная разделяющая граница между классами. Покажите, что при таком сценарии машина опорных векторов с полиномиальным ядром (со степенью выше 1) или радиальным ядром даст более точные предсказания на обучающих данных, чем классификатор на опорных векторах. Какой из этих методов сработает лучше всего на контрольной выборке? Постройте графики и приведите частоты ошибок на обучающей и контрольной выборках для подтверждения своих ответов.

5. Мы видели, что для выполнения классификации с помощью нелинейной решающей границы можно построить SVM с нелинейным ядром. Теперь мы увидим, что нелинейную решающую границу можно получить также путем построения логистической регрессионной модели, включающей нелинейно преобразованные предикторы.

(a) Сгенерируйте набор данных с $n = 500$ и $p = 2$ таким образом, чтобы наблюдения принадлежали к двум классам, разделенным квадратичной решающей границей. Вы можете сделать это, например, следующим образом:

```
> x1 = runif(500) - 0.5
> x2 = runif(500) - 0.5
> y = 1*(x1^2 - x2^2 > 0)
```

(b) Изобразите эти наблюдения на графике, обозначив их разными цветами в зависимости от класса, к которому они принадлежат. По оси X на вашем графике должны быть отложены значения переменной X_1, а по оси Y — значения переменной X_2.

(c) Постройте логистическую регрессионную модель, используя X_1 и X_2 в качестве предикторов.

(d) Примените эту модель к *обучающей выборке* для предсказания класса каждого наблюдения. Изобразите наблюдения на графике, используя разные цвета в зависимости от *предсказанного* класса. Решающая граница должна быть линейной.

(e) Теперь подгоните к этим данным логистическую регрессионную модель, используя в качестве предикторов нелинейные функции X_1 и X_2 (например, $X_1^2, X_1 \times X_2, \log(X_2)$ и т. п.).

(f) Примените эту модель к обучающим данным для предсказания класса каждого наблюдения. Изобразите наблюдения на графике, используя разные цвета в зависимости от *предсказанного* класса. Должно быть хорошо видно, что решающая граница является нелинейной. Если это не так, то повторите (a)–(e) несколько раз до тех пор, пока вы не получите пример, в котором классы явно разделены нелинейной границей.

(g) Постройте по этим данным классификатор на основе опорных векторов, используя X_1 и X_2 в качестве предикторов. Получите предсказания класса для каждого обучающего наблюдения. Изобразите наблюдения на графике, используя разные цвета в зависимости от *предсказанного класса.*

(h) Постройте по этим данным SVM с каким–либо нелинейным ядром. Получите предсказания класса для каждого обучающего наблюдения. Изобразите наблюдения на графике, используя разные цвета в зависимости от *предсказанного класса.*

(i) Прокомментируйте свои результаты.

6. В конце подраздела 9.6.1 было сделано утверждение о том, что при работе с данными, которые трудно разделить линейной решающей границей, классификатор на опорных векторах с малым значением параметра `cost`, совершающий ошибки в отношении нескольких обучающих наблюдений, может сработать на контрольной выборке лучшем, чем классификатор с большим значением `cost`, не совершающий ошибок на обучающей выборке. Теперь вы ознакомитесь с этим утверждением подробнее.

(a) Создайте данные с двумя классами и $p = 2$ таким образом, чтобы классы были едва разделимы линейной границей.

(b) Вычислите частоты ошибок методом перекрестной проверки для классификатора опорных векторов с разными значениями `cost`. Сколько обучающих наблюдений классифицированы неверно при каждом из рассмотренных значений `cost` и как это соотносится с частотами ошибок, полученными по результатам перекрестной проверки?

(c) Сгенерируйте подходящую контрольную выборку и вычислите частоты ошибок для каждого из рассмотренных значений `cost`. Какое значение `cost` дает наименьшее количество ошибок на контрольной выборке, и отличается ли оно от значений `cost`, обеспечивающих наименьшее количество ошибок на обучающей выборке и наименьшее количество ошибок по результатам перекрестной проверки?

(d) Обсудите свои результаты.

7. В этой задаче вы примените методы, основанные на опорных векторах, для предсказания уровня расхода топлива автомобилем (высокий или низкий) по данным из таблицы `Auto`.

(a) Создайте бинарную переменную, принимающую значение 1 для автомобилей, у которых расход топлива превышает соответствующее медианное значение, и 0 для автомобилей, у которых расход топлива меньше этого медианного значения.

(b) Постройте по этим данным классификатор на опорных векторах с разными значениями `cost` для предсказания уровня расхода топлива (низкий или высокий). Приведите частоты ошибок, полученные в ходе перекрестной проверки при разных значениях этого параметра. Прокомментируйте свои результаты.

(c) Теперь повторите (b), используя SVM с радиальным и полиномиальным ядрами для нескольких значений параметров `gamma`, `degree` и `cost`. Прокомментируйте свои результаты.

(d) Постройте графики, подтверждающие ваши ответы для пунктов (b) и (c).

Подсказка: в лабораторной работе мы использовали функцию `plot()` *для объектов класса* `svm` *только для случаев с* $p = 2$. *При* $p > 2$ *вы можете применить функцию* `plot()` *для создания графиков, каждый из которых изображает определенную пару переменных. По сути, вместо выполнения команды*

```
> plot(svmfit, dat)
```

в которой `svmfit` *содержит построенную вами модель, а* `dat` *— это таблица с вашими данными, вы можете выполнить*

```
> plot(svmfit, dat, x1 ~ x4)
```

для изображения только первой и четвертой переменных. Однако вы должны заменить `x1` *и* `x4` *на правильные имена переменных. Для получения дополнительной информации выполните команду* `?plot.svm`.

8. Эта задача касается данных `OJ`, входящих в состав пакета `ISLR`.

   (a) Создайте обучающую выборку, содержащую 800 случайно выбранных наблюдений, и контрольную выборку, содержащую все остальные наблюдения.

   (b) Постройте классификатор на опорных векторах по обучающим данным с `cost = 0.01`, используя `Purchase` в качестве отклика, а все остальные переменные в качестве предикторов. Примените функцию `summary()` для вывода сводной информации по модели и опишите полученные результаты.

   (c) Чему равны частоты ошибок на обучающих и контрольных данных?

   (d) Примените функцию `tune()` для выбора оптимального значения `cost`. Рассмотрите значения, лежащие в интервале от 0.01 до 10.

   (e) Вычислите частоты ошибок на обучающих и контрольных данных, используя это новое значение `cost`.

   (f) Повторите пункты (b)–(e) для машины опорных векторов с радиальным ядром. Используйте значение `gamma`, принятое по умолчанию.

   (g) Повторите пункты (b)–(e) для машины опорных векторов с полиномиальным ядром. Используйте значение `degree = 2`.

   (h) Говоря в целом, какой из этих методов дает оптимальные результаты на этих данных?

Глава 10

Обучение без учителя

Большая часть этой книги посвящена методам *обучения с учителем*, таким как методы регрессии и классификации. В случае обучения с учителем у нас обычно есть p признаков $X_1, X_2, \ldots, X_p$, измеренных у n объектов, и некоторый отклик Y, измеренный у тех же объектов. Задача заключается в предсказании Y по $X_1, X_2, \ldots, X_p$.

Эта глава будет посвящена *обучению без учителя* — совокупности статистических инструментов, предназначенных для случаев, когда у нас есть только некоторый набор признаков $X_1, X_2, \ldots, X_p$, измеренных у n объектов. Нас не интересует предсказание, поскольку у нас нет соответствующей зависимой переменной Y. Вместо этого задача заключается в обнаружении интересных аспектов, касающихся измеренных значений $X_1, X_2, \ldots, X_p$. Есть ли какой–то информативный способ визуализации этих данных? Можем ли мы обнаружить подгруппы среди переменных или наблюдений? Под «обучением без учителя» понимают широкий набор методов, позволяющих получить ответы на подобные вопросы. В этой главе мы сосредоточимся на двух конкретных типах обучения без учителя: *анализе главных компонент* — инструменте, позволяющем визуализировать данные и сделать их пригодными для использования в обучении с учителем, — а также *кластеризации* — обширном классе методов, предназначенных для обнаружения неизвестных групп в данных.

10.1 Трудность обучения без учителя

Обучение с учителем является хорошо исследованной областью. Более того, если вы читали предыдущие главы этой книги, то на данный момент у вас уже должно было сформироваться его понимание на приличном уровне. Например, если вам поставили задачу предсказать бинарный отклик из некоторого набора данных, в вашем распоряжении есть набор очень хорошо разработанных инструментов (например, логистическая регрессия, линейный дискриминантный анализ, деревья классификации, машины опорных векторов и др.), а также четкое понимание того, как оценить качество полученных результатов (используя перекрестную проверку, проверку на контрольной выборке и т. д.).

Однако обучение без учителя часто является намного более трудной задачей. Выполнение подобного анализа, как правило, имеет субъективный характер и лишено простой и понятной цели, такой как предсказание

некоторого отклика. Обучение без учителя часто выполняется как часть *разведочного анализа данных*. Более того, соответствующие результаты бывает трудно оценить, поскольку не существует широко принятых механизмов для выполнения перекрестной проверки или проверки результатов на независимых данных. Причина этого отличия очень проста. Когда мы строим предсказательную модель при помощи некоторого метода обучения с учителем, у нас есть возможность *проверить нашу работу*, выяснив, насколько хорошо модель предсказывает отклик Y на наблюдениях, которые не участвовали в построении этой модели. Однако в случае обучения без учителя способов проверки качества нашей работы не существует, поскольку мы не знаем правильного ответа — модель обучается без соответствующих примеров.

разведочный анализ данных

Методы обучения без учителя приобретают все большее значение в целом ряде областей. Исследователь, изучающий раковые заболевания, мог бы измерить уровни экспрессии генов у 100 пациентов с раком легких. Он или она могли бы затем попытаться выявить группы среди образцов раковых тканей или среди генов, чтобы лучше понять болезнь. Интернет–магазин мог бы попытаться обнаружить группы покупателей с похожими историями посещений сайта и покупок, а также выявить товары, представляющие особый интерес для покупателей из каждой группы. Затем, зная истории покупок похожих покупателей, некоторому новому покупателю можно показывать товары, которые с большой вероятностью будут ему интересны. Поисковая машина могла бы выбрать результаты, подлежащие показу тому или иному человеку, на основе истории посещений сайтов другими людьми, которые выполняли похожие поисковые запросы. Подобные (и многие другие) задачи статистического обучения можно решить при помощи методов обучения без учителя.

10.2 Анализ главных компонент

Главные компоненты обсуждаются в подразделе 6.3.1 в контексте регрессии на главные компоненты. При работе с большим набором коррелирующих переменных главные компоненты позволяют нам обобщить информацию из этого набора при помощи меньшего числа репрезентативных переменных, которые в сумме объясняют большую часть дисперсии, заключенной в исходных переменных. Направления главных компонент представлены в подразделе 6.3.1 как направления в пространстве признаков, вдоль которых исходные данные обладают *высокой дисперсией*. Эти направления определяют также линии и подпространства, которые находятся *максимально близко* к облаку исходных точек. Чтобы выполнить регрессию на главные компоненты, мы просто используем главные компоненты в качестве предикторов в регрессионной модели, заменяя ими большой набор исходных переменных.

Под *анализом главных компонент* (PCA)[1] понимают процесс вычисления главных компонент и последующее их использование для понимания структуры данных. PCA представляет собой метод обучения без учителя, поскольку он основан только на наборе признаков $X_1, X_2, \ldots, X_p$ и не подразумевает наличия соответствующей зависимой переменной Y. По-

анализ главных компонент

[1] Аббревиатура от «principal component analysis». — *Прим. пер.*

мимо создания производных переменных, пригодных для использования в обучении с учителем, PCA служит также в качестве инструмента визуализации данных (графического представления наблюдений или переменных). Сейчас мы обсудим PCA более подробно и в соответствии с темой данной главы уделим особое внимание применению PCA в качестве метода обнаружения закономерностей в данных «без учителя».

10.2.1 Что представляют собой главные компоненты?

Предположим, что при выполнении разведочного анализа данных мы хотим представить графически n наблюдений, каждое из которых описано по p признакам $X_1, X_2, \ldots, X_p$. Мы могли бы сделать это, изучив двумерные диаграммы рассеяния, каждая из которых содержит результаты измерений двух признаков у n наблюдений. Однако имеется $\binom{p}{2} = p(p-1)/2$ таких диаграмм; например, при $p = 10$ будет 45 графиков! При большом p визуально исследовать все эти графики будет практически невозможно; более того, ни один из них, скорее всего, не будет полезным, поскольку они содержат лишь небольшую долю общей информации, заключенной в данных. Ясно, что необходим другой метод для визуализации n наблюдений при большом p. В частности, мы хотели бы найти представление данных в пространстве малой размерности, содержащем максимально возможный объем информации. Например, если мы сможем получить двумерное представление данных, которое отражает большую часть исходной информации, то сможем и визуализировать наблюдения в этом двумерном пространстве.

PCA как раз и является инструментом, который позволяет это сделать. Он находит представление данных в пространстве малой размерности, которое содержит максимально возможное количество исходной информации. Идея заключается в том, что каждое из n наблюдений находится в p–мерном пространстве, однако не все оси этого пространства одинаково интересны. PCA находит небольшое число максимально интересных осей; при этом уровень *интереса* выражается величиной разброса наблюдений вдоль каждой оси. Каждая новая ось, найденная PCA, является линейной комбинацией p переменных. Ниже мы объясним способ, при помощи которого находят эти новые оси, или *главные компоненты*.

Первая главная компонента набора переменных $X_1, X_2, \ldots, X_p$ представляет собой такую нормализованную линейную комбинацию этих переменных

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p, \tag{10.1}$$

которая обладает максимальной дисперсией. Под *нормализацией* мы имеем в виду то обстоятельство, что $\sum_{j=1}^{p} \phi_{j1}^2 = 1$. Элементы $\phi_{11}, \ldots, \phi_{p1}$ мы называем *нагрузками* первой главной компоненты; все вместе они образуют вектор нагрузок главной компоненты $\phi_1 = (\phi_{11}\ \phi_{21}\ \ldots \phi_{p1})^T$. Мы ограничиваем нагрузки таким образом, чтобы сумма их квадратов была равна 1, поскольку, не сделав этого и присвоив нагрузкам произвольно большие абсолютные значения, мы могли бы получить произвольно большую дисперсию.

нагрузка компоненты

Как нам вычислить первую главную компоненту для некоторой матрицы данных $\mathbf{X}$ размером $n \times p$? Поскольку нам интересна только дисперсия,

мы делаем предположение о том, что каждая переменная в $\mathbf{X}$ центрирована относительно нуля (т. е. средние значения всех столбцов в $\mathbf{X}$ равны нулю). Далее мы пытаемся найти такую линейную комбинацию значений переменных из этой выборки

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}, \tag{10.2}$$

которая обладает максимальной дисперсией при условии, что $\sum_{j=1}^{p} \phi_{j1}^2 = 1$. Другими словами, вектор нагрузок первой главной компоненты решает следующую оптимизационную проблему:

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{максимизировать}} \left\{ \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 \right\} \text{ при условии, что } \sum_{j=1}^{p}\phi_{j1}^2 = 1. \tag{10.3}$$

С учетом (10.2) мы можем переписать целевую функцию из (10.3) как $\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$. Поскольку $\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$, то среднее значение $z_{11}, \ldots, z_{n1}$ также будет равно нулю. Следовательно, целевая функция, которую мы максимизируем в (10.3), есть не что иное, как выборочная дисперсия n значений z_{i1}. Мы называем $z_{11}, \ldots, z_{n1}$ *значениями* первой главной компоненты. Проблему (10.3) можно решить путем отыскания вектора собственных чисел — стандартного метода линейной алгебры, рассмотрение деталей которого, однако, выходит за рамки данной книги.

значение компоненты

Существует элегантная геометрическая интерпретация первой главной компоненты. Вектор нагрузок ϕ_1 с элементами $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ определяет направление в пространстве признаков, вдоль которого данные демонстрируют наибольший разброс. Если мы спроецируем n точек на это направление, то полученные значения будут представлять собой значения главной компоненты $z_{11}, \ldots, z_{n1}$. Например, рис. 6.14 на стр. 252 иллюстрирует вектор факторных нагрузок первой главной компоненты (показана в виде сплошной зеленой линии) для данных по рекламе. В этом наборе данных есть только две переменные, благодаря чему на графике можно легко показать как исходные наблюдения, так и вектор факторных нагрузок. Как следует из (6.19), в этом наборе данных $\phi_{11} = 0.839$, а $\phi_{21} = 0.544$.

