Машинное обучение: линейные и вероятностные модели

Учебное пособие

Поляк Марк Дмитриевич

2027

Аннотация

Рассматриваются основы машинного обучения на примере линейных и вероятностных моделей: постановка задачи обучения по прецедентам, подготовка данных, линейная регрессия и метод наименьших квадратов, градиентный спуск, переобучение и валидация, регуляризация, логистическая регрессия, метрики качества классификации и наивный байесовский классификатор. Формулы разбираются по частям и проверяются на примерах, посчитанных вручную; приведены листинги на языке Python с библиотекой scikit-learn, вопросы для самоконтроля и задачи с ответами. Предназначено для студентов бакалавриата инженерных направлений в области информационных технологий, изучающих машинное обучение, а также для разработчиков, осваивающих предмет самостоятельно.

Содержание

Предисловие **5**

Основные обозначения **7**

1 Задача обучения по прецедентам **9**
1.1 Что такое машинное обучение 9
1.2 Объекты, признаки и ответы 10
1.3 Типы задач 11
1.4 Модель и метод обучения 12
1.5 Функция потерь и эмпирический риск 14
1.6 Постановка задачи на практике 17
Вопросы для самоконтроля 17
Задачи 17

2 Данные и признаки **19**
2.1 Таблица данных 19
2.2 Разведочный анализ 19
2.3 Пропуски 21
2.4 Категориальные признаки 21
2.5 Масштабирование 22
2.6 Конструирование признаков 23
2.7 Утечки данных 24
Вопросы для самоконтроля 26
Задачи 26

3 Линейная регрессия и метод наименьших квадратов **27**
3.1 Задача регрессии 27
3.2 Один признак 28
3.3 Много признаков 30
3.4 Геометрический смысл 32
3.5 Мультиколлинеарность 33
3.6 Полиномиальная регрессия 36
Вопросы для самоконтроля 38
Задачи 38

4 Градиентный спуск **39**
4.1 Зачем нужны итерации 39
4.2 Интуиция на одном параметре 39
4.3 Алгоритм 41

4.4 Градиентный спуск для линейной регрессии 42
4.5 Стохастический и мини-пакетный спуск 43
4.6 Практические приёмы 45
Вопросы для самоконтроля 46
Задачи 47

5 Переобучение и валидация **48**
5.1 Недообучение и переобучение 48
5.2 Обобщающая способность 49
5.3 Смещение и разброс 50
5.4 Схемы валидации 52
5.5 Метрики качества регрессии 54
5.6 Подбор гиперпараметров 55
Вопросы для самоконтроля 57
Задачи 57

6 Регуляризация **58**
6.1 Штраф за большие веса 58
6.2 Гребневая регрессия 59
6.3 LASSO 60
6.4 Геометрический смысл 61
6.5 Регуляризация на практике 64
Вопросы для самоконтроля 66
Задачи 66

7 Логистическая регрессия **67**
7.1 Почему не подходит линейная регрессия 67
7.2 Линейный классификатор 67
7.3 Отступ и функции потерь 68
7.4 Сигмоида и вероятность класса 70
7.5 Максимум правдоподобия 72
7.6 Вероятностный смысл регуляризации 74
7.7 Несколько классов 75
Вопросы для самоконтроля 76
Задачи 76

8 Метрики качества классификации **77**
8.1 Функция потерь и метрика качества 77
8.2 Матрица ошибок 78
8.3 Точность и полнота 80
8.4 Порог, ROC-кривая и AUC 81

8.5 Кривая точность–полнота и логарифмическая потеря . . . 84
8.6 Выбор метрики . . . 84
Вопросы для самоконтроля . . . 85
Задачи . . . 86

9 Наивный байесовский классификатор 87
9.1 Формула Байеса . . . 87
9.2 Оптимальный байесовский классификатор . . . 88
9.3 Наивное предположение . . . 88
9.4 Гауссовский наивный байес . . . 89
9.5 Тексты и мешок слов . . . 90
9.6 Сглаживание Лапласа . . . 92
9.7 Наивный байес и логистическая регрессия . . . 93
Вопросы для самоконтроля . . . 95
Задачи . . . 95

Заключение 96

Ответы к задачам 97

Список литературы 99

Приложение А. Математический минимум 100
А.1 Векторы и матрицы . . . 100
А.2 Производная и градиент . . . 101
А.3 Вероятность . . . 102
А.4 Экспонента и логарифм . . . 104

Приложение Б. Русско-английский словарь терминов 105

Предисловие

Машинное обучение стало частью инженерной практики: модели предсказывают дефекты в коде, фильтруют письма, оценивают стоимость и риски. Чтобы пользоваться готовыми библиотеками осмысленно — выбирать модель и метрику, замечать переобучение и утечки данных, понимать, почему модель ошибается, — нужно понимать, что происходит внутри них. Эта книга объясняет основы машинного обучения на линейных и вероятностных моделях. Они просты, но на них видны все главные идеи: обучение как минимизация эмпирического риска, градиентный спуск, переобучение и способы его обнаружить, регуляризация, вероятностный смысл функций потерь, метрики качества. Те же идеи лежат в основе более сложных моделей.

Текст рассчитан на студентов-бакалавров инженерных направлений в области информационных технологий, в частности «Программной инженерии», изучающих курс основ машинного обучения, а также на разработчиков, которые осваивают предмет самостоятельно. Для понимания материала нужна математика в объёме первых курсов технического университета — векторы и матрицы из линейной алгебры, производная и градиент из математического анализа, начала теории вероятностей, — а также умение программировать на Python. Необходимые сведения из математики собраны в приложении А.

Построение глав. Главный принцип — объяснять формулы простыми словами. Новое понятие вводится через задачу, которую оно решает; затем следует формула и её разбор по частям — абзацы «Читаем формулу». Почти каждая идея проверяется на примере из нескольких объектов, который можно посчитать вручную; полезно действительно пересчитать его, прежде чем читать дальше. Короткие листинги показывают, как то же самое делается в библиотеке scikit-learn. Глава заканчивается вопросами для самоконтроля и задачами с ответами.

Главы 1 и 2 вводят основные понятия: объекты, признаки, модель, функцию потерь, эмпирический риск, подготовку данных. Главы 3–6 на примере задачи регрессии строят инструменты, общие для всех моделей: метод наименьших квадратов, градиентный спуск, валидацию и регуляризацию. Главы 7–9 посвящены классификации: логистической регрессии, метрикам качества классификаторов и наивному байесовскому классификатору. Главы опираются на предыдущие, поэтому читать их лучше по порядку. Сквозные примеры — предсказание ошибок в модулях программы (гл. 1 и 8), оценка стоимости квартир на синтетических данных (гл. 2, 6) и фильтрация спама (гл. 9).

Метрические методы, метод опорных векторов, решающие деревья и

ансамбли, обучение без учителя и нейронные сети здесь не рассматриваются; понятия этой книги — основа для их изучения.

Основные обозначения

Латинские буквы набраны курсивом, векторы и матрицы — прямым полужирным шрифтом. Обозначения, которые нужны только в одной главе, вводятся там же.

X, Y — множество объектов и множество ответов;

$\varphi : X \to Y$ — неизвестная зависимость, которую восстанавливает модель;

x, x_i — объект; i-й объект выборки;

$f_j(x)$ — j-й признак объекта, $j = 1, \dots, n$; $f_0(x) = 1$ — фиктивный признак;

$x_{ij} = f_j(x_i)$ — значение j-го признака на i-м объекте;

$\mathbf{x}_i$ — признаковое описание объекта x_i (вектор значений его признаков);

n, ℓ — число признаков и число объектов обучающей выборки;

$X^\ell = (x_i, y_i)_{i=1}^{\ell}$ — обучающая выборка; X^k — тестовая выборка из k объектов;

$\mathbf{F}$ — матрица объекты–признаки: ℓ строк (объекты), столбцы — признаки;

$\mathbf{y}$ — вектор ответов $(y_1, \dots, y_\ell)^{\mathrm{T}}$;

$\tilde{y} \in \{0, 1\}$ — метка класса в записи 0/1, $\tilde{y} = (y + 1)/2$ для $y \in \{-1, +1\}$;

K — число классов;

$\theta = (\theta_0, \dots, \theta_n)$ — вектор параметров модели;

$g(x, \theta)$ — модель: прогноз ответа на объекте x;

$\hat{y} = g(x, \theta)$ — прогноз модели;

θ^* — оптимальные параметры: решение задачи обучения;

$h(x, \theta)$ — разделяющая функция классификатора: $g = \operatorname{sign} h$;

$\mathcal{M}$ — метод обучения: по выборке находит параметры, $\mathcal{M}(X^\ell) = \theta$;

$\mathcal{L}$ — функция потерь (ошибка модели на одном объекте);

$Q(\theta, X^\ell)$ — функционал качества; чаще всего — эмпирический риск (средняя потеря на выборке);

∇Q — градиент функции Q по параметрам;

η — скорость обучения в градиентном спуске;

λ — коэффициент регуляризации;

M_i — отступ объекта x_i;

$\sigma(z)$ — сигмоида (с аргументом); σ, σ_j без аргумента — стандартное (среднеквадратическое) отклонение;

$P(\cdot)$, $p(\cdot)$ — вероятность и плотность распределения;

$\hat{p} = P(\tilde{y} = 1 \mid x)$ — вероятность класса 1, предсказанная моделью для объекта x; $\hat{p}_i$ — для объекта x_i;

$\mathbb{E}(\cdot)$, $\mathrm{D}(\cdot)$ — математическое ожидание и дисперсия;

$[A]$ — индикатор: 1, если утверждение A истинно, иначе 0;
$\langle \mathbf{a}, \mathbf{b} \rangle$ — скалярное произведение векторов;
$\|\theta\|_2$, $\|\theta\|_1$ — нормы вектора: евклидова и сумма модулей;
$\mathbf{A}^{\mathrm{T}}$ — транспонированная матрица;
s_j — сингулярные числа матрицы $\mathbf{F}$;
$\varkappa$ — число обусловленности матрицы;
$\mathbf{I}$ — единичная матрица.