После определения первой главной компоненты признаков Z_1 мы можем найти вторую главную компоненту Z_2. Вторая главная компонента представляет собой линейную комбинацию $X_1, \ldots, X_p$, которая обладает наибольшей дисперсией среди всех линейных комбинаций, *не коррелирующих* с Z_1. Значения $z_{12}, z_{22}, \ldots, z_{n2}$ второй главной компоненты имеют форму

$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \cdots + \phi_{p2}x_{ip}, \tag{10.4}$$

где ϕ_2 — это вектор нагрузок второй главной компоненты, состоящий из элементов $\phi_{12}, \phi_{22}, \ldots, \phi_{p2}$. Оказывается, что ограничение на отсутствие корреляции между Z_2 и Z_1 эквивалентно ограничению, согласно которому направление ϕ_2 должно быть ортогональным (перпендикулярным) направлению ϕ_1. В примере на рис. 6.14 наблюдения лежат в двумерном

пространстве (поскольку $p = 2$), и поэтому после нахождения ϕ_1 у нас будет только одна возможность в отношении направления вектора ϕ_2 (показан прерывистой линией). (Из подраздела 6.3.1 мы знаем, что $\phi_{12} = 0.544$, а $\phi_{22} = -0.839$.) Однако в случае с данными, где $p > 2$, имеется несколько главных компонент, и их находят аналогичным образом. Для нахождения ϕ_2 мы решаем проблему вида (10.3), заменяя ϕ_1 на ϕ_2 и добавляя дополнительное ограничение на то, чтобы вектор ϕ_2 был ортогональным вектору ϕ_1[2].

Вычислив главные компоненты, мы можем построить графики их парных зависимостей для представления данных в пространстве меньшей размерности. Например, мы можем изобразить зависимость вектора Z_1 от Z_2, Z_1 от Z_3, Z_2 от Z_3 и т. д. С геометрической точки зрения, это сводится к проецированию исходных данных на подпространство, образованное ϕ_1, ϕ_2 и ϕ_3, и графическому представлению спроецированных точек.

Мы продемонстрируем применение PCA к набору данных `USArrest`. Для каждого из 50 штатов США в этом наборе имеются значения числа арестов на 100 000 населения для трех типов преступлений: `Assault`, `Murder` и `Rape`. Мы также учитываем `UrbanPop` (процент городского населения в каждом штате). Векторы значений главных компонент имеют размер $n = 50$, а векторы нагрузок главных компонент имеют размер $p = 4$. PCA был выполнен после стандартизации, в результате которой среднее значение каждой переменной стало равно 0, а стандартное отклонение — 1. На рис. 10.1 показаны первые две главные компоненты этих данных. Здесь на одном графике типа «*biplot*»[3] одновременно представлены как значения главных компонент, так и факторные нагрузки. Факторные нагрузки приведены также в табл. 10.1.

biplot

ТАБЛИЦА 10.1. *Векторы нагрузок главных компонент ϕ_1 и ϕ_2 для данных* `USArrest`. *См. также рис. 10.1*

	PC1	PC2
`Murder`	0.5358995	−0.4181809
`Assault`	0.5831836	−0.1879856
`UrbanPop`	0.2781909	0.8728062
`Rape`	0.5434321	0.1673186

Как видно на рис. 10.1, первый вектор нагрузок присваивает примерно одинаковые веса переменным `Assault`, `Murder` и `Rape` и значительно более низкий вес переменной `UrbanPop`. Таким образом, эта компонента примерно описывает общий уровень преступности. Второй вектор нагрузок присваивает основной вес переменной `UrbanPop` и намного меньшие веса остальным переменным. Следовательно, эта компонента примерно соответствует уровню урбанизации штата. В целом мы видим, что переменные, имеющие отношение к уровню преступности (`Murder`, `Assault` и

[2] С технической точки зрения, направления главных компонент $\phi_1, \phi_2, \phi_3, \ldots$ представляют собой упорядоченные последовательности собственных векторов матрицы $\mathbf{X}^T\mathbf{X}$, а дисперсии элементов этих векторов — это собственные числа. Существует не более $\min(n-1, p)$ главных компонент.

[3] Точного перевода этого термина на русский язык не существует. — *Прим. пер.*

Rape), расположены близко друг к другу, а UrbanPop находится далеко от остальных трех переменных. Это указывает на то, что переменные, характеризующие уровень преступности, коррелируют друг с другом, т. е. штаты с высоким количеством убийств обычно имеют также много нападений и изнасилований, тогда как переменная UrbanPop коррелирует с этими тремя переменными слабее.

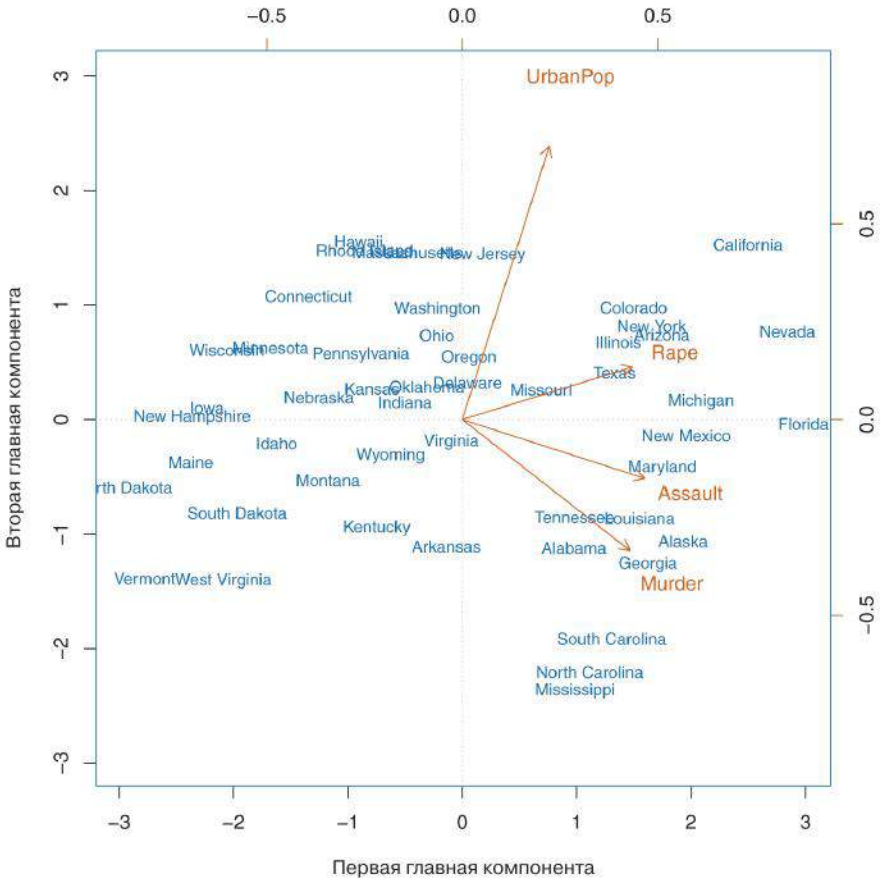

РИСУНОК 10.1. *Первые две компоненты данных* USArrests. *Названия штатов, показанные голубым шрифтом, соответствуют значениям первых двух главных компонент. Коричневые стрелки показывают нагрузки первых двух главных компонент (соответствующие оси приведены справа и вверху графика). Например, нагрузка* Rape *на первую главную компоненту составляет 0.54, а на вторую компоненту — 0.17 (слово* Rape *центрировано относительно точки (0.54, 0.17)). Этот график известен как «biplot», поскольку на нем одновременно показаны как значения главных компонент, так и нагрузки этих компонент*

Мы можем оценить разницу между штатами при помощи показанных на рис. 10.1 векторов значений главных компонент. Как следует из нашего предыдущего обсуждения, штаты с большими положительными значени-

ями первой компоненты, такие как Калифорния, Невада и Флорида, характеризуются высокими уровнями преступности, тогда как штаты вроде Северной Дакоты, которые имеют отрицательные значения первой компоненты, характеризуются низкими уровнями преступности. Калифорния обладает также высоким значением второй компоненты, что говорит о высоком уровне урбанизации, тогда как в штатах вроде Миссисипи наблюдается обратная ситуация. Штаты, расположенные близко к нулю по обеим компонентам (например, Индиана), имеют примерно средние уровни как преступности, так и урбанизации.

10.2.2 Альтернативная интерпретация главных компонент

Слева на рис. 10.2 показаны векторы факторных нагрузок первых двух главных компонент для набора имитированных данных; эти два вектора образуют плоскость, вдоль которой наблюдения демонстрируют наибольшую дисперсию.

В предыдущем подразделе мы описывали векторы нагрузок главных компонент как направления в пространстве признаков, вдоль которых данные варьируют в наибольшей степени, а значения главных компонент — как проекции на эти направления. Однако полезной может оказаться также и другая интерпретация: главные компоненты образуют линейные поверхности малой размерности, расположенные *максимально близко* к наблюдениям. Рассмотрим эту интерпретацию подробнее.

Вектор нагрузок первой главной компоненты обладает особым свойством: он представляет собой линию в p–мерном пространстве, расположенную максимально близко к n наблюдениям (в качестве меры близости используется среднеквадратичное евклидово расстояние). Эта интерпретация проиллюстрирована на рис. 6.15 слева: прерывистые линии показывают расстояние между каждым наблюдением и вектором нагрузок первой главной компоненты. Очевидна привлекательность подобной интерпретации: мы пытаемся найти единственную ось в данных, которая находится максимально близко ко всем наблюдениям, поскольку такая линия с большой вероятностью хорошо обобщит свойства этих данных.

Представление главных компонент в виде осей, находящихся максимально близко к n наблюдениям, применимо не только к первой главной компоненте. Например, первые две главные компоненты некоторого набора данных образуют плоскость, находящуюся максимально близко к n наблюдениям (в терминах среднеквадратичного евклидова расстояния). Слева на рис. 10.2 приведен пример. Первые три главные компоненты некоторого набора данных образуют трехмерную гиперплоскость, находящуюся максимально близко к n наблюдениями, и т. д.

В рамках этой интерпретации первые M векторов значений главных компонент, а также первые M векторов нагрузок главных компонент вместе обеспечивают оптимальную M–мерную аппроксимацию (по евклидову расстоянию) i–го наблюдения x_{ij}. Это можно записать следующим образом (предполагая, что все столбцы исходной матрицы данных $\mathbf{X}$ были центрированы):

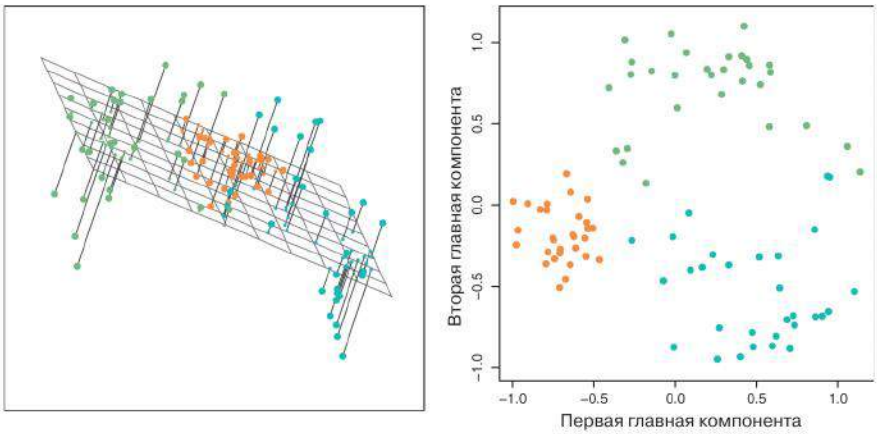

РИСУНОК 10.2. *Девяносто наблюдений из набора имитированных данных с тремя переменными.* Слева: *первые две главные компоненты образуют плоскость, которая аппроксимирует данные наилучшим образом. Сумма квадратов расстояний от каждой точки до плоскости минимальна.* Справа: *первые два вектора значений главных компонент дают координаты 90 наблюдений, спроецированных на плоскость. Дисперсия на этой плоскости максимальна*

$$x_{ij} \approx \sum_{m=1}^{M} z_{im}\phi_{jm}. \tag{10.5}$$

Другими словами, при достаточно большом M первые M векторов значений главных компонент и первые M векторов нагрузок главных компонент в совокупности могут дать удовлетворительную аппроксимацию исходных данных. При $M = \min(n-1, p)$ это представление данных будет точным: $x_{ij} = \sum_{m=1}^{M} z_{im}\phi_{jm}$.

10.2.3 Дополнительный материал по PCA

Масштабирование переменных

Мы уже упоминали, что перед выполнением PCA переменные следует центрировать, для того чтобы их средние значения стали равны нулю. Более того, результаты, полученные в ходе выполнения PCA, зависят также от выполнения масштабирования отдельных переменных (т. е. умножение переменных на те или иные константы). Этим PCA отличается от некоторых других методов обучения с учителем и без учителя, таких как линейная регрессия, где масштабирование переменных не оказывает никакого эффекта. (В случае с линейной регрессией умножение некоторой переменной на константу c просто приведет к умножению оценки соответствующего коэффициента на $1/c$ и поэтому не окажет никакого влияния на итоговую модель.)

Например, рис. 10.1 был получен после масштабирования каждой переменной таким образом, чтобы стандартное отклонение каждой из них стало равно 1. Эта ситуация показана еще раз слева на рис. 10.3. Почему масштабирование переменных имеет значение? В этом наборе данных переменные выражаются в разных единицах: `Murder`, `Rape` и `Assault` учтены как число случаев на 100 000 населения, а `UrbanPop` представляет собой процент городского населения в том или ином штате. Дисперсии этих четырех переменных составляют 18.97, 87.73, 6945.16 и 209.5 соответственно. Следовательно, если мы выполним PCA без масштабирования переменных, то вектор факторных нагрузок первой главной компоненты будет содержать очень большую нагрузку для `Assault`, поскольку эта переменная обладает наибольшей дисперсией. Справа на рис. 10.3 показаны первые две главные компоненты для данных `USArrests` без масштабирования переменных, в результате которого их стандартные отклонения становятся равными 1. Как и ожидалось, первая главная компонента почти полностью определяется переменной `Assault`, тогда как вектор факторных нагрузок второй главной компоненты присваивает основной вес переменной `UrbanPop`. Сравнивая эти результаты с графиком, представленным слева, мы видим, что масштабирование действительно оказывает существенный эффект на получаемые результаты.

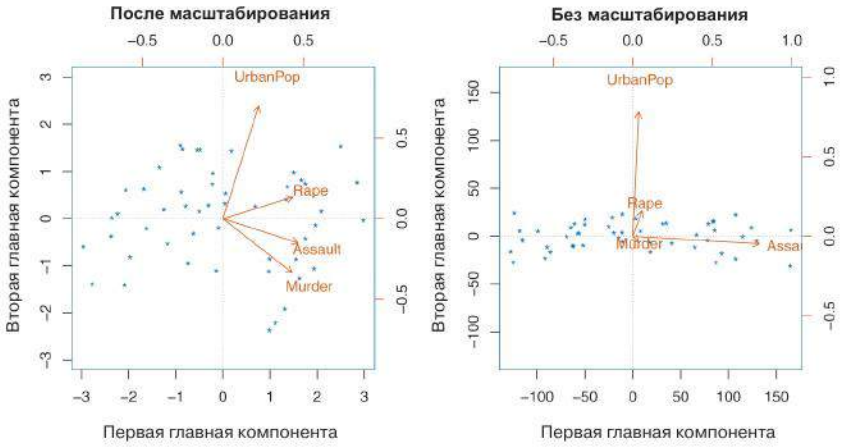

РИСУНОК 10.3. *Графики типа «biplot» для двух главных компонент данных* `USArrests`. Слева: *то же, что и на рис. 10.1, где переменные масштабированы и имеют стандартные отклонения, равные 1.* Справа: *главные компоненты данных без масштабирования переменных. Из всех четырех переменных* `Assault` *характеризуется наибольшей дисперсией и поэтому оказывает чрезвычайно высокую нагрузку на первую главную компоненту. В целом рекомендуется масштабировать переменные таким образом, чтобы приравнять их стандартные отклонения к единице*

Однако этот результат просто является следствием использования разных шкал измерения. Например, если бы переменная `Assault` выражалась как число случаев на 100 человек (в отличие от 100 000 человек), то это было бы эквивалентно делению всех значений этой переменной на 1000.

В таком случае дисперсия Assault оказалась бы очень низкой, и вектор нагрузок первой главной компоненты содержал бы очень низкое значение для этой переменной. Поскольку ситуация, когда получаемые главные компоненты зависят от произвольно выбранного способа масштабирования, является нежелательной, то обычно перед выполнением PCA мы масштабируем каждую переменную таким образом, чтобы ее стандартное отклонение стало равным 1.

Однако иногда переменные могут выражаться в одинаковых единицах. В таких случаях необходимость в масштабировании переменных перед выполнением PCA отпадает. Предположим, например, что переменные в некотором наборе данных соответствуют уровням экспрессии p генов. Поскольку уровень экспрессии выражается в одинаковых «единицах» для всех генов, мы можем не масштабировать значения по отдельным генам так, чтобы их стандартные отклонения стали равны 1.