1. Задача обучения по прецедентам

В этой главе вводятся основные понятия: объект, признак, ответ, модель, функция потерь и эмпирический риск. Обучение модели сводится к задаче оптимизации — найти параметры, при которых средняя ошибка на известных примерах минимальна; остальные главы по-разному решают именно эту задачу.

1.1. Что такое машинное обучение

Перед выпуском новой версии программы тестировщики не успевают одинаково тщательно проверить все модули программы. Хочется заранее знать, в каких модулях ошибки наиболее вероятны. Обычно пишут правила, например: «модуль подозрителен, если в нём больше тысячи строк и за месяц его меняли больше десяти раз». Но пороги в таких правилах взяты на глаз, а когда проект изменится, правила устареют.

Машинное обучение предлагает другой подход. В системе контроля версий и трекере задач хранится история: для каждого модуля прошлых версий известно, сколько в нём было строк, как часто его меняли и нашли ли в нём ошибки после выпуска новой версии программного продукта. Компьютер получает эти примеры и сам находит зависимость между характеристиками модуля и наличием ошибок. Результат обучения — программа (модель), которая выдаёт прогноз для нового модуля. Разница подходов показана на рис. 1.1: в традиционном программировании человек пишет правила, а компьютер выдаёт ответы; в машинном обучении человек даёт примеры с ответами, а компьютер находит правила.

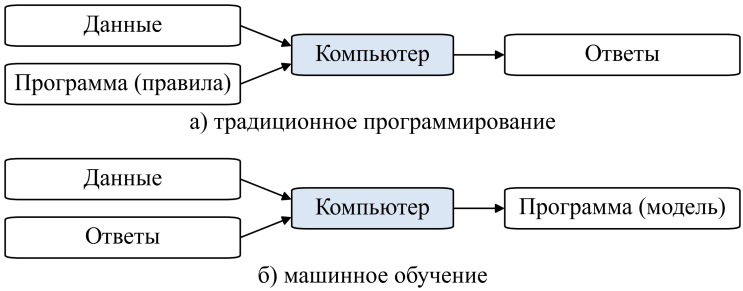


Рис. 1.1. Традиционное программирование и машинное обучение

Для инженера удобно определение Т. Митчелла [1]: программа *обу-*

чается на опыте E относительно задачи T и меры качества P, если её качество в задаче T, измеренное мерой P, растёт с накоплением опыта E.

Чтобы воспользоваться этим определением, нужно явно указать все три составляющие. В примере с модулями программы задача T — предсказать, будет ли в модуле найдена ошибка; опыт E — модули прошлых версий программы, для которых уже известно, нашли ли в них ошибки после выпуска; мера качества P — например, доля верных прогнозов. Пока не названы T, E и P, задача обучения не поставлена.

Машинное обучение — раздел искусственного интеллекта, но не синоним «разумной машины»: это математические методы восстановления зависимостей по примерам. Модель знает о предметной области только то, что содержится в данных.

1.2. Объекты, признаки и ответы

Перейдём к формальной постановке. Обозначим через X множество *объектов* — то, о чём мы делаем прогноз (модули программы, заявки на кредит, квартиры), а через Y — множество *ответов* (есть ли ошибка, вернёт ли заёмщик кредит, сколько стоит квартира); объект $x \in X$, ответ $y \in Y$. Считается, что существует неизвестная зависимость $\varphi : X \to Y$, и для каждого объекта x его ответ $y = \varphi(x)$.

Сама зависимость неизвестна, зато известны ответы на конечном наборе объектов — *обучающей выборке*

$$X^\ell = (x_i, y_i)_{i=1}^{\ell}, \quad y_i = \varphi(x_i),$$

где ℓ — число объектов в выборке. Пары (x_i, y_i) называют прецедентами, отсюда название задачи и этой главы. Требуется построить функцию, которая приближает φ не только на объектах выборки, но и на новых объектах: запомнить известные ответы легко, трудно не ошибаться на объектах, которых модель не видела.

Компьютер не умеет работать с модулем программы или квартирой напрямую. Объект описывают набором *признаков* — функций $f_j : X \to D_j$, $j = 1, \dots, n$, каждая из которых сопоставляет объекту одно значение из множества D_j. По множеству значений различают четыре типа признаков:

- бинарный, $D_j = \{0, 1\}$: есть ли у модуля автоматические тесты;
- номинальный, D_j конечно и не упорядочено: язык программирования, на котором написан модуль;
- порядковый, D_j конечно и упорядочено: приоритет компонента

программы (низкий, средний, высокий);
- количественный, $D_j = \mathbb{R}$: число строк кода, цикломатическая сложность, число изменений за месяц.

Тип признака определяет, какие действия с ним осмысленны: количества строк кода в разных модулях можно складывать, языки программирования — нет; приоритеты можно сравнивать, но не вычитать. Как перевести признаки всех типов в числа, описано в гл. 2.

Набор значений всех признаков объекта x_i называют его *признаковым описанием* и записывают вектором $\mathbf{x}_i = (f_1(x_i), \dots, f_n(x_i))$. Значение j-го признака на i-м объекте для краткости обозначают $x_{ij} = f_j(x_i)$. Важно различать объект x_i (модуль программы) и его признаковое описание $\mathbf{x}_i$ (вектор чисел): модель машинного обучения видит объект только через признаки.

Описания всех объектов выборки складывают в *матрицу объекты–признаки*

$$\mathbf{F} = \big(f_j(x_i)\big)_{\ell \times n} = \begin{pmatrix} f_1(x_1) & \dots & f_n(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_\ell) & \dots & f_n(x_\ell) \end{pmatrix}.$$

Строка матрицы — один объект, столбец — один признак. Ответы образуют вектор $\mathbf{y} = (y_1, \dots, y_\ell)^{\mathrm{T}}$. В программах на Python матрицу объекты–признаки обычно называют `X`, а вектор ответов — `y`. Пример выборки из пяти модулей приведён в табл. 1.1.

Таблица 1.1. Обучающая выборка: модули программы

Модуль	Строк кода, f_1	Цикломатическая сложность, f_2	Изменений за месяц, f_3	Есть тесты, f_4	Ответ y
x_1	120	4	2	1	-1
x_2	850	23	15	0	$+1$
x_3	400	12	3	1	-1
x_4	1500	31	9	1	$+1$
x_5	300	9	11	0	$+1$

Здесь $y = +1$ означает, что после выпуска новой версии программы в модуле нашли ошибку, $y = -1$ — что не нашли. Матрица $\mathbf{F}$ для этой выборки имеет размер 5×4, а, например, $x_{22} = f_2(x_2) = 23$.

1.3. Типы задач

Тип задачи определяется множеством ответов Y.

Классификация — множество ответов конечно, каждый ответ означает класс. При $Y = \{-1, +1\}$ это бинарная классификация (пример с модулями); при $Y = \{1, \dots, K\}$ — классификация на K непересекающихся классов (определить компонент, к которому относится сообщение об ошибке). Если объект может принадлежать сразу нескольким классам, ответ записывают вектором из нулей и единиц, $Y = \{0, 1\}^K$: сообщение об ошибке может касаться одновременно интерфейса и базы данных.

Регрессия — ответ числовой, $Y = \mathbb{R}$: сколько часов уйдёт на задачу, сколько стоит квартира. Если нужно предсказать сразу несколько чисел (например, координаты объекта), $Y = \mathbb{R}^m$.

Ранжирование — важен порядок объектов: какие документы показать выше в поисковой выдаче. На рис. 1.2 показаны две основные задачи: в классификации модель ищет границу между классами, в регрессии — кривую, проходящую ближе к точкам.

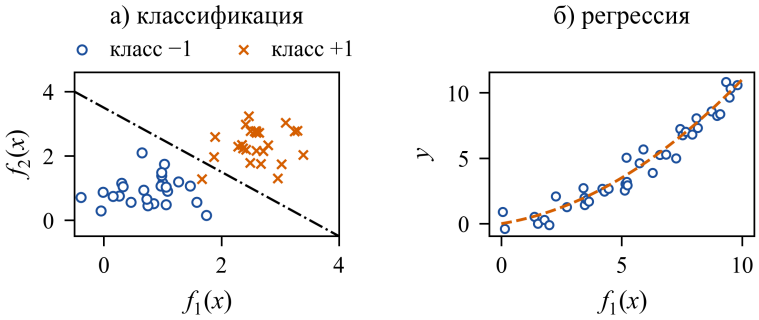


Рис. 1.2. Классификация и регрессия

Во всех этих задачах ответы на обучающих объектах известны — это *обучение с учителем*. Если ответов нет и требуется найти структуру в самих объектах (разбить на группы похожих, описать меньшим числом признаков), говорят об *обучении без учителя*; здесь оно не рассматривается.

1.4. Модель и метод обучения

Зависимость ищут в заранее выбранном семействе функций. *Модель* — это параметрическое семейство функций

$$A = \{g(x, \theta) \mid \theta \in \Theta\},$$

где g — фиксированная функция, а θ — вектор параметров из множества

допустимых значений Θ. Выбрать модель — значит выбрать вид функции; обучить модель — значит найти конкретные значения параметров.