Уникальность главных компонент

Каждый вектор нагрузок главных компонент является уникальным (по абсолютным значениям его элементов). Это означает, что два разных статистических пакета дадут одинаковые векторы факторных нагрузок главных компонент, хотя знаки у элементов этих векторов могут различаться. Знаки могут различаться в связи с тем, что каждый вектор факторных нагрузок главных компонент задает направление в p–мерном пространстве: изменение знака на обратный не окажет никакого влияния, поскольку направление при этом не изменится. (Рассмотрите рис. 6.14: вектор нагрузок главных компонент представляет собой линию, которая простирается в обе стороны, и поэтому изменение знака на обратный не оказало бы никакого эффекта). Векторы значений главных компонент также уникальны в абсолютном выражении своих элементов, поскольку дисперсия Z ничем не отличается от дисперсии $-Z$. Следует отметить, что при использовании (10.5) для аппроксимации x_{ij} мы умножаем z_{im} на ϕ_{jm}. Таким образом, при изменении знака на обратный одновременно в векторах нагрузок и в векторах значений главных компонент результат произведения этих двух величин останется неизменным.

Доля объясненной дисперсии

На рис. 10.2 приведены результаты применения PCA к набору данных с тремя переменными (слева) и проецирования наблюдений на векторы нагрузок первых двух главных компонент для представления этих данных в двумерном пространстве (т. е. для нахождения векторов значений главных компонент; см. график справа). Мы видим, что это двумерное представление трехмерных данных действительно удачно передает основные их свойства: оранжевые, зеленые и голубые точки, находящиеся близко друг к другу в трехмерном пространстве, по–прежнему лежат близко друг к другу и в двумерном пространстве. Аналогичную ситуацию мы видели и в случае с набором данных USArrests, где 50 наблюдений и четыре переменные можно было обобщить с использованием первых двух векторов значений и первых двух векторов нагрузок главных компонент.

Теперь мы можем задать естественный вопрос: как много информации, заключенной в некотором наборе данных, утрачивается в результате проецирования наблюдений на первые несколько главных компонент? Другими словами, как много от имеющейся в данных дисперсии *не содержится* в первых нескольких главных компонентах? В более общей формулировке, нам интересна *доля дисперсии, объясненная каждой главной компонентой* (PVE)[4]. *Общую дисперсию*, заключенную в некотором наборе данных, находят следующим образом (предполагается, что переменные были центрированы таким образом, чтобы их средние значения стали равны 0):

доля объясненной дисперсии

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2, \tag{10.6}$$

а дисперсию, объясненную m–й компонентой как

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2. \tag{10.7}$$

Отсюда PVE m–й главной компоненты составляет

$$\frac{\sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}. \tag{10.8}$$

PVE каждой главной компоненты является положительной величиной. Для вычисления кумулятивной PVE первых M главных компонент мы можем просто суммировать значения PVE, вычисленные для этих M главных компонент по (10.8). В целом существует $\min(n-1, p)$ главных компонент, и значения их PVE в сумме дают 1.

В случае с данными `USArrests` первая главная компонента объясняет 62.0% общей дисперсии, а следующая главная компонента объясняет 24.7% дисперсии. В сумме первые две главные компоненты объясняют почти 87% дисперсии, заключенной в этих данных, а последние две компоненты — только 13% дисперсии. Это означает, что рис. 10.1 обеспечивает довольно точное обобщение данных, используя только два измерения. PVE каждой главной компоненты, а также кумулятивная PVE показаны на рис. 10.4. Справа на этом рисунке приведен график, известный как *«график каменистой осыпи»*[5], и мы обсудим его ниже.

Выбор числа главных компонент

В целом матрица данных $\mathbf{X}$ размером $n \times p$ имеет $\min(n-1, p)$ главных компонент. Однако обычно нам не интересны все из них: вместо этого для визуализации или интерпретации данных мы предпочли бы использовать только первые несколько главных компонент. Более того, мы предпочли бы использовать наименьшее число главных компонент, достаточное для

[4] Аббревиатура от «proportion of variance explained». — *Прим. пер.*

[5] В оригинале используется термин «scree plot». Устоявшегося перевода этого термина на русский язык не существует. — *Прим. пер.*

хорошего понимания данных. Сколько главных компонент будет достаточно? К сожалению, однозначного (равно как и простого!) ответа на этот вопрос не существует.

Обычно для визуализации данных мы выбираем оптимальное число главных компонент при помощи «*графика каменистой осыпи*», подобного тому, который приведен слева на рис. 10.4. Мы выбираем наименьшее число главных компонент, необходимое для объяснения значительной доли заключенной в данных дисперсии. Это происходит путем визуального изучения графика с целью обнаружения точки, в которой доля объясненной дисперсии заметно снижается. Часто такую точку называют *изломом*. Например, изучая рис. 10.4, можно было бы заключить, что довольно большая доля дисперсии объясняется первыми двумя главными компонентами и что излом возникает после второй главной компоненты. Третья главная компонента объясняет менее 10% общей дисперсии, а четвертая главная компонента — менее половины этого, в связи с чем она практически бесполезна.

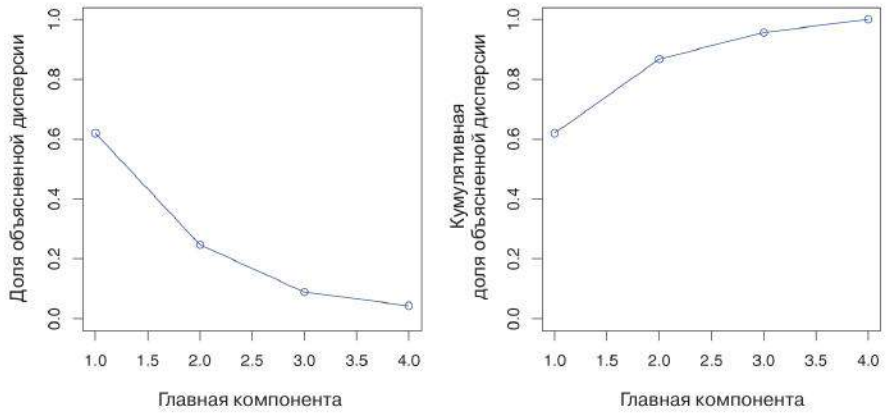

РИСУНОК 10.4. Слева: *«график каменистой осыпи», на котором показана доля дисперсии, объясненной каждой из четырех главных компонент данных* `USArrests`. Справа: *кумулятивная доля дисперсии, объясненная четырьмя главными компонентами данных* `USArrests`

Однако подобный визуальный анализ, по определению, является *субъективным*. К сожалению, общепринятого объективного подхода для определения *достаточного* числа главных компонент не существует. Более того, сам вопрос о том, какое число главных компонент является достаточным, плохо определен и будет зависеть от конкретной области применения анализа и свойств конкретного набора данных. На практике мы обычно рассматриваем несколько первых главных компонент для обнаружения интересных закономерностей в данных. Если такие закономерности не найдены для первых нескольких компонент, то маловероятно, что интересными окажутся последующие главные компоненты. И наоборот: если первые несколько главных компонент оказываются интересными, то мы обычно будем продолжать изучение последующих главных компонент до того момента, пока интересные закономерности в данных не исчезнут. Ко-

нечно, это тоже субъективный подход, который отражает тот факт, что PCA обычно используется как инструмент разведочного анализа данных.

С другой стороны, если мы вычисляем главные компоненты для последующего применения в обучении с учителем (например, в регрессии на главные компоненты, описанной в подразделе 6.3.1), то существует простой и объективный метод определения оптимального числа главных компонент: мы можем рассматривать число векторов со значениями главных компонент как гиперпараметр, подлежащий нахождению при помощи перекрестной проверки или какого–либо другого родственного ей метода. Относительная простота выбора числа главных компонент для обучения с учителем — это одно из проявлений того факта, что обучение с учителем обычно является более четко определенной и объективно решаемой задачей, чем обучение без учителя.

10.2.4 Другие приложения PCA

В подразделе 6.3.1 мы видели, что регрессионный анализ можно выполнить, используя векторы значений главных компонент в качестве предикторов. Более того, многие статистические методы регрессии, классификации и кластеризации можно легко применять не к исходной матрице данных размером $n \times p$, а к матрице $n \times M$, столбцы в которой представляют собой первые $M \ll p$ векторов со значениями главных компонент. Это может приводить к *менее зашумленным* результатам, поскольку часто случается так, что сигнал в данных (в отличие от шума) сконцентрирован в первых нескольких главных компонентах.

10.3 Методы кластеризации

Под кластеризацией понимают очень широкий круг методов, предназначенных для обнаружения *групп*, или *кластеров*, в данных. Выполняя кластеризацию наблюдений из некоторого набора данных, мы пытаемся разбить их на отдельные группы таким образом, чтобы наблюдения внутри каждой группы были похожи друг на друга, а наблюдения из разных групп заметно отличались друг от друга. Конечно, чтобы конкретизировать эту задачу, мы должны дать определение тому, что подразумевается под *сходством* или *различием* двух или более наблюдений. Безусловно, часто этот выбор будет определяться предметной областью и знанием исследуемых данных.

Предположим, например, что у нас есть набор из n наблюдений, каждое из которых описано по p признакам. Эти n наблюдений могли бы соответствовать образцам тканей, взятых у пациенток с раком груди, а p признаков могли бы соответствовать измерениям, выполненным на каждом образце (например, измерения клинических параметров, таких как стадия рака или уровень экспрессии генов). Возможно, у нас есть основания считать, что эти n наблюдений в некоторой степени неоднородны; например, могут существовать несколько *неизвестных* типов рака груди. Для обнаружения этих типов можно применить кластеризацию. Здесь мы имеем дело с задачей обучения без учителя, поскольку мы пытаемся выявить структуру (в данном случае различные кластеры) на основе

некоторого набора данных. Цель же задач обучения с учителем заключается в предсказании вектора значений некоторой зависимой переменной, такой как время выживания или реакция на введение лекарства.

И кластерный анализ, и PCA пытаются упростить данные, вычисляя небольшой набор сводных величин, однако механизмы этих методов различаются:

- цель PCA состоит в представлении данных в подпространстве малой размерности, объясняющем достаточную долю общей дисперсии;
- цель кластеризации состоит в нахождении однородных групп наблюдений.

С еще одним приложением кластеризации можно столкнуться в маркетинге. У нас может иметься доступ к большому числу измерений (например, медианного уровня дохода в расчете на домохозяйство, профессионального положения, расстояния до ближайшего городского поселения и т. п.) для большого числа людей. Наша задача заключается в *сегментировании рынка* путем обнаружения групп людей, которые могли бы с большей частотой реагировать на рекламу определенного вида или с большей вероятностью покупать определенный продукт. Эта задача сводится к кластеризации людей, входящих в наш набор данных.

Поскольку кластерный анализ популярен во многих областях, существует большое количество методов для его выполнения. В этом разделе мы сосредоточимся на двух, возможно, наиболее популярных подходах: *кластеризации по методу K средних* и *иерархической кластеризации*. В случае с кластеризацией по методу K средних мы пытаемся разбить наблюдения на некоторое заранее заданное число кластеров. В случае же с иерархической кластеризацией желаемое число кластеров нам заранее неизвестно; более того, в результате такого анализа мы получаем древовидное представление наблюдений — *дендрограмму*, которая позволяет нам одновременно увидеть все возможные кластеры — от 1 до n. Каждый из этих методов обладает своими преимуществами и недостатками, которые мы опишем в этой главе.

метод K средних

иерархическая кластеризация

дендрограмма

В целом мы можем выполнить кластеризацию наблюдений по их признакам для обнаружения групп среди этих наблюдений или кластеризацию признаков по наблюдениям для обнаружения групп среди этих признаков. Ниже для простоты мы будем обсуждать кластеризацию наблюдений на основе признаков, хотя обратную задачу можно выполнить посредством простой транспозиции матрицы данных.

10.3.1 Кластеризация по методу K средних

Кластеризация по методу K средних представляет собой простой и элегантный подход для разбиения некоторого набора данных на K отдельных, непересекающихся кластеров. Для выполнения кластеризации по этому методу мы должны сначала указать желаемое число кластеров K; затем соответствующий алгоритм отнесет каждое наблюдение в точности к одному из K кластеров. На рис. 10.5 показаны результаты, полученные в результате применения кластеризации по методу K средних с разными

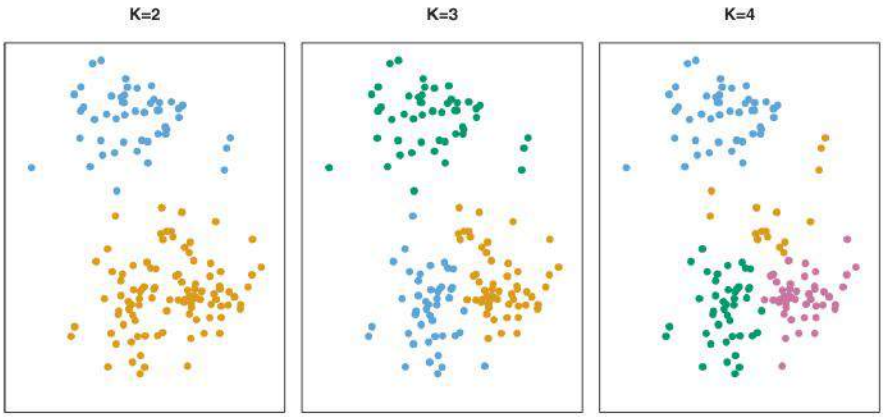

РИСУНОК 10.5. *Набор имитированных данных, содержащий 150 наблюдений и два признака. Графики показывают результаты применения метода K средних с разными значениями K (число кластеров). Цвет каждого наблюдения обозначает кластер, к которому оно было отнесено алгоритмом кластеризации. Заметьте, что кластеры никак не упорядочены, в связи с чем выбор цвета точек произволен. Данные метки кластеров не были использованы в ходе кластеризации — они являются результатом работы самой этой процедуры*

значениями K к набору имитированных данных, содержащему 150 наблюдений в двумерном пространстве.

Процедура кластеризации на основе K средних вытекает из простой и интуитивно понятной математической проблемы. Мы начнем с введения некоторых обозначений. Пусть $C_1, \ldots, C_K$ обозначают наборы индексов наблюдений из каждого кластера. Эти наборы обладают двумя свойствами:

1. $C_1 \cup C_2 \cup \cdots \cup C_k = \{1, \ldots, n\}$. Другими словами, каждое наблюдение принадлежит по меньшей мере к одному из K кластеров.

2. $C_k \cap C_{k'} = \emptyset$ для всех $k \neq k'$. Другими словами, кластеры не пересекаются: ни одно из наблюдений не может принадлежать к более чем одному кластеру.

Например, если i–е наблюдение входит в состав k–го кластера, то $i \in C_k$. Идея, лежащая в основе метода K средних, заключается в том, что *хорошей* является кластеризация, при которой *внутрикластерный разброс*[6] минимален. Внутрикластерный разброс для кластера C_k — это величина $W(C_k)$, отражающая степень отличий наблюдений из данного кластера друг от друга. Таким образом, мы хотим решить следующую задачу:

$$\underset{C_1,\ldots,C_K}{\text{минимизировать}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}. \tag{10.9}$$

[6] В оригинале используется термин «within–cluster variation». — *Прим. пер.*

Другими словами, согласно этой формуле, мы хотим разбить наблюдения на K кластеров таким образом, чтобы общий внутрикластерный разброс, полученный путем суммирования по всем K кластерам, был минимальным.

Попытка найти решение для (10.9) выглядит как разумная идея, однако для ее практической реализации нам необходимо дать определение внутрикластерного разброса. Для этого существует большое число способов, но чаще всего это определение базируется на возведенном в квадрат *евклидовом расстоянии*:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2, \tag{10.10}$$

где $|C_k|$ обозначает число наблюдений в k–м кластере. Другими словами, внутрикластерный разброс в кластере k — это сумма квадратов евклидовых расстояний между всеми парами наблюдений в этом кластере, разделенная на общее число входящих в него наблюдений. Объединение (10.9) и (10.10) дает оптимизационную задачу кластеризации по методу K средних:

$$\underset{C_1,\ldots,C_K}{\text{минимизировать}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}. \tag{10.11}$$

Теперь мы хотели бы найти алгоритм для решения (10.11), т. е. такой метод разбиения наблюдений на K кластеров, который минимизирует целевую функцию из (10.11). Оказывается, что получить точное решение этой задачи очень трудно, поскольку существует почти K^n способов разбиения n наблюдений на K кластеров. Это огромное число (если только K не является очень малым)! К счастью, можно показать, что один очень простой алгоритм позволяет найти локальный оптимум, т. е. *очень хорошее приблизительное решение* оптимизационной задачи (10.11). Этот подход представлен в алгоритме 10.1.

Алгоритм 10.1 Кластеризация по методу K средних

1. Каждому наблюдению присвойте случайно выбранное число из интервала от 1 до K. Эти числа будут служить в качестве исходных меток кластеров.