Рассмотрим *линейную* модель. Для задачи регрессии она имеет вид

$$g(x, \theta) = \sum_{j=0}^{n} \theta_j f_j(x) = \theta_0 + \theta_1 f_1(x) + \ldots + \theta_n f_n(x). \tag{1.1}$$

Здесь добавлен фиктивный признак $f_0(x) = 1$, чтобы свободный член θ_0 записывался той же суммой, что и остальные слагаемые.

Читаем формулу. Каждый признак умножается на свой вес θ_j, и произведения складываются. Вес показывает, на сколько изменится прогноз, если признак увеличится на единицу, а остальные останутся прежними; знак веса говорит о направлении влияния. Свободный член θ_0 — прогноз для объекта, у которого все признаки равны нулю. Пусть, например, трудоёмкость задачи в часах оценивается по двум признакам: f_1 — число файлов, которые придётся изменить, $f_2 = 1$, если задача добавляет в программу новую функциональность, и 0 иначе. При параметрах $\theta = (2; 0{,}5; 3)$ задача, затрагивающая 10 файлов и добавляющая новую функциональность, оценивается в $2 + 0{,}5 \cdot 10 + 3 \cdot 1 = 10$ часов.

Сумма в (1.1) — скалярное произведение векторов параметров и признаков (включая $f_0 = 1$), её коротко записывают как $\langle \theta, \mathbf{x} \rangle$.

Для бинарной классификации с ответами ± 1 линейная модель получается из той же суммы взятием знака:

$$g(x, \theta) = \operatorname{sign} \sum_{j=0}^{n} \theta_j f_j(x).$$

Функция sign возвращает $+1$ для положительного аргумента и -1 для отрицательного, то есть одну из двух меток класса. Поэтому классы удобно обозначать -1 и $+1$.

Параметры находит *метод обучения* $\mathcal{M}$ — алгоритм, который по обучающей выборке вычисляет вектор параметров: $\mathcal{M}(X^\ell) = \theta$. На этапе *обучения* метод $\mathcal{M}$ подбирает параметры по выборке X^ℓ, на этапе *применения* модель вычисляет прогнозы $g(x, \theta)$ для новых объектов; в библиотеке scikit-learn этим этапам соответствуют методы `fit` и `predict`.

Кроме параметров, есть *гиперпараметры* — величины, которые задают до обучения и которые метод обучения сам не подбирает. Например, набор признаков: в задаче о модулях программы какие характеристики модуля учитывать (число строк кода, сложность, частоту изменений), решает человек, а веса при них находит метод $\mathcal{M}$. Другой пример — степень d, если зависимость ищут в виде многочлена $\theta_0 + \theta_1 x + \ldots + \theta_d x^d$:

коэффициенты θ_j — параметры, а d — гиперпараметр. Гиперпараметры подбирают, сравнивая качество моделей на данных, не участвовавших в обучении.

1.5. Функция потерь и эмпирический риск

Чтобы выбрать лучшие параметры, нужно уметь измерять, насколько модель ошибается. *Функция потерь* $\mathcal{L}(g, x)$ — это величина ошибки модели g на одном объекте x: она тем больше, чем сильнее прогноз $g(x, \theta)$ отличается от правильного ответа $y = \varphi(x)$. Самые распространённые функции потерь:

- для классификации — индикатор ошибки $\mathcal{L}(g, x) = [g(x, \theta) \neq y]$;
- для регрессии — абсолютная ошибка $\mathcal{L}(g, x) = |g(x, \theta) - y|$ и квадратичная ошибка $\mathcal{L}(g, x) = (g(x, \theta) - y)^2$.

Квадратные скобки обозначают индикатор: $[A] = 1$, если утверждение A истинно, и 0, если ложно.

Ошибку на одном объекте усредняют по всей выборке. Получается *эмпирический риск*:

$$Q(\theta, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}(g, x_i). \tag{1.2}$$

Читаем формулу. Знак суммы с индексом i от 1 до ℓ означает: пройти по всем объектам выборки и сложить их потери; множитель $1/\ell$ превращает сумму в среднее. С индикатором ошибки Q — доля неверных ответов, с квадратичной потерей — средний квадрат ошибки. Эмпирический риск зависит от параметров θ: изменили параметры — изменились прогнозы, а с ними и потери.

Теперь обучение можно записать как задачу оптимизации — найти параметры, при которых эмпирический риск минимален:

$$\mathcal{M}(X^\ell) = \arg\min_{\theta \in \Theta} Q(\theta, X^\ell). \tag{1.3}$$

Этот принцип называют *минимизацией эмпирического риска*. Запись $\arg\min$ означает не само наименьшее значение Q, а тот вектор θ, при котором оно достигается. Если модель линейная, а потеря квадратичная, получается метод наименьших квадратов, которому посвящена гл. 3.

Потеря и функционал качества. Величину, по которой при обучении выбирают параметры, называют *функционалом качества* и обозначают буквой Q — от англ. quality. Она сопоставляет каждой модели одно число — насколько хорошо модель описывает всю выборку. Слово

«функционал» означает, что аргумент Q — сама модель, то есть функция g. Когда вид модели выбран, модель задаётся параметрами, поэтому пишут $Q(\theta, X^\ell)$. Эмпирический риск — самый распространённый функционал качества, но не единственный: в вероятностных моделях, например, максимизируют вероятность получить наблюдаемые ответы.

Функцию потерь и функционал качества часто путают, тем более что названия у них похожи и в разных книгах разные (табл. 1.2); в англоязычной литературе функционал качества часто называют функцией стоимости (cost function) и обозначают буквой J. Различие простое: потеря — ошибка на *одном* объекте, функционал качества — *одно* число для всей выборки, обычно средняя потеря. Если ошибки модели на трёх объектах равны 1, -2 и 0, квадратичные потери — три числа, 1, 4 и 0, а эмпирический риск — одно число, $(1 + 4 + 0)/3 = 5/3$. Обучение оптимизирует именно функционал качества: уменьшить потерю на одном объекте легко, но ценой роста потерь на других.

Таблица 1.2. Функция потерь и функционал качества

Характеристика	Функция потерь	Функционал качества
Что измеряет	ошибку на одном объекте	качество модели на всей выборке
Обозначение	$\mathcal{L}(g, x_i)$	$Q(\theta, X^\ell)$, $J(\theta)$
Другие названия	потеря, loss	эмпирический риск, полная (средняя) потеря, функция стоимости; total loss, cost function, objective
Пример	$(g(x_i, \theta) - y_i)^2$	$\frac{1}{\ell}\sum_{i=1}^{\ell}(g(x_i, \theta) - y_i)^2$ — среднеквадратичная ошибка

Путаницу усиливают библиотеки: в программах словом loss часто называют уже усреднённое по выборке значение, то есть функционал качества. Встретив слово «потеря», полезно проверить, о чём речь — об одном объекте или обо всех сразу.

Пример. Выбор функции потерь — не формальность: от него зависит, какая модель окажется лучшей. Пусть выборка состоит из четырёх объектов с одним признаком (табл. 1.3), причём последний объект выбивается из общей закономерности. Сравним две модели: А: $g(x) = x$ и Б: $g(x) = 1{,}5x$ (рис. 1.3).

Таблица 1.3. Прогнозы двух моделей на четырёх объектах

x	y	Прогноз А	Ошибка А	Прогноз Б	Ошибка Б
1	1	1	0	1,5	0,5
2	2	2	0	3	1
3	3	3	0	4,5	1,5
4	8	4	−4	6	−2

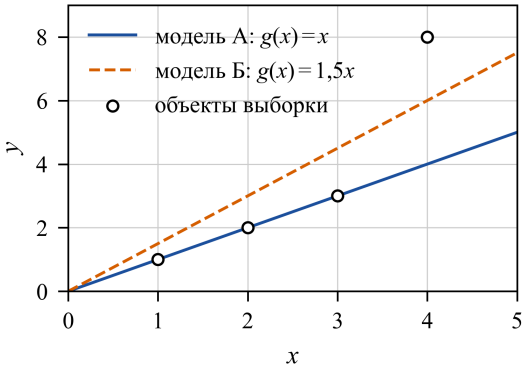


Рис. 1.3. Две модели на выборке с выбросом

С абсолютной потерей эмпирический риск модели А равен $(0 + 0 + 0 + 4)/4 = 1$, модели Б — $(0{,}5 + 1 + 1{,}5 + 2)/4 = 1{,}25$; лучше модель А. С квадратичной потерей риск модели А равен $(0+0+0+16)/4 = 4$, модели Б — $(0{,}25+1+2{,}25+4)/4 = 1{,}875$; лучше модель Б. Квадрат сильно увеличивает большие ошибки, поэтому квадратичная потеря «тянет» модель к выбросу, а абсолютная почти не обращает на него внимания. Какая модель правильнее, зависит от задачи: если выброс — ошибка измерения, предпочтительна модель А; если такие значения действительно встречаются и занижать прогноз для них дорого — модель Б.

Однако важна ошибка не на обучающей выборке, а на новых объектах. Среднюю потерю по всем возможным объектам называют *риском*; её нельзя вычислить, потому что неизвестны ни φ, ни распределение объектов. Обычно предполагают, что объекты выборки и будущие объекты получены независимо из одного распределения. Тогда средняя потеря на *новых* данных — разумная оценка риска, а эмпирический риск на обучающей выборке её занижает: параметры подобраны именно под эти объекты. Как оценить ошибку на новых объектах, рассказано в гл. 5.