2. Повторите следующие шаги несколько раз до тех пор, пока метки классов не перестанут изменяться:

   (a) вычислите *центроид* для каждого из K кластеров. Центроид k–го класса представляет собой вектор из p средних значений признаков, описывающих наблюдения из этого кластера;

   (b) присвойте каждому наблюдению метку того кластера, чей центроид находится ближе всего к этому наблюдению (здесь удаленность выражается в виде евклидова расстояния).

Алгоритм 10.1 на каждом шаге будет гарантированно снижать значение целевой функции (10.11). Следующий пример помогает понять, почему это так:

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2, \tag{10.12}$$

где $\bar{x}_{kj} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij}$ — это среднее значение признака j в кластере C_k. На шаге 2(a) средние значения признаков в каждом кластере являются константами, которые минимизируют сумму квадратов отклонений, а на шаге 2(b) перераспределение меток кластеров может только улучшать (10.12). Это означает, что по мере выполнения алгоритма получаемая кластеризация будет постепенно улучшаться до тех пор, пока результат не перестанет изменяться; целевая функция (10.11) никогда не будет возрастать. Стабилизация результата кластеризации означает, что *локальный оптимум* достигнут. На рис. 10.6 приведена динамика выполнения алгоритма на примере имитированных данных из рис. 10.5. Название метода K средних обусловлено тем обстоятельством, что на шаге 2(a) центроиды вычисляются как средние значения наблюдений, отнесенных к каждому кластеру.

Поскольку метод K средних находит локальный, а не глобальный оптимум, то полученные результаты будут зависеть от исходного (случайного) разбиения наблюдений на кластеры на первом шаге алгоритма 10.1. По этой причине важно запускать алгоритм несколько раз, используя разные исходные случайные конфигурации. После этого выбирают *оптимальное* решение, т. е. решение, обеспечивающее наименьшее значение целевой функции (10.11). На рис. 10.7 показаны локальные оптимумы, полученные в результате шестикратного запуска алгоритма K средних на данных из рис. 10.5 с использованием шести разных исходных разбиений на кластеры. В данном случае оптимальной является кластеризация со значением целевой функции, равным 235.8.

Итак, для выполнения кластеризации по методу K средних мы должны решить, сколько кластеров мы ожидаем обнаружить в данных. Проблема выбора K далеко непроста. Эта проблема, а также некоторые другие практические аспекты, возникающие при выполнении кластеризации по методу K средних, будут рассмотрены в подразделе 10.3.3.

10.3.2 Иерархическая кластеризация

Одним из потенциальных недостатков кластеризации по методу K средних является то, что она требует от нас предварительного указания числа кластеров. *Иерархическая кластеризация* представляет собой альтернативный подход, который не требует, чтобы мы придерживались какого-то конкретного выбора K. Дополнительное преимущество иерархической кластеризации в сравнении с методом K средних заключается в том, что она приводит к привлекательному представлению данных в виде древовидной структуры, которую называют *дендрограммой*.

В этом подразделе мы опишем кластеризацию типа «*снизу вверх*», или

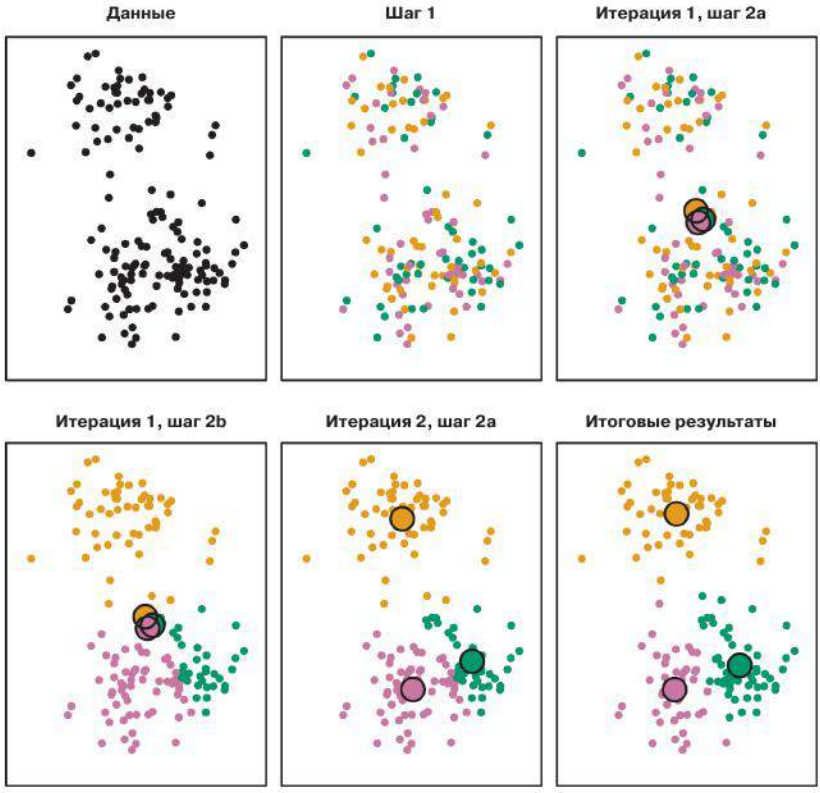

РИСУНОК 10.6. *Динамика выполнения алгоритма K средних на примере данных из рис. 10.5 с K = 3.* Слева вверху: *представлены исходные данные.* В центре вверху: *на первом шаге алгоритма наблюдения разбиваются на кластеры случайным образом.* Справа вверху: *на шаге 2(a) вычисляются центроиды кластеров. Они показаны в виде больших цветных кружков. Изначально центроиды почти полностью пересекаются, поскольку исходное разбиение на кластеры выполняется случайным образом.* Слева внизу: *на шаге 2(b) метка кластера присваивается каждому наблюдению в соответствии с ближайшим центроидом.* Справа внизу: *результаты, полученные после шести итераций*

агломеративная кластеризация

агломеративную кластеризацию[7]. Это наиболее распространенный тип иерархической кластеризации, который заключается в том, что построение дендрограммы (обычно изображается в виде перевернутого дерева — см. рис. 10.9) начинается с листьев и продолжается путем их последовательного объединения в кластеры вплоть до самого «ствола». Мы начнем с обсуждения интерпретации дендрограммы, а затем обсудим собственно механизм выполнения иерархической кластеризации, т. е. способ построения дендрограммы.

[7] В оригинале используются термины «bottom–up clustering» и «agglomerative clustering» соответственно. — *Прим. пер.*

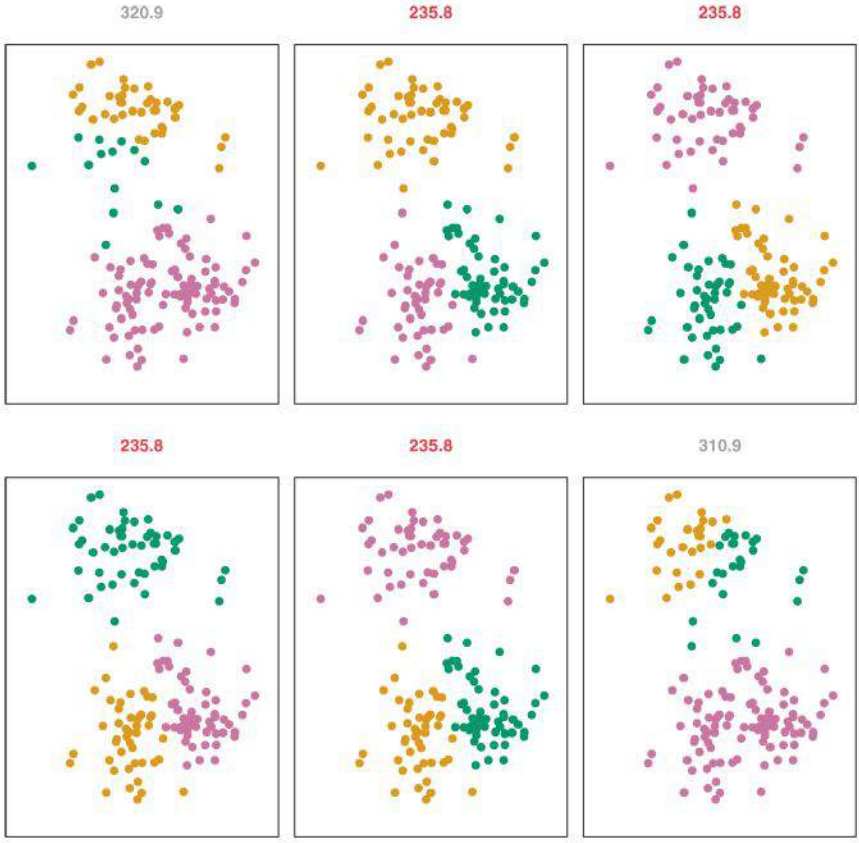

РИСУНОК 10.7. *Кластеризация по методу K средних ($K = 3$), выполненная шесть раз на данных из рис. 10.5 с разными случайными разбиениями наблюдений на кластеры на первом шаге алгоритма. Над каждым графиком показано значение целевой функции (10.11). Получены три локальных оптимума, один из которых привел к более низкому значению целевой функции и дал более качественное разбиение на кластеры. Все итерации, у которых значения целевой функции выделены красным цветом (235.8), дали одинаково оптимальные решения*

Интерпретация дендрограммы

Сначала рассмотрим представленный на рис. 10.8 набор имитированных данных, включающий 45 наблюдений и две переменные. Эти данные были сгенерированы на основе модели с тремя классами; истинные метки классов показаны разными цветами. Предположим, однако, что данные были получены без информации о метках классов и что мы хотели бы выполнить иерархическую кластеризацию этих наблюдений. Иерархическая кластеризация (с «полным присоединением», которое обсуждается ниже) дает результат, показанный слева на рис. 10.9. Как нам интерпретировать эту дендрограмму?

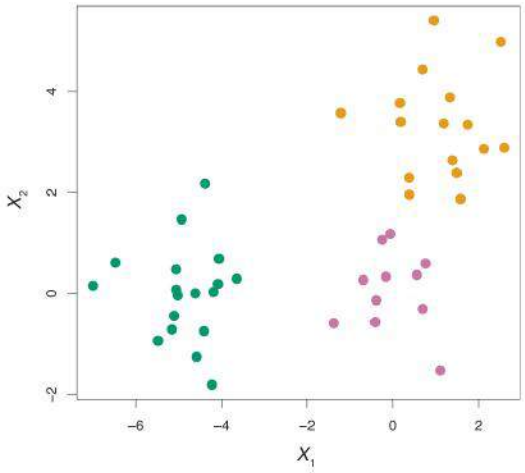

РИСУНОК 10.8. *Сорок пять имитированных наблюдений в двумерном пространстве. В действительности есть три отдельных класса, показанных разными цветами. Однако мы будем считать, что эти метки классов неизвестны, и попытаемся объединить наблюдения в группы для восстановления информации о классах по имеющимся данным*

Слева на рис. 10.9 каждый *лист* дендрограммы соответствует одному из 45 наблюдений, показанных на рис. 10.8. Однако по мере продвижения вверх по дереву некоторые листья начинают *сливаться* в ветви. Это наблюдения, которые похожи друг на друга. При дальнейшем продвижении вверх ветви также начинают сливаться — либо с листьями, либо с другими ветвями. Чем раньше (т. е. чем ниже в структуре дерева) происходит слияние, тем выше степень сходства между группами наблюдений. В то же время наблюдения, которые сливаются позже (т. е. недалеко от ствола дерева), могут быть довольно непохожими друг на друга. Более того, это утверждение можно сделать строгим: для любых двух наблюдений мы можем найти точку в структуре дерева, где происходит первое слияние ветвей, содержащих эти два наблюдения. Высота, на которой происходит слияние (показано на вертикальной оси), отражает степень отличий между двумя наблюдениями. Таким образом, наблюдения, которые сливаются у самого основания дерева, являются очень похожими друг на друга, тогда как наблюдения, которые сливаются у ствола дерева, обычно довольно разные.

Это подчеркивает одно очень важное обстоятельство, возникающее при интерпретации дендрограмм и часто понимаемое неправильно. Рассмотрим простую дендрограмму, приведенную слева на рис. 10.10, которая была получена в результате иерархической кластеризации девяти наблюдений. Можно увидеть, что наблюдения 5 и 7 довольно похожи друг на друга, поскольку они сливаются в наиболее низко расположенной точке дендрограммы. В то же время неверным (хотя и заманчивым) было бы заключение о том, что в силу своего близкого расположения друг к другу на этой дендрограмме похожими являются также наблюдения 9 и 2.

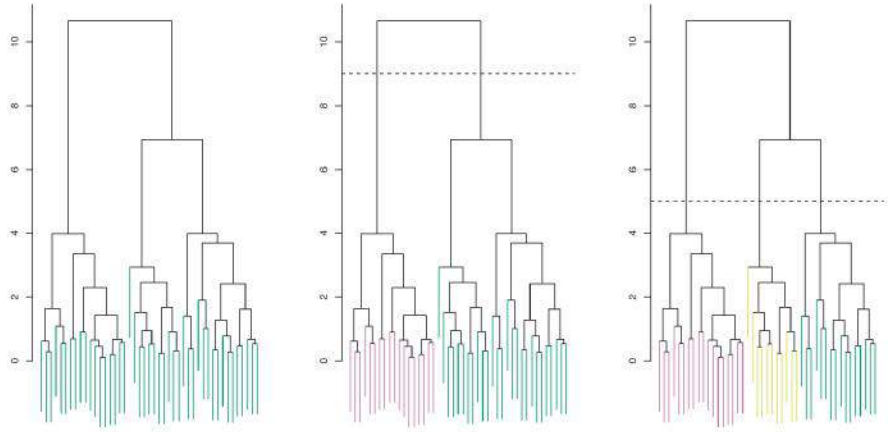

РИСУНОК 10.9. Слева: *дендрограмма, полученная путем иерархической кластеризации данных из рис. 10.8 с использованием полного присоединения и евклидова расстояния.* В центре: *дендрограмма, приведенная слева, рассечена на высоте 9 (отмечена прерывистой линией). Это приводит к двум кластерам, показанным при помощи разных цветов.* Справа: *дендрограмма, приведенная слева, рассечена на высоте 5. Это приводит к трем отдельным кластерам, показанным при помощи разных цветов. Обратите внимание, что цветовые метки не были задействованы в ходе кластеризации и используются здесь просто для наглядности*

В действительности, как следует из представленной на дендрограмме информации, наблюдение 9 похоже на наблюдение 2 не больше, чем оно похоже на наблюдения 8, 5 и 7. (Это можно увидеть справа на рис. 10.10, где приведены исходные данные.) С математической точки зрения, существует 2^{n-1} возможных вариантов упорядочения ветвей дендрограммы, где n — это число листьев. Это обусловлено тем, что в каждой из $n-1$ точек слияния позиции двух слившихся ветвей можно поменять местами, не изменяя смысла дендрограммы. Следовательно, мы не можем делать выводы о сходстве двух наблюдений на основе их взаимной близости по *горизонтальной оси*. Вместо этого мы делаем выводы о сходстве двух наблюдений, исходя из положения на вертикальной оси той точки, где происходит слияние этих двух наблюдений.

Теперь, когда мы узнали, как интерпретировать дендрограмму, представленную на рис. 10.9 слева, мы можем продолжить и определить кластеры на ее основе. Для этого мы проводим горизонтальную линию, рассекающую дендрограмму, как показано на рис. 10.9 в центре и справа. Отдельные наборы наблюдений, лежащие ниже этой линии, можно интерпретировать как кластеры. В центре на рис. 10.9 рассечение дендрограммы на высоте 9 приводит к получению двух кластеров, выделенных разными цветами. На графике справа рассечение дендрограммы на высоте 5 приводит к решению с тремя кластерами. Спускаясь ниже, можно выполнить дополнительные рассечения для получения любого числа кластеров — от 1 (что эквивалентно отсутствию рассечения) до n (что соответствует

рассечению на высоте 0, в результате которого каждое наблюдение оказывается в своем собственном кластере). Другими словами, высота, на которой выполняется рассечение дендрограммы, играет ту же роль, что и K в кластеризации по методу K средних: она задает число получаемых кластеров.

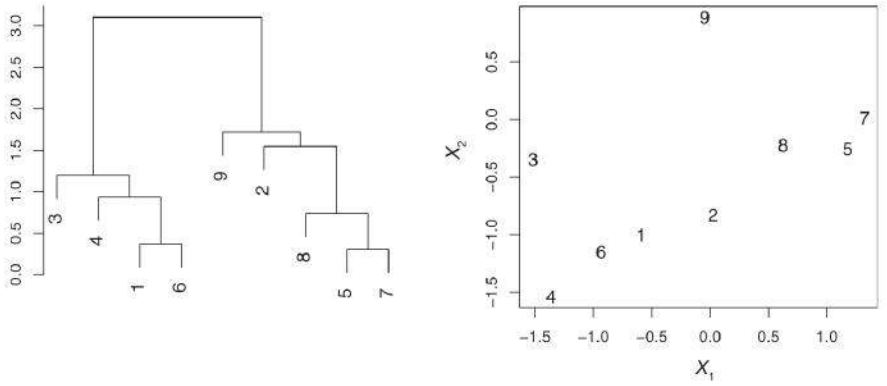

РИСУНОК 10.10. *Иллюстрация того, как правильно интерпретировать дендрограмму, построенную по девяти наблюдениям в двумерном пространстве.* Слева: *дендрограмма, созданная с использованием евклидова расстояния и полного присоединения. Наблюдения 5 и 7 довольно похожи друг на друга, равно как и наблюдения 1 и 6. Однако наблюдение 9 похоже на наблюдение 2 не больше, чем на наблюдения 8, 5 и 7, даже несмотря на то, что наблюдения 9 и 2 находятся близко друг к другу по горизонтали. Это обусловлено тем, что наблюдения 2, 8, 5 и 7 одновременно объединяются с наблюдением 9 на одинаковой высоте, равной примерно 1.8.* Справа: *исходные данные, по которым была построена дендрограмма, можно использовать для подтверждения того, что наблюдение 9 действительно похоже на наблюдение 2 не больше, чем на наблюдения 8, 5 и 7*

Таким образом, рис. 10.9 подчеркивает очень привлекательный аспект иерархической кластеризации: одну и ту же дендрограмму можно использовать для получения любого числа кластеров. На практике подходящее число кластеров выбирают путем визуального изучения дендрограммы, учитывая высоту, на которой происходит слияние отдельных ветвей, и желаемое число кластеров. В случае с рис. 10.9 можно было бы выбрать два или три кластера. Однако часто место рассечения дендрограммы не будет таким очевидным.

Термин «*иерархическая* кластеризация» предполагает, что кластеры, полученные в результате рассечения дендрограммы на некоторой высоте, обязательно являются «вложенными» в кластеры, полученные при рассечении этой дендрограммы на любой другой большей высоте. Однако для некоторых наборов данных это условие может не выполняться. Представьте, например, что наши наблюдения соответствуют группе людей, в которой в равных пропорциях представлены женщины и мужчины, а также жители Америки, Японии и Франции. Возможен сценарий, в кото-

ром оптимальное разбиение на две группы могло бы разделить этих людей по полу, а оптимальное разбиение на три группы могло бы разделить их по национальности. В таком случае кластеры не являются вложенными (в том смысле, что оптимальное разбиение на три группы не является естественным результатом предварительного разбиения на две группы). Следовательно, иерархическая кластеризация не смогла бы хорошо описать эту ситуацию. Из–за подобных ситуаций результаты, получаемые для некоторого заданного числа кластеров при помощи иерархической кластеризации, могут быть хуже (т. е. менее точными), чем результаты, получаемые при помощи кластеризации по методу K средних.

Алгоритм иерархической кластеризации

В ходе выполнения иерархической кластеризации дендрограмму получают при помощи очень простого алгоритма. Мы начинаем с определения некоторой меры *различия*[8] для каждой пары наблюдений. Чаще всего используется евклидово расстояние; мы обсудим выбор меры различия позже в этой главе. Выполнение алгоритма происходит за несколько итераций. Начиная с самого основания дендрограммы, каждое наблюдение рассматривается как самостоятельный кластер. Далее два наиболее похожих друг на друга кластера *сливаются*, в результате чего образуется $n-1$ кластеров. Затем два наиболее похожих друг на друга кластера снова сливаются, что приводит к формированию $n-2$ кластеров. Выполнение алгоритма продолжается аналогичным образом до тех пор, пока все наблюдения не становятся частью одного большого кластера, завершая формирование дендрограммы. На рис. 10.11 показаны первые несколько шагов алгоритма для данных из рис. 10.9. В обобщенном виде реализация иерархической кластеризации описана в алгоритме 10.2.

Алгоритм 10.2 Иерархическая кластеризация

1. Начните с n наблюдений и вычислите значения меры различия (например, евклидова расстояния) для всех $\binom{n}{2} = n(n-1)/2$ пар наблюдений. Рассматривайте каждое наблюдение как самостоятельный кластер.

2. Для $i = n, n-1, \ldots, 2$:

   (a) ранжируйте значения меры межкластерных различий для всех i кластеров и найдите пару наименее различающихся (т. е. наиболее похожих) кластеров. Объедините эти два кластера. Различие между этими двумя кластерами соответствует высоте, на которой должно происходить их слияние в дендрограмме;

   (b) вычислите новые значения меры различия для всех пар оставшихся $i-1$ кластеров.

[8] В оригинале используется термин «dissimilarity». — *Прим. пер.*

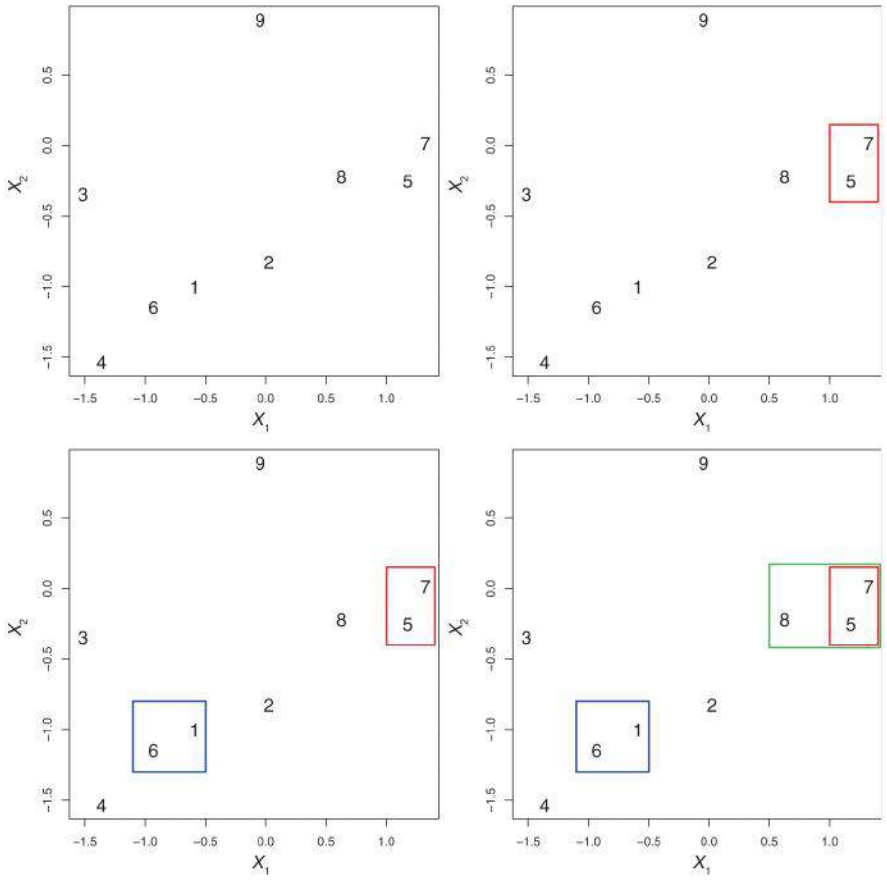

РИСУНОК 10.11. *Иллюстрация первых нескольких шагов алгоритма иерархической кластеризации с использованием полного присоединения и евклидова расстояния на примере данных из рис. 10.10.* Слева вверху: *сначала есть девять кластеров:* $\{1\}, \{2\}, \ldots, \{9\}$. Справа вверху: *два наиболее близких кластера —* $\{5\}$ *и* $\{7\}$ *— объединены в один кластер.* Слева внизу: *два наиболее близких кластера —* $\{6\}$ *и* $\{1\}$ *— объединены в один кластер.* Справа внизу: *два наиболее близких кластера —* $\{8\}$ *и* $\{5, 7\}$ *— объединены в один кластер по методу полного присоединения*

Этот алгоритм довольно прост, однако нерешенной остается одна проблема. Рассмотрим график, представленный справа внизу на рис. 10.11. Как мы определили, что кластер $\{5, 7\}$ должен быть объединен с кластером $\{8\}$? У нас есть понятие о различии между парой наблюдений, но как нам определить различие между двумя кластерами, если один или оба из них содержат несколько наблюдений? Идею о различии между парой наблюдений необходимо обобщить на случай пары *групп, включающих несколько наблюдений*. Это обобщение получают, вводя понятие *присоеди-*

присоединение

нения[9], которое обозначает способ выражения степени различий между группами наблюдений. В табл. 10.2 кратко описаны четыре основных типа присоединения — *полное*, *среднее*, *одиночное* и *центроидное*. У статистиков наиболее популярными типами присоединения являются среднее, полное и одиночное. Среднее и полное присоединения обычно предпочтительны одиночному присоединению, поскольку применение этих двух типов обычно приводит к более сбалансированным дендрограммам. Центроидное присоединение часто применяется в геномных исследованиях, однако у него есть один существенный недостаток: оно может сопровождаться *инверсией*, при которой два кластера в дендрограмме объединяются на высоте, находящейся ниже любого из этих кластеров. Это может затруднять визуализацию и интерпретацию дендрограммы. Значения меры различия, вычисляемые на шаге 2(b) алгоритма иерархической кластеризации, будут зависеть от используемого типа присоединения, а также от выбора самой меры различия. Как следствие получаемая дендрограмма обычно довольно сильно зависит от выбранного типа присоединения (см. рис. 10.12).

ТАБЛИЦА 10.2. *Краткое описание четырех наиболее часто используемых типов присоединения в иерархической кластеризации*

Присоединение	*Описание*
Полное	Максимальное различие между кластерами. Вычислите все парные различия между наблюдениями в кластере A и наблюдениями в кластере B и используйте *наибольшее* из полученных значений
Одиночное	Минимальное различие между кластерами. Вычислите все парные различия между наблюдениями в кластере A и наблюдениями в кластере B и используйте *наименьшее* из полученных значений. Одиночное присоединение может привести к «растянутым» кластерам, в которых отдельные наблюдения входят в кластер по одному за раз
Среднее	Среднее различие между кластерами. Вычислите все парные различия между наблюдениями в кластере A и наблюдениями в кластере B и используйте *среднее* из полученных значений
Центроидное	Различие между центроидом кластера A (средний вектор длиной p) и центроидом кластера B. Центроидное присоединение может приводить к нежелательному явлению *инверсии*

Выбор меры различия

До сих пор в примерах из этой главы в качестве меры различия применялось евклидово расстояние. Однако иногда предпочтительными могут ока-

[9] В оригинале используется термин «linkage». — *Прим. пер.*

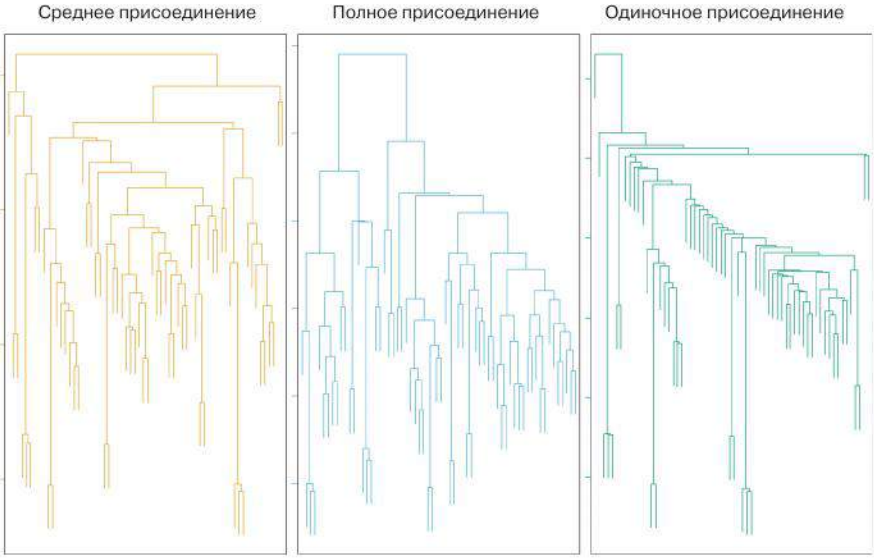

РИСУНОК 10.12. *Среднее, полное и одиночное присоединения применены к некоторому набору данных. Среднее и полное присоединения обычно дают более сбалансированные кластеры*

заться и другие меры. Например, согласно *расстоянию, основанному на корреляции*, два наблюдения являются похожими, если их признаки тесно коррелируют, даже несмотря на то, что наблюдаемые значения могут при этом находиться далеко друг от друга в терминах евклидова расстояния. Это необычный пример использования корреляции, которая в большинстве случаев рассчитывается для переменных; здесь же она рассчитывается для профилей наблюдений. Рисунок 10.13 демонстрирует разницу между евклидовым расстоянием и расстоянием, основанным на корреляции. Расстояние, основанное на корреляции, уделяет основное внимание очертанию «профилей» наблюдений, а не относительной удаленности наблюдений друг от друга.

Выбор меры различия очень важен, поскольку он оказывает существенное влияние на итоговую дендрограмму. В целом необходимо очень внимательно относиться к типу кластеризуемых данных, а также к стоящему научному вопросу. Эти аспекты должны определять выбор меры различия для выполнения иерархической кластеризации.

Рассмотрим, например, интернет–магазин, который хотел бы сгруппировать своих покупателей на основе их прошлых покупок. Цель заключается в нахождении групп *похожих* покупателей, чтобы в последующем показывать каждой группе такие товары и рекламные объявления, которые с большой вероятностью будут интересны членам соответствующих групп. Предположим, что данные представлены в виде матрицы, в которой строки соответствуют покупателям, а столбцы — всем доступным для покупки товарам; в ячейках матрицы представлены значения, показывающие, сколько раз тот или иной товар был куплен тем или иным

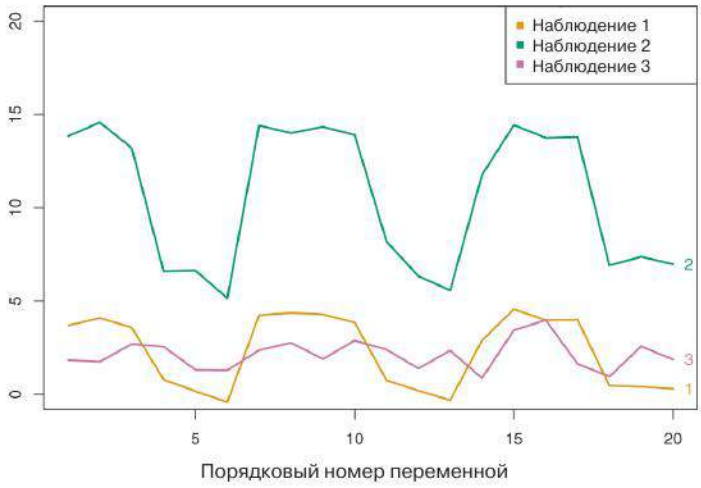

РИСУНОК 10.13. *Показаны три наблюдения, каждое из которых описано по 20 признакам. Наблюдения 1 и 3 имеют похожие значения для всех признаков, в связи с чем евклидово расстояние между ними невелико. Однако они очень слабо коррелируют, и поэтому с точки зрения корреляции они находятся далеко друг от друга. С другой стороны, значения большинства признаков у наблюдений 1 и 2 заметно различаются, в связи с чем евклидово расстояние между ними велико. В то же время эти два наблюдения сильно коррелируют, и поэтому с точки зрения корреляции расстояние между ними невелико*

покупателем (0, если товар не был куплен ни разу, 1, если товар был куплен один раз, и т. д.). Какую меру различия следует применить для кластеризации покупателей? Если использовать евклидово расстояние, то покупатели, которые купили лишь небольшое количество разных товаров (т. е. нечастые посетители сайта магазина), попадут в один кластер. Такой результат может оказаться нежелательным. С другой стороны, если использовать расстояние, основанное на корреляции, то в одном кластере окажутся покупатели с одинаковыми предпочтениями (т. е. те, кто купил товары A и B, но никогда не покупал C и D), даже если некоторые покупатели с такими предпочтениями совершают очень много покупок в сравнении с другим покупателями. Следовательно, для этой конкретной задачи расстояние, основанное на корреляции, может оказаться более подходящим.

Помимо тщательного выбора меры различия, необходимо также подумать о том, следует ли перед ее вычислением масштабировать все переменные таким образом, чтобы их стандартные отклонения стали равны 1. Для иллюстрации этого аспекта мы продолжим рассматривать только что описанный пример с интернет–магазином. Некоторые товары могут быть популярнее других; например, покупатель мог бы приобретать десять пар носков в год, но при этом очень редко заказывать компьютеры. В связи с этим часто покупаемые товары вроде носков будут оказывать большее влияние на различия между товарами, а следовательно, и на ре-

зультат кластеризации, чем редко покупаемые товары вроде компьютеров. Такой результат может оказаться нежелательным. Если же перед вычислением различий между наблюдениями масштабировать все переменные таким образом, чтобы их стандартные отклонения стали равны 1, то вклад каждой переменной в получаемое решение задачи иерархической кластеризации, по сути, становится одинаковым. Подобное масштабирование переменных может потребоваться также в случае, когда они выражаются в разных единицах, иначе выбор шкалы измерения для той или иной переменной (например, сантиметры вместо километров) окажет значительное влияние на получаемое значение меры различия. Конечно, решение о выполнении масштабирования переменных перед вычислением меры различия зависит также от характера решаемой практической задачи. На рис. 10.14 приведен пример. Как видим, решение о масштабировании переменных перед выполнением кластерного анализа является важным также и для метода K средних.