1.6. Постановка задачи на практике

Постановка начинается с четырёх вопросов. Что предсказываем и какое решение примут по прогнозу (какие модули программы проверить первыми)? Какой это тип задачи (предсказание дефекта — классификация, оценка трудоёмкости — регрессия)? Какие данные доступны в момент прогноза и откуда взять ответы (метрики кода, история изменений, сведения об ошибках из трекера)? Как измерять качество? Мера качества должна отражать реальную цель: если пропустить модуль с ошибкой дороже, чем лишний раз его проверить, доля верных ответов — плохая мера.

Постановка неудачна, если признаки не связаны с ответом: по цвету подсветки синтаксиса в редакторе не предсказать ошибки в модуле, а по номеру квартиры не определить её цену. Никакой метод не извлечёт информацию, которой нет в данных, — «мусор на входе, мусор на выходе».

Обучение модели — лишь один этап решения прикладной задачи машинного обучения. Обычно работа над такой задачей включает следующие стадии: постановка задачи; сбор и подготовка данных (гл. 2); построение простой базовой модели для сравнения (baseline); выбор основной модели, её обучение и подбор гиперпараметров; проверка качества на данных, не участвовавших в обучении (гл. 5 и 8); внедрение; наблюдение за качеством в работе. Данные со временем меняются, модель устаревает, и цикл повторяется.

Вопросы для самоконтроля

1. Назовите задачу T, опыт E и меру качества P для задачи оценки трудоёмкости.
2. Чем объект отличается от его признакового описания?
3. Чем параметры модели отличаются от гиперпараметров? Приведите по два примера.
4. Почему эмпирический риск на обучающей выборке занижает ошибку модели на новых объектах?
5. Что меняется при замене абсолютной функции потерь на квадратичную?

Задачи

1. Для выборки из табл. 1.1 вычислите эмпирический риск с индикатором ошибки для классификатора $g(x) = \operatorname{sign}(f_2(x) - 10)$, который считает подозрительными модули со сложностью больше 10.

2. Для выборки из табл. 1.3 найдите параметр θ модели $g(x) = \theta x$, минимизирующий эмпирический риск с квадратичной потерей. *Указание:* приравняйте к нулю производную Q по θ.

2. Данные и признаки

В гл. 1 обучающая выборка представлена готовой матрицей чисел $\mathbf{F}$. На практике данные приходят в виде таблицы, где есть пропуски, текстовые значения и признаки в несопоставимых единицах. В этой главе показано, как превратить такую таблицу в матрицу объекты–признаки и не допустить при этом утечки данных — ошибки, из-за которой оценка качества модели оказывается завышенной, а на новых данных модель работает заметно хуже.

2.1. Таблица данных

Сквозной пример главы — оценка стоимости квартир. Объект — квартира, ответ — цена в миллионах рублей, задача — регрессия. Фрагмент выборки приведён в табл. 2.1; данные условные.

Таблица 2.1. Фрагмент выборки: квартиры

№	Площадь, м²	Район	Этаж	Год постройки	Цена, млн руб.
1	34	А	3	1975	6,1
2	52	Б	9	2012	9,0
3	41	В	1	1968	5,2
4	75	А	5	нет данных	14,8
5	63	Б	12	2019	11,3
6	47	В	4	2005	6,9

Строки таблицы — объекты, столбцы — признаки и ответ. Площадь, этаж и год — количественные признаки, район — номинальный, а для квартиры № 4 год постройки неизвестен. Прежде чем строить модель, нужно заполнить пропуск, превратить районы в числа и привести признаки к сопоставимому масштабу. Но сначала данные нужно изучить.

2.2. Разведочный анализ

Разведочный анализ данных — это знакомство с таблицей до построения модели. Обычно проверяют:

- типы и диапазоны признаков: нет ли отрицательной площади или этажа 250;
- распределения: симметричны ли они, есть ли выбросы;
- долю пропусков в каждом столбце;
- связи признаков с ответом и между собой;

- в задачах классификации — соотношение классов: если положительных объектов 1 %, модель, которая всегда отвечает «отрицательный», будет права в 99 % случаев.

Распределение цены квартир (рис. 2.1, а) несимметрично: большинство квартир стоит 5–10 млн руб., но есть длинный «хвост» дорогих. Так обычно распределены величины, которые получаются произведением многих множителей: цены, доходы, время выполнения задач. Если вместо цены предсказывать её логарифм, распределение становится почти симметричным (рис. 2.1, б), а модель меньше подстраивается под редкие дорогие квартиры. Прогноз цены получают обратным преобразованием $e^{\hat{y}}$.

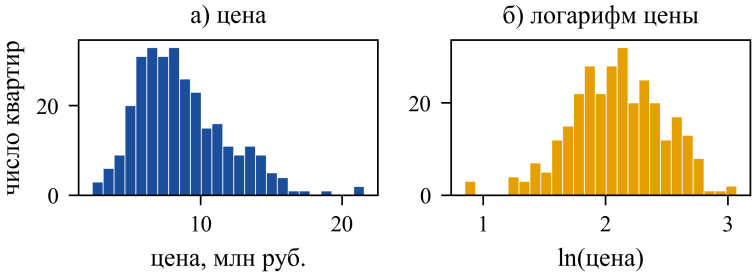


Рис. 2.1. Распределение цены квартиры и её логарифма (синтетические данные)

Силу линейной связи двух признаков (или признака и ответа) измеряет *коэффициент корреляции Пирсона*. Для значений u_i и v_i на ℓ объектах

$$r = \frac{\sum\limits_{i=1}^{\ell}(u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum\limits_{i=1}^{\ell}(u_i - \bar{u})^2 \sum\limits_{i=1}^{\ell}(v_i - \bar{v})^2}},$$

где $\bar{u}$ и $\bar{v}$ — средние значения.

Читаем формулу. В числителе для каждого объекта перемножаются отклонения двух величин от их средних. Если обе величины одновременно выше среднего или одновременно ниже, произведение положительно; если одна выше, а другая ниже — отрицательно. Сумма показывает, какое поведение преобладает. Знаменатель нормирует результат так, что $-1 \le r \le 1$: $r = 1$ — точная возрастающая линейная зависимость, $r = -1$ — убывающая, $r \approx 0$ — линейной связи нет (но нелинейная может быть). Сильно коррелированные между собой признаки — повод на-

сторожиться: они несут почти одну и ту же информацию, модели трудно разделить между ними вклад в ответ, и её веса становятся неустойчивыми. И наконец, корреляция говорит о связи, а не о причине. Например, в задаче из гл. 1 число найденных в модуле ошибок связано с числом тестов для него: чем больше тестов, тем больше ошибок находят. Но если удалить тесты, ошибок в модуле не станет меньше — их просто перестанут находить.

2.3. Пропуски

Пропуски возникают, когда данные не собрали, потеряли или они не существуют (у нового дома нет истории ремонтов). Основные способы с ними справиться:

- удалить объекты с пропусками — допустимо, если их мало, но если пропуски не случайны (например, год постройки неизвестен в основном у старых домов), выборка станет смещённой;
- удалить признак, если пропусков в нём большинство;
- заполнить пропуск средним, медианой или (для номинальных признаков) самым частым значением. Медиана устойчивее к выбросам: в табл. 2.1 медиана известных годов равна 2005, ею и заполняется пропуск у квартиры № 4;
- добавить бинарный признак «значение было пропущено». Сам факт пропуска бывает информативен: если продавец не указал год постройки, это может что-то говорить о доме.

Среднее и медиана для заполнения вычисляются по обучающей выборке; почему это важно, объяснено в разд. «Утечки данных».

2.4. Категориальные признаки

Линейная модель умеет только умножать признаки на веса и складывать, поэтому номинальный признак «район» нужно превратить в числа. Закодировать районы числами 1, 2, 3 нельзя: модель решит, что район В «вдвое дальше» от А, чем Б, и что влияние района на цену меняется равными шагами. Порядка и расстояний между районами нет, а кодирование их навязывает.

Правильный способ — *прямое кодирование* (one-hot): каждому значению признака соответствует свой бинарный признак, равный 1, если объект имеет это значение, и 0 иначе:

$$f_{\text{А}}(x) = [\text{район}(x) = \text{А}], \quad f_{\text{Б}}(x) = [\text{район}(x) = \text{Б}], \quad f_{\text{В}}(x) = [\text{район}(x) = \text{В}].$$

У каждой квартиры ровно один из трёх признаков равен единице, поэтому их сумма всегда равна 1 — так же, как фиктивный признак f_0. Значит, столбцы матрицы $\mathbf{F}$ линейно зависимы, и веса линейной модели определяются неоднозначно: если прибавить к θ_0 любое число и вычесть его из весов всех трёх районов, прогнозы не изменятся. Поэтому один из столбцов, например $f_{\text{А}}$, удаляют. Район А становится *базовым*: свободный член θ_0 относится к нему, а веса $\theta_{\text{Б}}$ и $\theta_{\text{В}}$ показывают, на сколько цена в районах Б и В отличается от А при прочих равных.

Порядковые признаки (приоритет «низкий», «средний», «высокий») можно кодировать числами 1, 2, 3 — порядок у них есть. Но и здесь числа предполагают равные шаги между уровнями; если это сомнительно, используют прямое кодирование.

Если значений у признака много (улица, автор коммита), прямое кодирование порождает тысячи столбцов. Тогда, например, заменяют значение средним ответом по объектам с этим значением: улицу — средней ценой квартиры на ней. Такое *кодирование средним* удобно, но опасно утечкой: если среднее посчитано с участием самого объекта, в признак попадает его собственный ответ.