10.3.3 Практические аспекты применения кластеризации

Кластеризация может оказаться очень полезным инструментом при решении задач обучения без учителя. Однако при выполнении кластеризации возникает ряд затруднений. Рассмотрим эти затруднения.

Небольшие решения с большими последствиями

При выполнении кластерного анализа необходимо принять ряд решений:

- Следует ли выполнить определенную стандартизацию переменных или наблюдений? Возможно, например, что переменные следует центрировать и масштабировать таким образом, чтобы их средние значения стали равны 0, а стандартные отклонения — 1.

- В случае с иерархической кластеризацией:

  — Какую меру различия выбрать?

  — Какой тип присоединения выбрать?

  — На какой высоте рассекать дендрограмму для получения кластеров?

- В случае с кластеризацией по методу K средних сколько кластеров нам следует искать в данных?

Каждое из этих решений может оказать большое влияние на получаемые результаты. На практике мы обычно проверяем несколько разных вариантов, пытаясь найти наиболее полезное или интерпретируемое решение. Простого ответа не существует ни для одного из этих методов — рассматривать следует любое решение, которое обнажает интересные свойства данных.

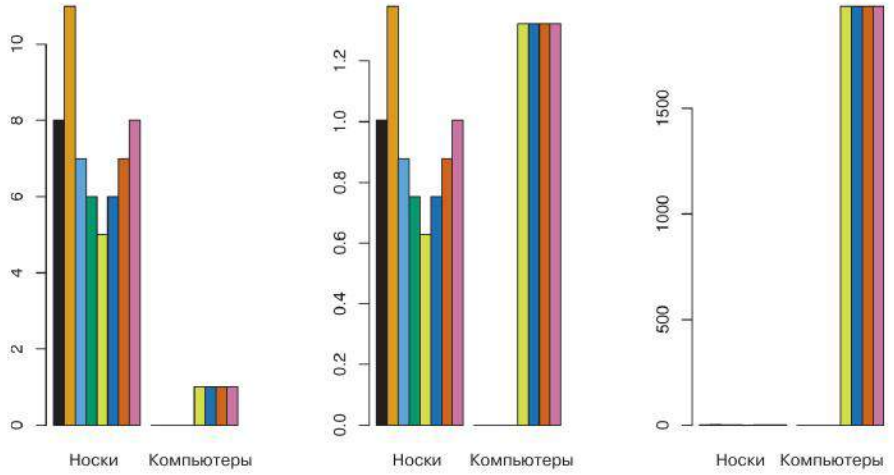

РИСУНОК 10.14. *Необычный интернет–магазин продает два вида товаров: носки и компьютеры.* Слева: *показано количество покупок носков и компьютеров, совершенных восемью покупателями. Каждый покупатель выделен отдельным цветом. Если выразить различия между наблюдениями при помощи евклидова расстояния, рассчитанного по исходным значениям переменных, то полученные различия в основном будут определяться количеством покупок носков, а количество покупок компьютеров будет оказывать очень слабый эффект. Такая ситуация может быть нежелательной, поскольку: (1) компьютеры являются более дорогими, чем носки, в связи с чем продавец может быть больше заинтересован в подталкивании посетителей магазина к покупке компьютеров, а не носков; (2) большая разница в количестве носков, купленных двумя покупателями, может оказаться менее информативной в отношении покупательских предпочтений этих людей, чем небольшое различие в количестве купленных компьютеров.* В центре: *показаны те же данные, но после деления каждой переменной на ее стандартное отклонение. Теперь количество приобретенных компьютеров будет оказывать гораздо большее влияние на уровень различий между наблюдениями.* Справа: *показаны те же данные, но теперь по оси Y отложены общие суммы (в долларах), потраченные каждым покупателем на носки и компьютеры. Поскольку компьютеры гораздо дороже носков, то полученные различия между наблюдениями будут определяться историей покупок компьютеров*

Проверка качества полученных кластеров

Мы всегда найдем какие–то кластеры, выполняя кластеризацию на том или ином наборе данных. Поэтому важно знать, соответствуют ли найденные кластеры реальным группам в данных или это просто результат *кластеризации содержащегося в данных шума.* Например, если у нас в распоряжении окажется независимый набор наблюдений, образуют ли эти наблюдения тот же набор кластеров? Ответить на этот вопрос

нелегко. Существует целый ряд методов, позволяющих присвоить кластеру p–значение для описания вероятности того, что этот кластер получен неслучайно. Однако мнения в отношении наиболее оптимального метода для решения этой задачи расходятся. Более подробное обсуждение данной проблемы можно найти в работе Hastie et al. (2009).

Другие аспекты кластерного анализа

И метод K средних, и иерархическая кластеризация отнесут каждое наблюдение к определенному кластеру. Однако иногда такой результат может оказаться неприемлемым. Предположим, например, что большинство наблюдений в действительности принадлежит к небольшому (и неизвестному) количеству групп, а остальная часть наблюдений существенно отличается как друг от друга, так и от большинства других наблюдений. Поскольку метод K средних и иерархическая кластеризация заставляют *каждое* наблюдение быть частью того или иного кластера, то найденные кластеры могут оказаться очень неоднородными из–за наличия выбросов, которые на самом деле не принадлежат ни к одному из этих кластеров. Привлекательным альтернативным подходом, позволяющим учесть наличие подобных выбросов, является применение смешанных моделей[10]. По сути, эти модели эквивалентны *мягкой* версии метода K средних, описанной в книге Hastie et al. (2009).

Кроме того, методы кластеризации обычно не очень устойчивы к возмущениям в данных. Предположим, например, что мы выполняем кластеризацию n наблюдений, а затем повторяем эту процедуру после удаления нескольких случайно выбранных наблюдений. Хотелось бы надеяться, что полученные два набора кластеров будут довольно похожими, однако часто это совсем не так!

Здравый подход к интерпретации результатов кластеризации

Мы описали несколько трудностей, связанных с выполнением кластерного анализа. Однако при правильном использовании кластеризация может быть очень полезным и эффективным инструментом. Мы уже отметили, что небольшие решения, определяющие ход выполнения кластеризации (например, необходимость стандартизации и выбор типа присоединения), могут оказать большое влияние на результаты. Поэтому мы рекомендуем выполнять кластеризацию с разными сочетаниями этих параметров и рассматривать всю совокупность результатов с целью обнаружения повторяющихся закономерностей. Поскольку кластеризация может давать неустойчивые решения, мы рекомендуем выполнять анализ на нескольких частях данных, чтобы получить представление об устойчивости получаемых кластеров. И самое главное: мы должны быть осторожны с представлением результатов кластерного анализа. Эти результаты не должны восприниматься как абсолютная истина в отношении данных. Вместо этого они должны служить начальной точкой для разработки той или иной научной гипотезы и последующего исследования, которое в идеале должно выполняться на независимом наборе данных.

[10] В оригинале используется термин «mixture models». — *Прим. пер.*

10.4 Лабораторная работа 1: анализ главных компонент

В этой лабораторной работе мы выполним PCA на примере набора данных USArrests, который поставляется вместе с R. В строках этой таблицы данных в алфавитном порядке перечислены 50 штатов.

```
> states = row.names(USArrests)
> states
```

В столбцах таблицы представлены четыре переменные:

```
> names(USArrests)
[1] "Murder" "Assault" "UrbanPop" "Rape"
```

Сначала мы кратко познакомимся со свойствами этих данных. Как видим, средние значения переменных существенно различаются:

```
> apply(USArrests, 2, mean)
 Murder  Assault  UrbanPop   Rape
   7.79   170.76     65.54  21.23
```

Обратите внимание на то, что функция apply() позволяет нам применить любую другую функцию (в данном случае mean()) к каждой строке или каждому столбцу таблицы. Второй параметр здесь указывает на то, как мы хотим вычислять среднее значение — по строкам (1) или по столбцам (2). Как мы видим, число изнасилований в среднем в три раза превышает число убийств, а число нападений в восемь раз превышает число изнасилований. С помощью функции apply() мы можем также вычислить дисперсии этих четырех переменных:

```
> apply(USArrests, 2, var)
 Murder  Assault  UrbanPop   Rape
   19.0   6945.2     209.5   87.7
```

Неудивительно, что эти переменные имеют и очень разные дисперсии: переменная UrbanPop содержит значения доли городского населения, которая несравнима с числом изнасилований в расчете на 100 000 жителей в каждом штате. Если бы мы не масштабировали переменные перед выполнением PCA, то большинство полученных главных компонент определялось бы переменной Assault, поскольку ее среднее значение и дисперсия намного превышают таковые у других переменных. Следовательно, перед выполнением PCA важно стандартизовать переменные таким образом, чтобы их средние значения стали равны 0, а стандартные отклонения — 1.

prcomp()

Теперь мы выполним анализ главных компонент при помощи функции prcomp(), которая является одной из нескольких функций в R для реализации PCA.

```
> pr.out = prcomp(USArrests, scale = TRUE)
```

По умолчанию функция prcomp() центрирует переменные, чтобы их средние значения стали равны 0. При помощи опции scale = TRUE мы масштабируем переменные так, чтобы их стандартные отклонения стали равны 1. Результаты работы prcomp() включают ряд полезных величин.

```
> names(pr.out)
[1] "sdev"  "rotation" "center"  "scale"  "x"
```

Элементы `center` и `scale` соответствуют средним значениям и стандартным отклонениям, использованным для стандартизации переменных перед выполнением PCA.

```
> pr.out$center
 Murder  Assault  UrbanPop   Rape
   7.79   170.76     65.54  21.23
> pr.out$scale
 Murder  Assault  UrbanPop   Rape
   4.36    83.34     14.47   9.37
```

В матрице `rotation` хранятся нагрузки главных компонент; каждый столбец `pr.out$rotation` содержит соответствующий вектор нагрузок[11].

```
> pr.out$rotation
             PC1    PC2    PC3    PC4
Murder    -0.536  0.418 -0.341  0.649
Assault   -0.583  0.188 -0.268 -0.743
UrbanPop  -0.278 -0.873 -0.378  0.134
Rape      -0.543 -0.167  0.818  0.089
```

Как видим, имеются четыре отдельные главные компоненты. Этого следовало ожидать, поскольку в целом в наборе данных с n наблюдениями и p переменными существует $\min(n-1, p)$ информативных главных компонент.

Используя функцию `prcomp()`, нам нет необходимости самостоятельно умножать данные на векторы нагрузок главных компонент для получения векторов со значениями главных компонент. Векторы значений главных компонент представлены в виде столбцов матрицы `x`, которая имеет размер 50×4. Другими словами, k–й столбец представляет собой вектор значений k–й компоненты.

```
> dim(pr.out$x)
[1] 50  4
```

Мы можем изобразить первые две главные компоненты следующим образом:

```
> biplot(pr.out, scale = 0)
```

Аргумент `scale = 0` функции `biplot()` позволяет убедиться в том, что стрелки на графике масштабированы для отображения нагрузок главных компонент; другие значения `scale` дают графики с несколько отличающейся интерпретацией.

Заметьте, что этот график является зеркальным отражением рис. 10.1. Вспомните, что главные компоненты уникальны только с точностью до знака, и поэтому мы можем воспроизвести рис. 10.1, внеся лишь небольшие изменения:

[11] Данная функция называет эту матрицу матрицей вращения («rotation matrix») в связи с тем, что умножение матрицы $\mathbf{X}$ на `pr.out$rotation` дает нам координаты наблюдений в повернутой системе координат. Эти координаты представляют собой значения главных компонент.

```
> pr.out$rotation = -pr.out$rotation
> pr.out$x = -pr.out$x
> biplot(pr.out, scale = 0)
```

Функция prcomp() возвращает также стандартное отклонение каждой главной компоненты. Так, в случае с набором данных USArrests мы можем получить эти стандартные отклонения следующим образом:

```
> pr.out$sdev
[1] 1.575 0.995 0.597 0.416
```

Дисперсию, объясненную каждой главной компонентой, получают путем возведения этих значений в квадрат:

```
> pr.var = pr.out$sdev^2
> pr.var
[1] 2.480 0.990 0.357 0.173
```

Для вычисления доли дисперсии, объясненной каждой главной компонентой, мы просто делим дисперсию, объясненную отдельной компонентой, на общую дисперсию, объясненную всеми компонентами.

```
> pve = pr.var / sum(pr.var)
> pve
[1] 0.6201 0.2474 0.0891 0.0434
```

Как видим, первая главная компонента объясняет 62.0% содержащейся в данных дисперсии, следующая за ней компонента объясняет 27.4% и т. д. Мы можем изобразить PVE каждой компоненты, а также кумулятивную PVE следующим образом:

```
> plot(pve, xlab = "Principal Component",
       ylab = "Proportion of Variance Explained",
       ylim = c(0, 1), type = "b")
> plot(cumsum(pve), xlab = "Principal Component",
       ylab = "Cumulative Proportion of Variance Explained",
       ylim = c(0, 1), type = "b")
```

cumsum() Результат показан на рис. 10.4. Заметьте, что функция cumsum() вычисляет накопленную сумму элементов некоторого числового вектора. Например:

```
> a = c(1, 2, 8, -3)
> cumsum(a)
[1] 1  3  11  8
```

10.5 Лабораторная работа 2: кластерный анализ

10.5.1 Кластеризация по методу K средних

kmeans() В R кластеризацию по методу K средних выполняет функция kmeans(). Мы начнем с простого примера, в котором имитированные данные образуют два кластера: среднее значение первых 25 наблюдений смещено относительно среднего значения следующих 25 наблюдений.

```
> set.seed(2)
> x = matrix(rnorm(50 * 2), ncol = 2)
> x[1:25, 1] = x[1:25, 1] + 3
> x[1:25, 2] = x[1:25, 2] - 4
```

Теперь мы выполним кластеризацию по методу K средних с $K = 2$.

```
> km.out = kmeans(x, 2, nstart = 20)
```

Метки кластеров для 50 наблюдений содержатся в векторе `km.out$cluster`.

```
> km.out$cluster
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
[30] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Кластеризация по методу K средних идеально разделила наблюдения на две группы, даже несмотря на то, что мы не предоставили функции `kmeans()` никакой информации по этим группам. Мы можем изобразить данные, выделяя каждое наблюдение отдельным цветом в соответствии с кластером, к которому оно принадлежит.

```
> plot(x, col = (km.out$cluster + 1),
       main = "K-Means Clustering Results with K = 2",
       xlab = "", ylab = "", pch = 20, cex = 2)
```

В данном случае наблюдения можно легко представить графически, поскольку они обладают двумя измерениями. Если бы число переменных превышало 2, то мы могли бы применить PCA и изобразить векторы значений первых двух компонент.

В этом примере мы знали, что в действительности было два кластера, поскольку мы сами создали эти данные. Однако при работе с реальными данными истинное число кластеров нам обычно неизвестно. В рассматриваемом примере мы могли бы выполнить кластеризацию по методу K средних, используя $K = 3$.

```
> set.seed(4)
> (km.out = kmeans(x, 3, nstart = 20))
K-means clustering with 3 clusters of sizes 10, 23, 17

Cluster means:
        [,1]         [,2]
1  2.3001545  -2.69622023
2 -0.3820397  -0.08740753
3  3.7789567  -4.56200798

Clustering vector:
[1] 3 1 3 1 3 3 3 1 3 1 3 1 3 1 3 1 3 3 3 3 3 1 3 3 3 2 2 2 2
    2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2

Within cluster sum of squares by cluster:
[1]  19.56137  52.67700  25.74089
(between_SS / total_SS = 79.3 %)
```

```
Available components:
[1] "cluster "      "centers"    "totss"   "withinss "
    "tot.withinss"  "betweenss"  "size"
> plot(x, col = (km.out$cluster + 1),
       main = "K-Means Clustering Results with K=3",
       xlab = "", ylab = "", pch = 20, cex = 2)
```

При $K = 3$ метод K средних приводит к расщеплению двух кластеров.

Для того чтобы запустить алгоритм функции `kmeans()` несколько раз с разными начальными метками кластеров, мы используем аргумент `nstart`. Если значение `nstart` превышает 1, то кластеризация по методу K средних будет выполнена с неоднократным случайным присваиванием меток кластеров на первом шаге алгоритма 10.1, после чего функция `kmeans()` выдаст наилучший результат. Ниже мы сравниваем использование `nstart = 1` с `nstart = 20`.

```
> set.seed(3)
> km.out = kmeans(x, 3, nstart = 1)
> km.out$tot.withinss
[1] 104.3319
> km.out = kmeans(x, 3, nstart = 20)
> km.out$tot.withinss
[1] 97.9793
```

Заметьте, что `km.out$tot.withinss` представляет собой общую внутрикластерную сумму квадратов, которую мы пытаемся минимизировать при помощи метода K средних (см. уравнение (10.11)). Отдельные внутрикластерные суммы квадратов хранятся в векторе `km.out$withinss`.

Мы *настоятельно* рекомендуем всегда выполнять кластеризацию по методу K средних, используя некоторое большое значение `nstart` (например, 20 или 50), иначе можно будет получить нежелательный локальный минимум целевой функции.

При выполнении кластеризации по методу K средних, помимо неоднократного присваивания исходных меток кластеров, важно также задать некоторое зерно генератора случайных чисел с помощью функции `set.seed()`. Благодаря этому присваивание меток кластеров на первом шаге алгоритма можно будет повторить, и результат кластеризации по методу K средних окажется полностью воспроизводимым.

10.5.2 Иерархическая кластеризация

`hclust()`

Иерархическую кластеризацию в `R` выполняет функция `hclust()`. В следующем примере мы воспользуемся данными из подраздела 10.5.1, чтобы построить иерархическую дендрограмму на основе полного, одиночного и среднего присоединения, применяя евклидово расстояние в качестве меры различия. Начнем с кластеризации наблюдений на основе полного присоединения. Функция `dist()` используется для вычисления матрицы размером 50×50, содержащей значения евклидова расстояния между наблюдениями.

`dist()`

```
> hc.complete = hclust(dist(x), method = "complete")
```

Не менее легко мы могли бы выполнить иерархическую кластеризацию на основе среднего или одиночного присоединения:

```
> hc.average = hclust(dist(x), method = "average")
> hc.single = hclust(dist(x), method = "single")
```

Теперь мы можем изобразить полученные дендрограммы при помощи обычной функции `plot()`. Числа в основании графика соответствуют порядковым номерам наблюдений.

```
> par(mfrow = c(1, 3))
> plot(hc.complete, main ="Complete Linkage",
      xlab = "", sub = "", cex = .9)
> plot(hc.average, main = "Average Linkage",
      xlab = "", sub = "", cex = .9)
> plot(hc.single, main ="Single Linkage",
      xlab = "", sub = "", cex = .9)
```

Для получения меток кластеров, возникающих в результате рассечения дендрограммы на той или иной высоте, мы можем воспользоваться функцией `cutree()`: `cutree()`

```
> cutree(hc.complete, 2)
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
[30] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
> cutree(hc.average, 2)
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
[30] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2
> cutree(hc.single, 2)
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
[30] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

В случае с этими данными полное и среднее присоединения в целом распределяют наблюдения по правильным группам. В то же время одиночное присоединение выделяет одно наблюдение в самостоятельный кластер. Более разумный результат получается при выборе четырех кластеров, хотя и в этом случае все еще остаются два кластера, каждый из который включает только одно наблюдение.

```
> cutree(hc.single, 4)
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 3 3 3
[30] 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3
```

Для масштабирования переменных перед выполнением иерархической кластеризации наблюдений мы применяем функцию `scale()`. `scale()`

```
> xsc = scale(x)
> plot(hclust(dist(xsc), method = "complete"),
      main = "Hierarchical Clustering with Scaled Features")
```

Для вычисления расстояния, основанного на корреляции, можно воспользоваться функцией `as.dist()`, конвертирующей любую квадратную симметричную матрицу в форму, которая распознается функцией `hclust()` как матрица расстояний. Однако это имеет смысл только для `as.dist()`

данных, которые содержат как минимум три переменные, поскольку абсолютное значение корреляции между любыми двумя переменными, описанными по двум признакам, всегда равно 1. Поэтому мы выполним кластеризацию трехмерного набора данных.

```
> x = matrix(rnorm(30 * 3), ncol = 3)
> dd = as.dist(1 - cor(t(x)))
> plot(hclust(dd, method = "complete"),
      main = "Complete Linkage with Correlation-Based Distance",
      xlab = "", sub = "")
```

10.6 Лабораторная работа 3: анализ данных NCI60

Методы обучения без учителя часто используются в ходе анализа геномных данных. В частности, популярными инструментами являются PCA и иерархическая кластеризация. Мы проиллюстрируем эти методы на примере набора данных NCI60, полученного при помощи технологии ДНК-чипов и включающего результаты измерений уровня экспрессии 6830 генов в 64 раковых клеточных линиях.

```
> library(ISLR)
> nci.labs = NCI60$labs
> nci.data = NCI60$data
```

Каждая клеточная линия имеет идентификатор типа ракового заболевания. Мы не используем информацию о типе рака при выполнении PCA и кластерного анализа, поскольку это методы обучения без учителя. Однако после выполнения PCA и кластерного анализа мы проверим, в какой степени результаты использования этих двух методов обучения без учителя совпадают с имеющимися типами рака.

Данные содержат 64 строки и 6830 столбцов.

```
> dim(nci.data)
[1]  64  6830
```

Начнем с выяснения типов рака, представленных имеющимися клеточными линиями.

```
> nci.labs[1:4]
[1] "CNS"   "CNS"   "CNS"   "RENAL"
> table(nci.labs)
nci.labs
   BREAST          CNS        COLON  K562A-repro  K562B-repro
        7            5            7            1            1
 LEUKEMIA  MCF7A-repro  MCF7D-repro     MELANOMA        NSCLC
        6            1            1            8            9
  OVARIAN     PROSTATE        RENAL      UNKNOWN
        6            2            9            1
```

10.6.1 Применение PCA к данным NCI60

Сначала мы применим к этим данным анализ главных компонент, предварительно выполнив масштабирование переменных (генов), чтобы приравнять их стандартные отклонения к единице (хотя в данном случае можно было бы резонно утверждать, что необходимости в подобном масштабировании нет).

```
> pr.out = prcomp(nci.data, scale = TRUE)
```

Теперь мы изобразим векторы значений первых нескольких главных компонент с целью визуализации данных. Наблюдения (клеточные линии), соответствующие определенному типу рака, будут выделены одним цветом, что позволит нам увидеть степень их сходства. Сначала мы создадим небольшую функцию, которая будет присваивать определенный цвет каждому элементу числового вектора. Эта функция поможет присвоить цвета каждой из 64 клеточных линий, исходя из типа рака, к которому они принадлежат.

```
> Cols = function(vec){cols = rainbow(length(unique(vec)))
                       return(cols[as.numeric(as.factor(vec))])}
```

Заметьте, что в качестве аргумента функция `rainbow()` принимает некоторое положительное целое число и возвращает вектор с соответствующим количеством цветовых меток. Теперь мы можем построить график с векторами значений главных компонент.

`rainbow()`

```
> par(mfrow = c(1, 2))
> plot(pr.out$x[, 1:2], col = Cols(nci.labs),
       pch = 19, xlab = "Z1", ylab = "Z2")
> plot(pr.out$x[, c(1, 3)], col = Cols(nci.labs),
       pch = 19, xlab = "Z1", ylab = "Z3")
```

Полученные графики приведены на рис. 10.15. В целом клеточные линии, соответствующие определенному типу рака, обладают похожими векторами значений первых нескольких главных компонент. Это указывает на то, что клеточные линии, принадлежащие к одному типу рака, обычно имеют похожие уровни экспрессии генов.

Мы можем выяснить долю дисперсии (PVE), объясненной несколькими первыми главными компонентами, применив к объекту `prcomp` метод `summary()` (для экономии места приводится только часть результатов):

```
> summary(pr.out)
Importance of components :
                          PC1     PC2     PC3     PC4     PC5
Standard deviation     27.853 21.4814 19.8205 17.0326 15.9718
Proportion of Variance  0.114  0.0676  0.0575  0.0425  0.0374
Cumulative Proportion   0.114  0.1812  0.2387  0.2812  0.3185
```

Воспользовавшись функцией `plot()`, мы можем также построить график доли дисперсии, объясненной первыми несколькими главными компонентами.

```
> plot(pr.out)
```

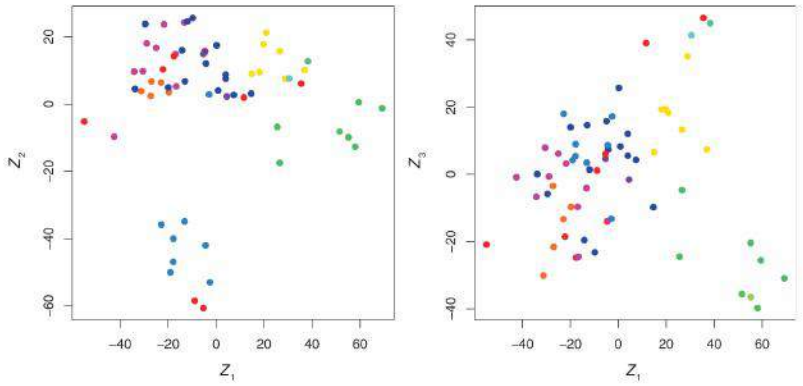

РИСУНОК 10.15. *Проекция раковых клеточных линий NCI60 на первые три главные компоненты (т. е. значения первых трех главных компонент). В целом наблюдения, принадлежащие к одному типу рака, находятся близко друг к другу в этом пространстве малой размерности. Визуализация этих данных без использования метода снижения размерности (вроде PCA) была бы невозможна, поскольку для исходного набора данных существует* $\binom{6830}{2}$ *возможных диаграмм рассеяния, ни одна из которых не была бы сколь–либо информативной*

Обратите внимание на то, что высота каждого столбца на этой диаграмме получена путем возведения в квадрат соответствующего элемента `pr.out$sdev`. Однако более информативной является визуализация PVE каждой главной компоненты (т. е. построение графика «каменистой осыпи») и кумулятивной PVE каждой компоненты. Много для этого делать не придется:

```
> pve = 100*pr.out$sdev^2 / sum(pr.out$sdev^2)
> par(mfrow = c(1, 2))
> plot(pve, type = "o", ylab = "PVE",
       xlab = "Principal Component", col = "blue")
> plot(cumsum(pve), type = "o", ylab = "Cumulative PVE",
       xlab = "Principal Component", col = "brown3")
```

(Заметьте, что элементы `pve` можно сразу получить из сводной информации по модели — `summary(pr.out)$importance[2, ]`, а элементы `cumsum(pve)` хранятся в `summary(pr.out)$importance[3, ]`). Полученные графики приведены на рис. 10.16. Видно, что первые семь главных компонент в сумме объясняют около 40% заключенной в данных дисперсии. Это не очень большая величина. Однако при анализе графика «каменистой осыпи» мы видим, что, несмотря на небольшую долю дисперсии, объясненной первыми семью главными компонентами, имеет место значительное снижение доли дисперсии, объясненной последующими компонентами. Другими словами, примерно после седьмой компоненты на этом графике наблюдается *излом*. Это указывает на то, что использование более семи главных компонент вряд ли будет полезным (при этом анализ даже этих семи главных компонент может оказаться непростым).

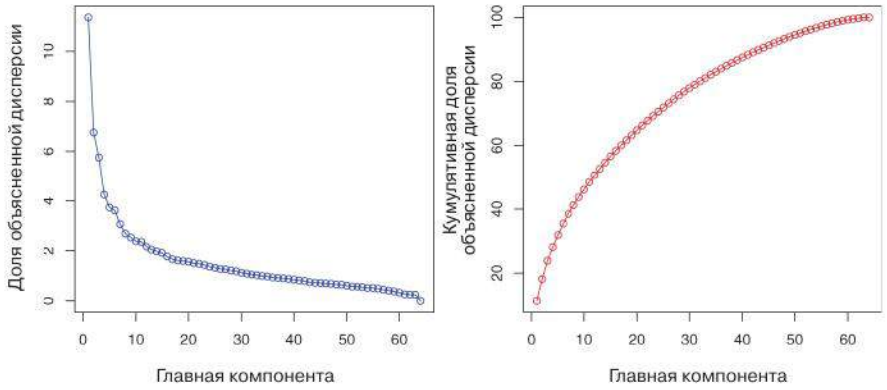

РИСУНОК 10.16. *PVE главных компонент набора данных* `NCI60`. Слева: *показаны PVE каждой главной компоненты.* Справа: *приведены кумулятивные PVE каждой главной компоненты. Все вместе главные компоненты объясняют 100% дисперсии*

10.6.2 Кластеризация наблюдений из набора данных NCI60

Перейдем теперь к иерархической кластеризации клеточных линий из набора данных `NCI60` и выясним, образуют ли эти наблюдения кластеры в соответствии с типом ракового заболевания. Для начала мы выполним стандартизацию переменных таким образом, чтобы их средние значения стали равны 0, а стандартные отклонения — 1. Как было отмечено ранее, этот шаг не является обязательным и должен выполняться только если мы хотим, чтобы экспрессия каждого гена выражалась в одинаковых единицах.

```
> sd.data = scale(nci.data)
```

Теперь мы выполним иерархическую кластеризацию наблюдений на основе полного, одиночного и среднего типов присоединения. В качестве меры различия используется евклидово расстояние.

```
> data.dist = dist(sd.data); par(mfrow = c(1, 3))

> plot(hclust(data.dist), labels = nci.labs,
       main = "Complete Linkage", xlab = "", sub = "", ylab="")

> plot(hclust(data.dist, method = "average"), labels = nci.labs,
       main = "Average Linkage", xlab = "", sub = "", ylab = "")

> plot(hclust(data.dist, method = "single"), labels = nci.labs,
       main = "Single Linkage", xlab = "", sub = "", ylab = "")
```

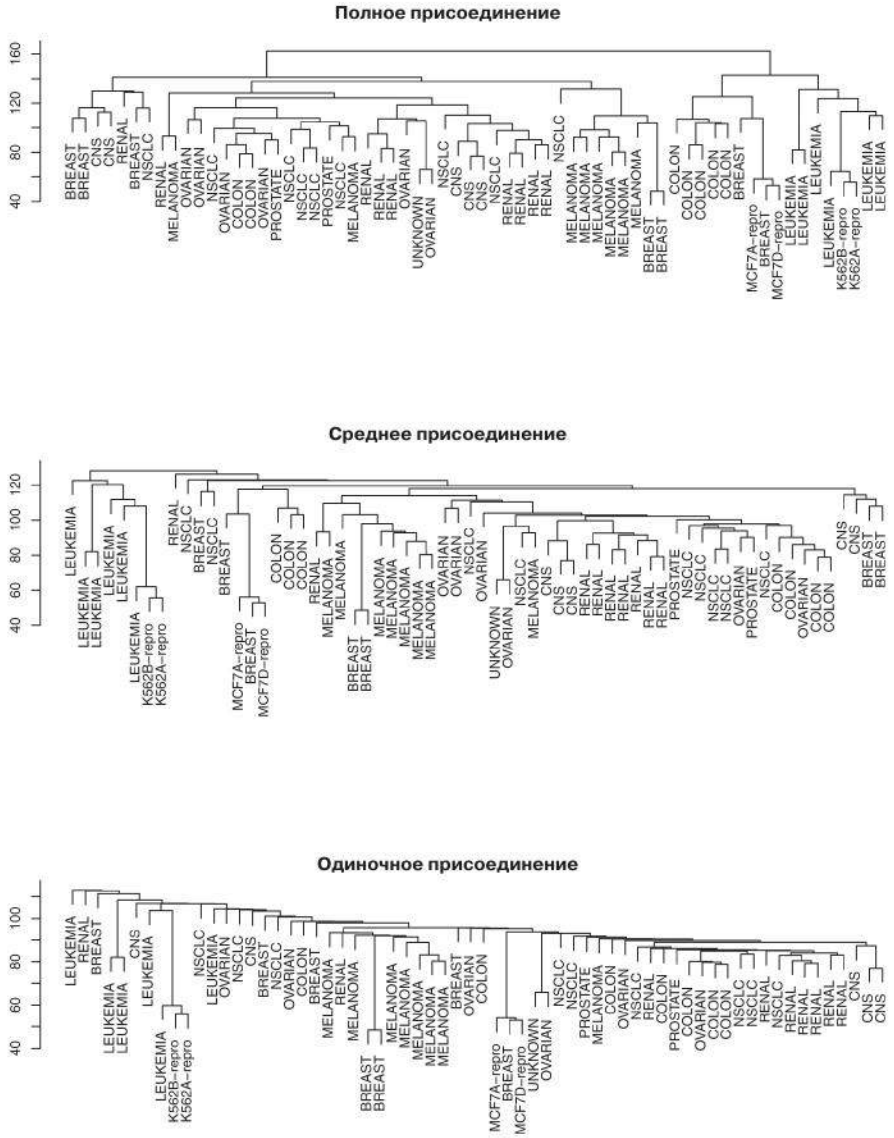

РИСУНОК 10.17. *Данные по раковым клеточным линиям NCI60, кластеризованные на основе полного, среднего и одиночного типов присоединения с использованием евклидова расстояния в качестве меры различия. Полное и среднее присоединения обычно дают сбалансированные кластеры, тогда как одиночное присоединение часто дает большие кластеры, в которых отдельные листья сливаются один за другим*

Результаты приведены на рис. 10.17. Мы видим, что выбор типа присоединения несомненно оказывает влияние на получаемые результаты. Обычно одиночное присоединение будет давать *стелющиеся* кластеры[12]: очень большие кластеры, к которым отдельные наблюдения присоединяются одно за другим. С другой стороны, полное и среднее присоединения обычно дают более сбалансированные и привлекательные кластеры. По этой причине полное и среднее присоединения обычно предпочтительнее одиночного присоединения. Хорошо видно, что клеточные линии, принадлежащие к одному типу рака, действительно имеют тенденцию входить в один кластер, хотя эта кластеризация и не идеальна. В описанном ниже анализе мы воспользуемся иерархической кластеризацией на основе полного присоединения.