2.5. Масштабирование

Признаки измеряются в разных единицах и принимают значения разного порядка: площадь — десятки квадратных метров, расстояние до метро — километры, а этаж — просто номер (рис. 2.2, а). Для самой линейной модели это не проблема: веса подстроятся под единицы измерения. Но разный масштаб признаков замедляет итерационные методы поиска параметров, искажает действие штрафа за большие веса, одинакового для всех признаков (гл. 6), и делает расстояние между объектами зависящим в основном от признака с самыми большими числами. Расстояние используют методы, которые относят объект к тому же классу, что и наиболее похожие на него объекты выборки (например, метод ближайших соседей). Для них разница площадей в 20 м² вносит в расстояние вчетверо больший вклад, чем разница расстояний до метро в 4,5 км, хотя на цену второе может влиять сильнее. Поэтому признаки приводят к общему масштабу.

Стандартизация вычитает из признака среднее и делит на стандартное отклонение:

$$x'_{ij} = \frac{x_{ij} - m_j}{\sigma_j}, \quad m_j = \frac{1}{\ell}\sum_{i=1}^{\ell} x_{ij}, \quad \sigma_j = \sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}(x_{ij} - m_j)^2}.$$

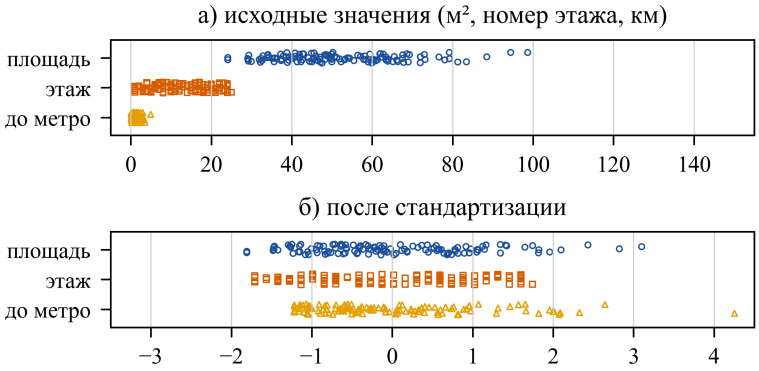


Рис. 2.2. Значения признаков до и после стандартизации

Читаем формулу. Число m_j — среднее значение j-го признака по выборке, σ_j — его типичное отклонение от среднего. Новое значение x'_{ij} говорит, на сколько «типичных отклонений» объект выше или ниже среднего. После стандартизации у каждого признака среднее 0 и стандартное отклонение 1 (рис. 2.2, б), а единицы измерения исчезают. Для площадей из табл. 2.1 $m = 52$ м², $\sigma \approx 13{,}7$ м², и квартира площадью 75 м² получает значение $(75 - 52)/13{,}7 \approx 1{,}68$.

Минимаксная нормализация $x'_{ij} = (x_{ij} - \min_i x_{ij})/(\max_i x_{ij} - \min_i x_{ij})$ переводит значения в отрезок $[0, 1]$. Она проще, но один выброс сжимает все остальные значения к краю отрезка.

2.6. Конструирование признаков

Линейная модель линейна по параметрам, но не обязана быть линейной по исходным данным: признаки $f_j(x)$ могут быть любыми функциями объекта. Так линейной моделью описывают нелинейные зависимости:

- степени признака: $f_1(x) = x$, $f_2(x) = x^2$, … — так строится полиномиальная регрессия;
- логарифм для величин с длинным «хвостом» распределения;
- составляющие даты: день недели, месяц, праздник; для временных рядов — значения той же величины за прошлые периоды;
- агрегаты: средняя цена квартир в районе; в задаче из гл. 1 — число изменений модуля программы за последний месяц.

Особый случай — *взаимодействие* признаков. Цена квартиры примерно пропорциональна площади, но цена квадратного метра в разных районах разная. Модель

$$g(x, \theta) = \theta_0 + \theta_Б f_Б(x) + \theta_В f_В(x) + \theta_1 \cdot \text{площадь}(x)$$

допускает для района только сдвиг: прямые цены от площади в трёх районах параллельны (рис. 2.3, а). Чтобы наклон стал своим в каждом районе, добавляют произведения площади на индикаторы районов, площадь$(x) \cdot f_k(x)$ для $k \in \{А, Б, В\}$ (а общий признак «площадь» убирают — он равен сумме этих произведений). Вес при таком признаке — цена квадратного метра в районе k (рис. 2.3, б). Умножать площадь на номер района, закодированный числом, бессмысленно: получится признак, который растёт с «номером» района, а не отражает цену метра.

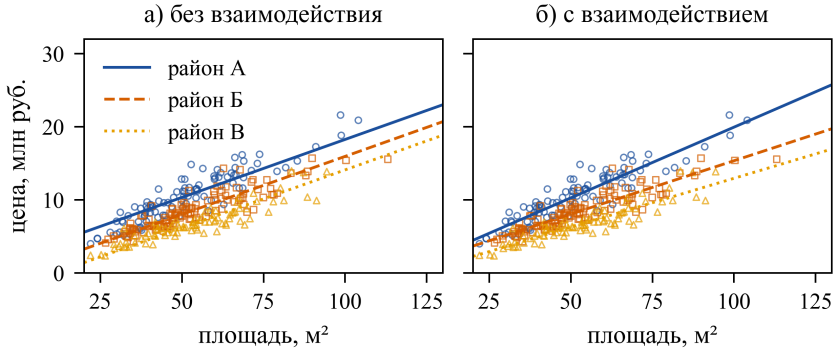


Рис. 2.3. Модель без взаимодействия «площадь × район» и с ним (синтетические данные)

Удачный новый признак нередко улучшает модель сильнее, чем смена метода обучения; придумать его помогает знание предметной области.

2.7. Утечки данных

Чтобы узнать, как модель будет работать на новых объектах, часть выборки откладывают и при обучении не используют. Модель обучают на *обучающей выборке*, а качество оценивают на *тестовой* — объектах, которых модель не видела.

Утечка данных — попадание в обучение информации, которой не будет в момент применения модели. Из-за утечки качество на тестовой выборке оказывается высоким, а на действительно новых данных модель работает заметно хуже. Типичные источники утечки:

- признак вычислен по событиям, которые происходят позже прогноза. В задаче из гл. 1 модель до выхода новой версии программы предсказывает, найдут ли ошибки в модуле. Признак «число исправлений, внесённых в модуль за месяц после выхода версии» в момент прогноза ещё неизвестен и, по сути, содержит сам ответ;
- статистики предобработки посчитаны по всем данным. Если среднее и стандартное отклонение для стандартизации, медиана для пропусков или кодирование средним вычислены до разбиения выборки на обучающую и тестовую, в обучение проникает информация о тестовых объектах;
- один и тот же объект (дубликат) попал в обе части выборки;
- для данных, упорядоченных во времени, модель обучена на более поздних данных, чем тестовые, то есть «знает будущее».

Чтобы статистики предобработки не несли информацию о тестовых объектах, всё, что вычисляется по данным (средние, медианы, список значений категорий, коды), вычисляют только по обучающей выборке. К тестовой выборке и к новым данным применяют уже вычисленные значения и не пересчитывают их. В библиотеке scikit-learn у преобразований два метода: `fit` вычисляет статистики, `transform` применяет их. Конвейер (`Pipeline`) объединяет преобразования и модель так, что `fit` вызывается только на обучающих данных (листинг 2.1) [2, 3]. Тот же принцип — одинаковые преобразования при обучении и применении — описан как шаблон проектирования систем машинного обучения [4, 5].