Мы можем рассечь дендрограмму на высоте, которая обеспечит определенное количество кластеров — например, четыре:

```
> hc.out = hclust(dist(sd.data))
> hc.clusters = cutree(hc.out, 4)
> table(hc.clusters, nci.labs)
```

Наблюдаются определенные закономерности. Все клеточные линии, соответствующие лейкемии, попадают в кластер 3, тогда как клеточные линии, соответствующие раку груди, распределены по трем различным кластерам. Мы можем изобразить на дендрограмме линию рассечения, дающую эти четыре кластера:

```
> par(mfrow = c(1, 1))
> plot(hc.out, labels = nci.labs)
> abline (h = 139, col = "red")
```

Функция `abline()` добавляет прямую линию поверх любого существующего графика в `R`. Аргумент `h = 139` позволяет нарисовать горизонтальную линию на дендрограмме на высоте 139 — это высота, на которой выделяются четыре самостоятельных кластера. Можно легко проверить, что эти кластеры не отличаются от тех, которые мы получили при помощи `cutree(hc.out, 4)`.

Вывод на экран результатов работы `hclust()` дает полезную сводку по соответствующему объекту:

```
> hc.out

Call:
hclust(d = dist(dat))

Cluster method   : complete
Distance         : euclidean
Number of objects: 64
```

Ранее в подразделе 10.3.2 мы утверждали, что кластеризация по методу K средних и иерархическая кластеризация, в которой дендрограмма рассекается для получения того же числа кластеров, могут дать очень разные результаты. Как результаты иерархической кластеризации данных

[12] В оригинале используется термин «trailing clusters». — *Прим. пер.*

NCI60 соотносятся с результатами кластеризации по методу K средних при $K = 4$?

```
> set.seed(2)
> km.out = kmeans(sd.data, 4, nstart = 20)
> km.clusters = km.out$cluster
> table(km.clusters, hc.clusters)
             hc.clusters
km. clusters   1   2   3   4
           1  11   0   0   9
           2   0   0   8   0
           3   9   0   0   0
           4  20   7   0   0
```

Как видим, четыре кластера, полученные с использованием иерархической кластеризации и метода K средних, несколько различаются. Кластер 2 в кластеризации по методу K средних идентичен кластеру 3 в иерархической кластеризации. Однако остальные кластеры различаются: например, кластер 4 в решении по методу K средних содержит часть наблюдений, отнесенных к кластеру 1 в иерархической кластеризации, а также все наблюдения, отнесенные в иерархической кластеризации к кластеру 2.

Вместо того чтобы выполнять иерархическую кластеризацию по всей матрице данных, мы можем выполнить ее просто по векторам значений первых нескольких главных компонент:

```
> hc.out = hclust(dist(pr.out$x[, 1:5]))
> plot(hc.out, labels = nci.labs,
       main = "Hier. Clust. on First Five Score Vectors")
> table(cutree(hc.out, 4), nci.labs)
```

Неудивительно, что эти результаты отличаются от тех, которые были получены при выполнении иерархической кластеризации на полном наборе данных. Иногда выполнение кластеризации с использованием векторов значений первых нескольких главных компонент может дать более приемлемые результаты, нежели кластеризация на полном наборе данных. В такой ситуации мы можем рассматривать предварительное применение PCA как шаг, позволяющий удалить шум из данных. Кластеризацию на основе векторов значений первых нескольких главных компонент, а не всего набора данных, можно было бы выполнить также по методу K средних.

10.7 Упражнения

Теоретические

1. Эта задача касается алгоритма кластеризации по методу K средних.

   (a) Докажите уравнение (10.12).

   (b) Исходя из этого доказательства, приведите доводы в пользу того, что на каждой итерации алгоритм кластеризации по методу K средних (алгоритм 10.1) уменьшает целевую функцию (10.11).

2. Предположим, что у нас есть четыре наблюдения, для которых мы рассчитали следующую матрицу со значениями парных различий:

$$\begin{bmatrix} & 0.3 & 0.4 & 0.7 \\ 0.3 & & 0.5 & 0.8 \\ 0.4 & 0.5 & & 0.45 \\ 0.7 & 0.8 & 0.45 & \end{bmatrix}.$$

Например, различие между первым и вторым наблюдениями составляет 0.3, а между вторым и четвертым наблюдениями — 0.8.

(a) Используя эту матрицу различий, изобразите набросок дендрограммы, которую можно получить в результате иерархической кластеризации этих четырех наблюдений на основе полного присоединения. Убедитесь, что вы указали на рисунке высоту, на которой происходит каждое слияние, а также наблюдения, соответствующие каждому из листьев дендрограммы.

(b) Повторите (a), но с использованием одиночного присоединения.

(c) Предположим, что мы рассекли дендрограмму из (a) и получили два кластера. Какие наблюдения попали в каждый их этих кластеров?

(d) Предположим, что мы рассекли дендрограмму из (b) и получили два кластера. Какие наблюдения попали в каждый их этих кластеров?

(e) Как отмечалось в этой главе, в каждой точке слияния на дендрограмме позиции двух объединяемых кластеров можно поменять местами без каких–либо последствий для интерпретации дендрограммы. Нарисуйте дендрограмму, эквивалентную дендрограмме из (a), изменив в ней позиции двух или более наблюдений, но сохранив при этом ее смысл.

3. В этой задаче вы «вручную» выполните кластеризацию по методу K средних с $K = 2$ для небольшого примера, где есть $n = 6$ наблюдений и $p = 2$ признака. Наблюдения выглядят следующим образом:

Наблюдение	X_1	X_2
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

(a) Изобразите эти наблюдения графически.

(b) Случайным образом присвойте метку кластера каждому наблюдению. Вы можете сделать это в `R` при помощи команды `sample()`. Какая метка была присвоена каждому наблюдению?

(c) Вычислите центроиды для каждого кластера.

(d) Отнесите каждое наблюдение в кластер, чей центроид находится к нему ближе всего (по евклидову расстоянию). Какая новая метка была присвоена каждому наблюдению?

(e) Повторяйте (c) и (d) до тех пор, пока ответы не перестанут изменяться.

(f) На своем рисунке из (a) закрасьте точки в соответствии с полученными метками кластеров.

4. Предположим, что для некоторого набора данных мы выполнили иерархическую кластеризацию на основе одиночного присоединения и полного присоединения. В итоге мы получили две дендрограммы.

   (a) В определенной точке на дендрограмме, полученной на основе одиночного присоединения, кластеры $\{1, 2, 3\}$ и $\{4, 5\}$ объединяются. На дендрограмме, полученной на основе полного присоединения, кластеры $\{1, 2, 3\}$ и $\{4, 5\}$ также объединяются в определенной точке. Какое из этих слияний произойдет на большей высоте? Или они произойдут на одинаковой высоте? Или у нас недостаточно информации для ответа на этот вопрос?

   (b) В определенной точке на дендрограмме, полученной на основе одиночного присоединения, кластеры $\{5\}$ и $\{6\}$ объединяются. На дендрограмме, полученной на основе полного присоединения, кластеры $\{5\}$ и $\{6\}$ также объединяются в определенной точке. Какое из этих слияний произойдет на большей высоте? Или они произойдут на одинаковой высоте? Или у нас недостаточно информации для ответа на этот вопрос?

5. Опишите словами результаты, которые вы ожидали бы получить, выполнив кластеризацию по методу K средних с $K = 2$ для восьми покупателей на рис. 10.14 на основе количества совершенных ими покупок носков и компьютеров. Дайте три ответа — по одному для каждого из показанных на рисунке способов масштабирования. Поясните свои ответы.

6. Исследователь получил измерения уровня экспрессии 1000 генов в 100 образцах тканей. Данные можно представить в виде матрицы $\mathbf{X}$ размером 1000×100, в которой строки соответствуют генам, а столбцы — образцам тканей. Каждый образец ткани был получен в отдельный день, в связи с чем более ранние образцы располагаются левее, а более поздние образцы — правее. Эти образцы тканей принадлежат к двум группам: контрольной (К) и экспериментальной (Э). В ходе эксперимента К– и Э–образцы получали в случайном порядке. Исследователь желает установить различия между контрольной и экспериментальной группами по уровню экспрессии каждого гена.

   В качестве предварительного анализа данных (т. е. до проведения сравнения групп К и Э) исследователь выполняет анализ главных

компонент и выясняет, что первая главная компонента (вектор длиной 100) имеет четко выраженный тренд в направлении слева направо и объясняет 10% дисперсии. После этого исследователь вспоминает, что образцы тканей каждого пациента обрабатывались на одной из двух машин — A и B и что машина A использовалась более часто в начале эксперимента, а машина B — в конце эксперимента. У исследователя есть информация относительно того, какая машина применялась для обработки каждого образца.

(a) Объясните значение утверждения о том, что первая главная компонента «объясняет 10% дисперсии».

(b) Исследователь решает заменить (i, j)–й элемент в $\mathbf{X}$ на

$$x_{ji} - \phi_{j1} z_{i1},$$

где z_{i1} — это i–е значение первой главной компоненты, а ϕ_{j1} — это j–й коэффициент ее нагрузки. Затем исследователь собирается применить двухвыборочный t–тест к каждому гену для установления значимости различий между двумя группами тканей. Приведите свои критические замечания относительно этой идеи и предложите более оптимальный подход.

(c) Разработайте схему выполнения и реализуйте небольшой имитационный эксперимент, подтверждающий верность вашей стратегии анализа.

Практические

7. В этой главе мы упомянули использование евклидова расстояния и расстояния, основанного на корреляции, в качестве мер различия для иерархической кластеризации. Оказывается, что эти две меры почти эквивалентны: если стандартизовать каждую переменную таким образом, чтобы ее среднее значение стало равно 0, а стандартное отклонение — 1, и обозначить корреляцию между i–м и j–м наблюдениями как r_{ij}, то величина $1 - r_{ij}$ будет пропорциональна квадрату евклидова расстояния между наблюдениями i и j.

   Используя данные `USArrests`, покажите, что эта пропорциональность действительно существует.

   Подсказка: евклидово расстояние можно вычислить при помощи функции `dist()`*, а коэффициенты корреляции — при помощи функции* `cor()`.

8. В подразделе 10.2.3 в уравнении (10.8) была приведена формула для вычисления PVE. Мы видели также, что PVE можно получить при помощи объекта `sdev`, создаваемого функцией `prcomp()`.

   Используя данные `USArrests`, вычислите PVE двумя способами:

   (a) Используя объект `sdev`, создаваемый функцией `prcomp()` (см. подраздел 10.2.3).

(b) Непосредственно используя уравнение (10.8). Другими словами, примените функцию `prcomp()` для вычисления нагрузок главных компонент. Далее подставьте полученные нагрузки в уравнение (10.8) для вычисления PVE.

Эти два подхода должны дать одинаковые результаты.

Подсказка: вы получите одинаковые ответы в (a) и (b) только, если будете использовать одинаковые данные в обоих случаях. Например, если в (a) вы применили `prcomp()` *к центрованным и масштабированным переменным, то вы также должны выполнить центрирование и масштабирование переменных перед применением уравнения (10.8) в (b).*

9. Обратимся к данным `USArrests`. Сейчас мы выполним иерархическую кластеризацию штатов.

(a) Выполните иерархическую кластеризацию штатов, используя полное присоединение и евклидово расстояние.

(b) Выполните рассечение дендрограммы на высоте, которая приводит к выделению трех кластеров. Какие штаты входят в каждый из этих кластеров?

(c) Выполните иерархическую кластеризацию штатов на основе полного присоединения и евклидова расстояния *после масштабирования переменных, в результате которого их стандартные отклонения становятся равными 1.*

(d) Какой эффект на полученную иерархическую кластеризацию оказывает масштабирование переменных? На ваш взгляд, следует ли выполнять масштабирование переменных перед вычислением различий между наблюдениями? Обоснуйте свой ответ.

10. В этой задаче вы создадите искусственный набор данных, а затем примените к нему PCA и кластеризацию по методу K средних.

(a) Создайте набор данных с 20 наблюдениями в каждом из трех классов (т. е. всего 60 наблюдений) и 50 переменными.

Подсказка: в `R` *есть несколько функций, которые можно использовать для создания имитированных данных. Один из примеров — это функция* `rnorm()`*; функция* `runif()` *— это еще один вариант. Убедитесь, что вы выполняете смещение среднего значения наблюдений в каждом классе, позволяющее создать хорошо отличающиеся друг от друга классы.*

(b) Примените PCA к 60 наблюдениям и изобразите на графике векторы значений первых двух главных компонент. Используйте отдельный цвет для наблюдений из каждого класса. Если три класса хорошо разделяются на этом графике, то переходите к пункту (c). Если же нет, то вернитесь к пункту (a) и модифицируйте процедуру создания данных, чтобы получить более выраженное разделение между тремя классами. Не переходите к пункту (c) до тех пор, пока не появится хотя бы небольшое разделение трех классов по векторам значений первых двух главных компонент.

(c) Выполните иерархическую кластеризацию наблюдений по методу K средних с $K = 3$. Насколько хорошо полученные кластеры согласуются с истинными метками классов?

Подсказка: в `R` *для сравнения истинных меток классов с метками кластеров вы можете применить функцию* `table()`. *Соблюдайте осторожность при интерпретации результатов: метод K средних присвоит произвольные номера кластерам, в связи с чем вы не сможете просто сравнить истинные метки классов с метками полученных кластеров.*

(d) Выполните кластеризацию по методу K средних с $K = 2$. Опишите свои результаты.

(e) Теперь выполните кластеризацию по методу K средних с $K = 4$ и опишите свои результаты.

(f) Теперь выполните кластеризацию по методу K средних с $K = 3$, используя векторы значений первых двух главных компонент, а не исходные данные. Другими словами, примените метод K средних к матрице размером 60×2, в которой первый столбец соответствует вектору значений первой главной компоненты, а второй столбец — вектору значений второй главной компоненты. Прокомментируйте полученные результаты.

(g) Используя функцию `scale()`, выполните кластеризацию по методу K средних с $K = 3$ *после масштабирования, в результате которого стандартные отклонения всех переменных становятся равными 1.* Как эти результаты соотносятся с результатами, полученными в (b)? Объясните, что вы видите.

11. На сайте этой книги (www.StatLearning.com) есть набор данных (`Ch10Ex11.csv`) со значениями уровня экспрессии 1000 генов в 40 образцах тканей. Первые 20 образцов получены от здоровых людей, а следующие 20 — от группы людей с определенным заболеванием.

(a) Загрузите данные при помощи функции `read.csv()`. Вам нужно будет воспользоваться аргументом `header = FALSE`.

(b) Примените иерархическую кластеризацию к образцам тканей, используя расстояние, основанное на корреляции, и изобразите дендрограмму. Разделяются ли эти образцы на две группы по уровням экспрессии генов? Зависят ли получаемые результаты от типа использованного присоединения?

(c) Ваш коллега желает знать, какие гены различаются в этих двух группах больше всего. Предложите способ получения ответа на этот вопрос и выполните соответствующий анализ.

Предметный указатель

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «АЛЬЯНС БУКС» наложенным платежом, выслав открытку или письмо по почтовому адресу: 123242, Москва, а/я 20 или по электронному адресу: orders@alians-kniga.ru.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в интернет-магазине: www.alians-kniga.ru.

Оптовые закупки: тел. (499) 782-38-89; электронный адрес books@alians-kniga.ru.

Гарет Джеймс, Даниела Уиттон,
Тревор Хасти, Роберт Тибширани

Введение в статистическое обучение
с примерами на языке R

Главный редактор Мовчан Д. А.
dmkpress@gmail.com
Перевод Мастицкий С. Э.
Корректор Синяева Г. И.
Верстка Мастицкий С. Э.
Дизайн обложки Мовчан А. Г.

Формат 70×100 1/16.
Гарнитура «Уорнок». Печать офсетная.
Усл. печ. л. 43. Тираж 200 экз.

Веб-сайт издательства: www.dmkpress.com