Листинг 2.1. Конвейер предобработки и линейной регрессии

```
from sklearn.compose import make_column_transformer
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler

num = make_pipeline(SimpleImputer(strategy="median"),
                    StandardScaler())
prep = make_column_transformer(
    (num, ["площадь", "этаж", "год"]),
    (OneHotEncoder(drop="first"), ["район"]))
model = make_pipeline(prep, LinearRegression())
# статистики вычисляются только по обучающей части
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Вопросы для самоконтроля

1. Почему номинальный признак нельзя кодировать числами 1, 2, 3?
2. Зачем при прямом кодировании удаляют один из столбцов?
3. Чем стандартизация отличается от минимаксной нормализации? Когда предпочтительнее каждая?
4. Как с помощью признаков описать нелинейную зависимость линейной моделью?
5. Назовите источники утечки данных. Как конвейер scikit-learn защищает от одного из них?

Задачи

1. Запишите матрицу **F** для табл. 2.1: заполните пропуск медианой, закодируйте район прямым кодированием с базовым районом A, добавьте фиктивный признак.
2. Стандартизуйте площади из табл. 2.1.

3. Линейная регрессия и метод наименьших квадратов

В гл. 1 обучение было сведено к минимизации эмпирического риска. В этой главе задача решается до конца для самого простого случая: модель линейная, потеря квадратичная. Тогда оптимальные параметры находятся точно — как решение системы линейных уравнений. Из этого решения видно и слабое место метода: при почти совпадающих признаках веса модели становятся неустойчивыми.

3.1. Задача регрессии

Пусть функция задана *таблично*: известны её значения y_i в точках x_i, $i = 1, \dots, \ell$, а аналитическое выражение неизвестно. Восстановить функцию по таблице можно с разными целями. *Интерполяция* — найти значения функции между узлами таблицы; искомая функция проходит через все табличные точки точно. *Экстраполяция* — найти значения за пределами отрезка, на котором задана таблица. *Аппроксимация* — подобрать функцию заданного вида, которая проходит близко к табличным точкам, но не обязательно через них. Во всех трёх случаях по конечной таблице восстанавливается аналитическое выражение функции.

Обучающая выборка в задаче регрессии — такая же таблица: объектам x_i сопоставлены ответы y_i. Регрессия — это аппроксимация, потому что ответы содержат шум: цена квартиры зависит не только от её признаков, но и от торга, срочности продажи, ошибок в объявлении. Обычно считают, что

$$y = \varphi(x) + \varepsilon,$$

где $\varphi(x)$ — закономерная часть ответа, а ε — случайная ошибка со средним 0. Функция, которая проходит через все точки выборки, повторяет вместе с закономерностью и шум, и на новых объектах её прогнозы будут ошибаться на величину этого шума. Поэтому от модели требуют не точного совпадения с ответами, а малой средней ошибки. Экстраполяция для регрессии опасна: за пределами диапазона обучающих данных модель ничем не проверена.

Термин «регрессия» ввёл Ф. Гальтон в конце XIX в., изучая связь роста детей и родителей. Метод наименьших квадратов опубликовал А. Лежандр в 1805 г.; К. Гаусс применял его раньше и дал ему вероятностное обоснование.

3.2. Один признак

Начнём со случая, когда у объекта один числовой признак. Для краткости будем обозначать его значение на i-м объекте через x_i (как в табл. 1.3). Модель — прямая

$$g(x, \theta) = \theta_0 + \theta_1 x,$$

где θ_1 — наклон, θ_0 — значение модели при $x = 0$, то есть ордината точки, в которой прямая пересекает ось ординат. Разность $g(x_i, \theta) - y_i$ называют *остатком* на i-м объекте. Эмпирический риск с квадратичной потерей — это средний квадрат остатков:

$$Q(\theta_0, \theta_1) = \frac{1}{\ell} \sum_{i=1}^{\ell} (\theta_0 + \theta_1 x_i - y_i)^2.$$

Найти параметры, при которых Q минимален, — это и есть *метод наименьших квадратов* (МНК).

Квадрат, в отличие от модуля, — гладкая функция: у Q есть производные по обоим параметрам, и точку минимума можно найти, приравняв их к нулю. График Q — «чаша» с единственным минимумом (рис. 3.1, б); линии уровня Q — вложенные эллипсы с общим центром в точке минимума. Недостаток квадратичной потери — чувствительность к выбросам: большой остаток после возведения в квадрат вносит в Q непропорционально большой вклад, и прямая смещается к выбросу.

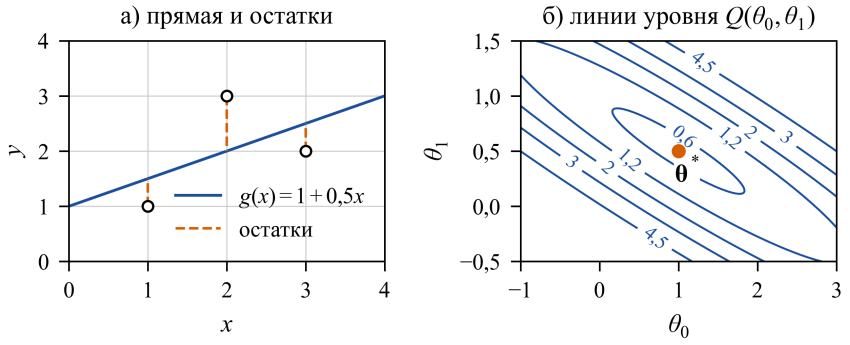


Рис. 3.1. МНК на трёх точках (1; 1), (2; 3), (3; 2)

Вывод формул. В точке минимума обе частные производные Q равны нулю. Производная по θ_0:

$$\frac{\partial Q}{\partial \theta_0} = \frac{2}{\ell}\sum_{i=1}^{\ell}(\theta_0 + \theta_1 x_i - y_i) = 0.$$

Разделим на 2 и раскроем сумму: $\theta_0 + \theta_1\bar{x} - \bar{y} = 0$, где $\bar{x}$ и $\bar{y}$ — средние значения признака и ответа. Отсюда

$$\theta_0 = \bar{y} - \theta_1\bar{x}. \tag{3.1}$$

Производная по θ_1 отличается множителем x_i под знаком суммы:

$$\frac{\partial Q}{\partial \theta_1} = \frac{2}{\ell}\sum_{i=1}^{\ell}(\theta_0 + \theta_1 x_i - y_i)\, x_i = 0.$$

Подставим сюда θ_0 из (3.1): под суммой окажется $\big(\theta_1(x_i - \bar{x}) - (y_i - \bar{y})\big)x_i$. Множитель x_i можно заменить на $x_i - \bar{x}$: добавка $\bar{x}\sum(\ldots)$ равна нулю, потому что сумма отклонений от среднего равна нулю. Получаем

$$\theta_1 = \frac{\sum\limits_{i=1}^{\ell}(x_i - \bar{x})(y_i - \bar{y})}{\sum\limits_{i=1}^{\ell}(x_i - \bar{x})^2}. \tag{3.2}$$

Читаем формулы. Равенство (3.1) означает, что прямая МНК проходит через точку средних $(\bar{x}, \bar{y})$, а сумма остатков равна нулю: модель в среднем не завышает и не занижает ответы. Числитель (3.2) — тот же, что у коэффициента корреляции из гл. 2: он положителен, если x и y чаще одновременно выше или одновременно ниже своих средних. Знаменатель — разброс признака. Если признак почти не меняется, знаменатель мал и наклон определяется ненадёжно; если все x_i одинаковы, наклон не определён вовсе. Можно проверить, что $\theta_1 = r s_y / s_x$, где r — коэффициент корреляции, а s_x и s_y — стандартные отклонения признака и ответа.

Пример. Выборка из трёх точек: $(1;1)$, $(2;3)$, $(3;2)$. Средние $\bar{x} = 2$, $\bar{y} = 2$. Числитель (3.2): $(-1)(-1) + 0 \cdot 1 + 1 \cdot 0 = 1$, знаменатель: $1 + 0 + 1 = 2$, значит $\theta_1 = 0{,}5$ и $\theta_0 = 2 - 0{,}5 \cdot 2 = 1$. Модель $g(x) = 1 + 0{,}5x$ даёт прогнозы $1{,}5;\ 2;\ 2{,}5$ и остатки $0{,}5;\ -1;\ 0{,}5$ (рис. 3.1, а). Их сумма равна нулю, а $Q = (0{,}25 + 1 + 0{,}25)/3 = 0{,}5$. Никакая другая прямая не даёт меньшего среднего квадрата остатков.

3.3. Много признаков

Пусть теперь признаков n. Линейная модель (1.1) из гл. 1 с фиктивным признаком $f_0(x) = 1$ записывается скалярным произведением $g(x, \theta) = \langle \theta, \mathbf{x} \rangle$. Добавим в матрицу объекты–признаки столбец из единиц:

$$\mathbf{F} = \begin{pmatrix} 1 & f_1(x_1) & \dots & f_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & f_1(x_\ell) & \dots & f_n(x_\ell) \end{pmatrix}, \quad \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}.$$

Тогда произведение $\mathbf{F}\theta$ — вектор прогнозов для всех объектов сразу: его i-я компонента равна $\sum_j \theta_j f_j(x_i) = g(x_i, \theta)$. Вектор остатков — $\mathbf{F}\theta - \mathbf{y}$, а эмпирический риск — квадрат его длины, делённый на ℓ:

$$Q(\theta) = \frac{1}{\ell} \sum_{i=1}^{\ell} \big(\langle \theta, \mathbf{x}_i \rangle - y_i \big)^2 = \frac{1}{\ell} \|\mathbf{F}\theta - \mathbf{y}\|_2^2.$$

Градиент. Частная производная по θ_k вычисляется так же, как в одномерном случае:

$$\frac{\partial Q}{\partial \theta_k} = \frac{2}{\ell} \sum_{i=1}^{\ell} \big(\langle \theta, \mathbf{x}_i \rangle - y_i \big) f_k(x_i), \quad k = 0, \dots, n.$$

Сумма по объектам произведений остатка на значение k-го признака — это скалярное произведение k-го столбца матрицы $\mathbf{F}$ на вектор остатков, то есть k-я компонента вектора $\mathbf{F}^{\mathrm{T}}(\mathbf{F}\theta - \mathbf{y})$. Все $n+1$ производных вместе образуют градиент

$$\nabla Q(\theta) = \frac{2}{\ell} \mathbf{F}^{\mathrm{T}}(\mathbf{F}\theta - \mathbf{y}). \tag{3.3}$$

Приравняв градиент к нулю, получаем *нормальную систему уравнений*

$$\mathbf{F}^{\mathrm{T}}\mathbf{F}\theta = \mathbf{F}^{\mathrm{T}}\mathbf{y}. \tag{3.4}$$

Если матрица $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ обратима, решение единственно:

$$\theta^* = (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y}. \tag{3.5}$$

Проверка размерностей. Ошибки в матричных формулах удобно ловить по размерам сомножителей. Матрица $\mathbf{F}$ имеет размер $\ell \times (n+1)$, вектор θ — $(n+1) \times 1$, вектор $\mathbf{y}$ — $\ell \times 1$. Тогда $\mathbf{F}\theta$ — $\ell \times 1$, как и $\mathbf{y}$, их можно вычитать. Матрица $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ квадратная, $(n+1) \times (n+1)$, — по строке

и столбцу на каждый параметр; вектор $\mathbf{F}^{\mathrm{T}}\mathbf{y}$ — $(n+1)\times 1$. Значит, в (3.4) столько же уравнений, сколько неизвестных, и размер системы зависит от числа признаков, а не от числа объектов.

Когда решение единственно. Матрица $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ обратима, если столбцы $\mathbf{F}$ линейно независимы: ни один признак не выражается линейно через остальные. Для этого объектов должно быть не меньше, чем параметров: $\ell \geq n+1$. Типичное нарушение описано в гл. 2: если при прямом кодировании оставить индикаторы всех районов, их сумма совпадёт со столбцом f_0. Тогда решений бесконечно много: увеличим θ_0 на любое число и уменьшим на то же число веса всех районов — прогнозы не изменятся.

Пример. Проверим результат разд. 3.2 по формуле (3.5). Для трёх точек

$$\mathbf{F} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}, \quad \mathbf{F}^{\mathrm{T}}\mathbf{F} = \begin{pmatrix} 3 & 6 \\ 6 & 14 \end{pmatrix}, \quad \mathbf{F}^{\mathrm{T}}\mathbf{y} = \begin{pmatrix} 6 \\ 13 \end{pmatrix}.$$

Определитель $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ равен $3\cdot 14 - 6\cdot 6 = 6$, обратная матрица и решение:

$$(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1} = \frac{1}{6}\begin{pmatrix} 14 & -6 \\ -6 & 3 \end{pmatrix}, \quad \theta^{*} = \frac{1}{6}\begin{pmatrix} 14\cdot 6 - 6\cdot 13 \\ -6\cdot 6 + 3\cdot 13 \end{pmatrix} = \begin{pmatrix} 1 \\ 0{,}5 \end{pmatrix}.$$

Получились те же $\theta_0 = 1$, $\theta_1 = 0{,}5$: формулы разд. 3.2 — частный случай (3.5).

На практике обратную матрицу не вычисляют: это дольше, а при почти линейно зависимых признаках ещё и теряет точность. Систему решают численно устойчивыми методами — через QR- или сингулярное разложение $\mathbf{F}$. В NumPy это функция `np.linalg.lstsq(F, y)`, в scikit-learn — класс `LinearRegression` (листинг 3.1); столбец из единиц он добавляет сам, свободный член хранится в атрибуте `intercept_`, остальные веса — в `coef_`.

Листинг 3.1. Линейная регрессия в NumPy и scikit-learn

```
import numpy as np
from sklearn.linear_model import LinearRegression

x = np.array([[1.0], [2.0], [3.0]])         # один признак
y = np.array([1.0, 3.0, 2.0])
F = np.hstack([np.ones((3, 1)), x])         # столбец f0 = 1
theta = np.linalg.lstsq(F, y, rcond=None)[0]
print(theta)                                # [1.  0.5]

model = LinearRegression().fit(x, y)        # f0 добавляется сам
print(model.intercept_, model.coef_)        # 1.0 [0.5]
```

Вычисление $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ требует порядка ℓn^2 операций, решение системы — порядка n^3. При сотнях тысяч признаков или выборке, которая не помещается в память, точное решение становится слишком дорогим; тогда минимум ищут приближённо, итерационными методами, — им посвящена следующая глава.

3.4. Геометрический смысл

Решение МНК можно увидеть геометрически [6, 7]. Векторы ответов и прогнозов — точки ℓ-мерного пространства, по одной координате на объект. При всевозможных θ векторы прогнозов $\mathbf{F}\theta$ — это линейные комбинации столбцов $\mathbf{F}$; вместе они образуют подпространство

$$\mathcal{F} = \{\mathbf{F}\theta \mid \theta \in \mathbb{R}^{n+1}\}$$

размерности $n+1$ — на рис. 3.2 это плоскость. Каждая точка плоскости $\mathcal{F}$ — вектор прогнозов при каком-то конкретном θ. Вектор ответов $\mathbf{y}$ из-за шума обычно плоскости $\mathcal{F}$ не принадлежит. МНК ищет точку плоскости, ближайшую к $\mathbf{y}$, ведь Q — квадрат расстояния между точками $\mathbf{y}$ и $\mathbf{F}\theta$ (с точностью до множителя $1/\ell$).

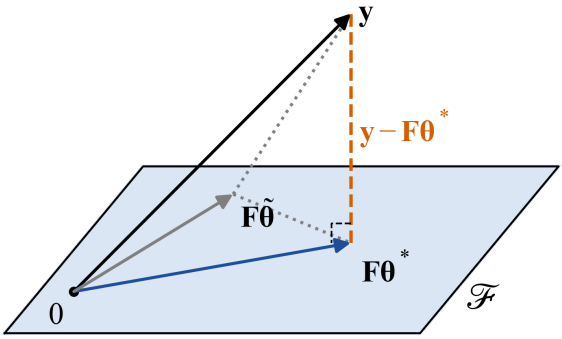


Рис. 3.2. Решение МНК — проекция вектора ответов на множество векторов прогнозов

Ближайшая точка — ортогональная проекция: вектор остатков $\mathbf{y}-\mathbf{F}\theta^*$ перпендикулярен плоскости $\mathcal{F}$, то есть каждому столбцу $\mathbf{F}$. Это и есть нормальная система (3.4), записанная в виде $\mathbf{F}^{\mathrm{T}}(\mathbf{y} - \mathbf{F}\theta^*) = \mathbf{0}$; отсюда и её название («нормаль» — перпендикуляр). Убедимся, что проекция действительно ближе к $\mathbf{y}$, чем любая другая точка плоскости. Возьмём

произвольный вектор параметров $\tilde{\theta} \neq \theta^*$ и соответствующую ему точку $\mathbf{F}\tilde{\theta}$ плоскости $\mathcal{F}$ (рис. 3.2). Точки $\mathbf{y}$, $\mathbf{F}\theta^*$ и $\mathbf{F}\tilde{\theta}$ образуют прямоугольный треугольник с прямым углом при вершине $\mathbf{F}\theta^*$: катет $\mathbf{y} - \mathbf{F}\theta^*$ перпендикулярен плоскости, а катет $\mathbf{F}\theta^* - \mathbf{F}\tilde{\theta}$ лежит в ней. По теореме Пифагора

$$\|\mathbf{y} - \mathbf{F}\tilde{\theta}\|_2^2 = \|\mathbf{y} - \mathbf{F}\theta^*\|_2^2 + \|\mathbf{F}\theta^* - \mathbf{F}\tilde{\theta}\|_2^2 > \|\mathbf{y} - \mathbf{F}\theta^*\|_2^2,$$

то есть гипотенуза длиннее катета: при любых других параметрах сумма квадратов остатков больше.

В примере из разд. 3.2 вектор остатков $(-0{,}5;\ 1;\ -0{,}5)$ перпендикулярен обоим столбцам $\mathbf{F}$: $-0{,}5 + 1 - 0{,}5 = 0$ и $-0{,}5 \cdot 1 + 1 \cdot 2 - 0{,}5 \cdot 3 = 0$. Ортогональность столбцу из единиц и означает, что сумма остатков равна нулю, — это верно для любой линейной регрессии со свободным членом.

Вектор прогнозов равен $\mathbf{F}\theta^* = \mathbf{P}\mathbf{y}$, где $\mathbf{P} = \mathbf{F}(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}$ — *проекционная матрица* размера $\ell \times \ell$. Прогнозы на обучающих объектах — линейная функция ответов: каждый прогноз — взвешенная сумма всех y_i.

3.5. Мультиколлинеарность

Мультиколлинеарностью называют ситуацию, когда признаки почти линейно зависимы: например, площадь квартиры по документам и по результатам обмера, длина модуля программы в строках и в символах. Формально матрица $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ обратима, но решение МНК становится неустойчивым.

Пример. Покажем, к чему приводит почти линейная зависимость признаков. Пусть у трёх объектов два признака почти совпадают (свободного члена нет):

$$\mathbf{F} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3{,}1 \end{pmatrix}, \quad \mathbf{F}^{\mathrm{T}}\mathbf{F} = \begin{pmatrix} 14 & 14{,}3 \\ 14{,}3 & 14{,}61 \end{pmatrix}.$$

Определитель матрицы $\mathbf{F}^{\mathrm{T}}\mathbf{F}$, равный $14 \cdot 14{,}61 - 14{,}3^2 = 0{,}05$, мал по сравнению с её элементами, поэтому элементы обратной матрицы велики, около 290, и любая погрешность в правой части $\mathbf{F}^{\mathrm{T}}\mathbf{y}$ умножается на эти большие числа.

Пусть ответы $\mathbf{y} = (2;\ 4;\ 6{,}1)$, причём последний ответ измерен с погрешностью $\pm 0{,}1$, то есть менее 2 %: с тем же успехом он мог оказаться равным 6 или 6,2. Решим нормальную систему для всех трёх вариантов (табл. 3.1).

Таблица 3.1. Решения МНК при малом изменении одного ответа

y_3	θ_1^*	θ_2^*	Прогнозы на обучающих объектах	Прогноз для объекта (2; 3)
6,0	2	0	2; 4; 6	4
6,1	1	1	2; 4; 6,1	5
6,2	0	2	2; 4; 6,2	6

Таблица показывает три следствия мультиколлинеарности.

1. Веса неустойчивы. Когда y_3 меняется всего на 0,1 (с 6,1 до 6,0 или до 6,2), решение меняется полностью: по первой модели важен только первый признак, по третьей — только второй, по второй — оба поровну. Делать по таким весам выводы о том, какой признак сильнее влияет на ответ, нельзя.
2. На обучающей выборке неустойчивость не видна: все три модели почти одинаково точно воспроизводят ответы, и по ошибке на обучающих объектах их не различить.
3. На новых объектах прогнозы расходятся. Если у нового объекта признаки различаются сильнее, чем в выборке, например (2; 3), три модели предсказывают 4, 5 и 6 — разброс в 50 %, хотя данные различались на 2 %.

Геометрически это видно на рис. 3.3, где показаны линии уровня $Q(\theta_1, \theta_2)$. Вместо «чаши» получился длинный узкий «овраг» вдоль прямой $\theta_1 + \theta_2 \approx 2$. На его дне все точки дают почти одинаковую ошибку, поэтому малое изменение данных сдвигает минимум далеко вдоль оврага.

Сингулярное разложение. Чтобы понять, откуда неустойчивость, воспользуемся *сингулярным разложением* (singular value decomposition, SVD). Любую матрицу $\mathbf{F}$ размера $\ell \times (n+1)$ при $\ell \geq n+1$ можно представить в виде

$$\mathbf{F} = \mathbf{V}\mathbf{D}\mathbf{U}^{\mathrm{T}},$$

где $\mathbf{V}$ — матрица $\ell \times (n+1)$ с ортонормированными столбцами $\mathbf{v}_j$ ($\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$), $\mathbf{U}$ — ортогональная матрица $(n+1)\times(n+1)$ со столбцами $\mathbf{u}_j$ ($\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$), $\mathbf{D} = \operatorname{diag}(s_1, \ldots, s_{n+1})$ — диагональная матрица *сингулярных чисел* $s_1 \geq \ldots \geq s_{n+1} \geq 0$. Подставим разложение в (3.5); так как $\mathbf{F}^{\mathrm{T}}\mathbf{F} = \mathbf{U}\mathbf{D}^2\mathbf{U}^{\mathrm{T}}$, после сокращений остаётся

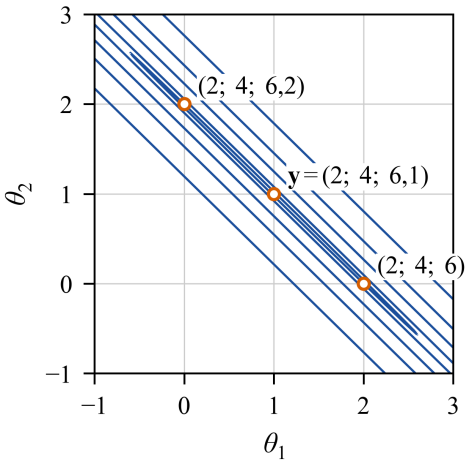


Рис. 3.3. Линии уровня Q при почти совпадающих признаках; точки — решения МНК при трёх векторах ответов

$$\theta^* = \mathbf{U}\mathbf{D}^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{y} = \sum_{j=1}^{n+1} \frac{1}{s_j}\,\mathbf{u}_j\,(\mathbf{v}_j^{\mathrm{T}}\mathbf{y}), \qquad \mathbf{F}\theta^* = \sum_{j=1}^{n+1} \mathbf{v}_j\,(\mathbf{v}_j^{\mathrm{T}}\mathbf{y}).$$

Читаем формулу. Число $\mathbf{v}_j^{\mathrm{T}}\mathbf{y}$ — проекция вектора ответов на направление $\mathbf{v}_j$. В решение θ^* каждая проекция входит, *делённая* на s_j. Если какое-то сингулярное число мало, соответствующее слагаемое огромно, и любая погрешность в $\mathbf{y}$ вдоль $\mathbf{v}_j$ умножается на $1/s_j$. В прогнозы $\mathbf{F}\theta^*$ сингулярные числа не входят вовсе. Поэтому в примере выше прогнозы на обучающих объектах устойчивы, а веса — нет. Малое s_j означает, что столбцы $\mathbf{F}$ почти линейно зависимы: комбинация признаков с коэффициентами из $\mathbf{u}_j$ почти равна нулю на всех объектах. В примере $s_1 \approx 5{,}35$, $s_2 \approx 0{,}042$, а $\mathbf{u}_2 \approx (0{,}71;\ -0{,}70)$ — разность двух признаков.

Насколько система чувствительна к погрешностям, показывает *число обусловленности* матрицы $\mathbf{S} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$ — отношение наибольшего и наименьшего собственных значений, которые равны квадратам сингулярных чисел:

$$\kappa(\mathbf{S}) = \frac{s_1^2}{s_{n+1}^2}.$$

Относительная погрешность решения системы $\mathbf{S}\theta = \mathbf{F}^{\mathrm{T}}\mathbf{y}$ может быть до

$\kappa(\mathbf{S})$ раз больше относительной погрешности правой части. В примере $\kappa \approx 16\,400$; при $\kappa \approx 1$ погрешности не усиливаются, при κ порядка 10^{12} и выше вычисления в типе `float64` (около 16 верных десятичных знаков) теряют большую часть точности.

Псевдообратная матрица $\mathbf{F}^{+} = \mathbf{U}\mathbf{D}^{-1}\mathbf{V}^{\mathrm{T}}$ даёт решение $\theta^{*} = \mathbf{F}^{+}\mathbf{y}$ и тогда, когда признаки строго линейно зависимы и $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ необратима: в $\mathbf{D}^{-1}$ величину $1/s_j$ для нулевых (и очень малых) s_j заменяют нулём. Из бесконечного множества решений получается решение с наименьшей длиной $\|\theta\|_2$. Так работают `np.linalg.pinv` и `np.linalg.lstsq`. В NumPy буквы другие: `np.linalg.svd(F)` возвращает матрицу $\mathbf{V}$ под именем `U`, вектор сингулярных чисел и матрицу $\mathbf{U}^{\mathrm{T}}$.

Что делать с мультиколлинеарностью:

- *отбор признаков* — удалить один из почти дублирующих признаков, исходя из смысла задачи или корреляций между признаками;
- *преобразование признаков* — перейти к меньшему числу новых признаков, например к сумме и разности почти совпадающих (на этом основан метод главных компонент; здесь он не рассматривается);
- *регуляризация* — добавить к Q штраф за большие веса, который превращает овраг в чашу с единственным минимумом (гл. 6).

3.6. Полиномиальная регрессия

Слово «линейная» в названии модели относится к параметрам, а не к исходным данным. Если зависимость нелинейна, в качестве признаков берут степени: $f_j(x) = x^j$, $j = 0, \dots, d$. Модель

$$g(x, \theta) = \sum_{j=0}^{d} \theta_j x^j = \theta_0 + \theta_1 x + \dots + \theta_d x^d$$

— многочлен степени d по x, но линейная функция параметров. Поэтому всё сказанное выше применимо без изменений: строится матрица $\mathbf{F}$ со столбцами $1, x, x^2, \dots, x^d$ и решается нормальная система.

На рис. 3.4 время отклика сервиса растёт с нагрузкой нелинейно. Прямая ($d = 1$) систематически ошибается: занижает время на краях и завышает в середине. Многочлен второй степени описывает зависимость хорошо. Многочлен девятой степени проходит ближе ко всем точкам, но между ними колеблется, повторяя шум. Степень d — гиперпараметр: с её ростом ошибка на обучающей выборке только уменьшается (при $d = \ell - 1$ многочлен проходит через все точки, регрессия превращается в интерполяцию), а ошибка на новых данных, начиная с некоторой сте-

пени, растёт. Как выбрать d по данным, не участвовавшим в обучении, рассказано в гл. 5.

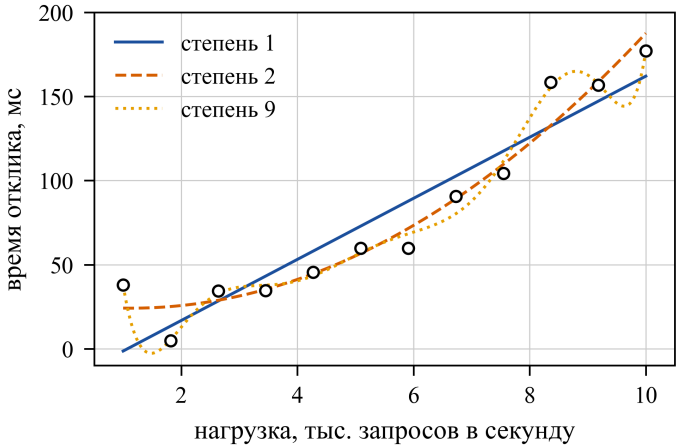


Рис. 3.4. Многочлены степеней 1, 2 и 9, построенные МНК по 12 точкам (синтетические данные)

У полиномиальных признаков есть и вычислительная особенность. Если x принимает значения от 1 до 10, столбцы x^4 и x^5 почти пропорциональны, и матрица $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ плохо обусловлена. Для многочлена пятой степени по 12 точкам на этом отрезке $\kappa(\mathbf{F}^{\mathrm{T}}\mathbf{F}) \approx 2 \cdot 10^{12}$. Если сначала стандартизовать x (гл. 2), число обусловленности падает до $1{,}6 \cdot 10^3$. Поэтому в scikit-learn полиномиальные признаки строят после стандартизации (листинг 3.2).

Листинг 3.2. Полиномиальная регрессия со стандартизацией признака

```
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler

model = make_pipeline(
    StandardScaler(),
    PolynomialFeatures(degree=2, include_bias=False),
    LinearRegression())
```

При нескольких исходных признаках `PolynomialFeatures` добавляет и их попарные произведения — взаимодействия из гл. 2. Число таких признаков быстро растёт со степенью, а с ним и риск мультиколлинеарности и того, что модель, как многочлен девятой степени на рис. 3.4,

начнёт повторять шум выборки.

Вопросы для самоконтроля

1. Чем аппроксимация отличается от интерполяции? Почему регрессия — это аппроксимация?
2. Почему прямая МНК всегда проходит через точку $(\bar{x}, \bar{y})$?
3. Каков размер матрицы $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ и от чего он зависит?
4. В каком случае нормальная система имеет бесконечно много решений? Приведите пример из гл. 2.
5. Объясните геометрически, почему вектор остатков ортогонален столбцам матрицы $\mathbf{F}$.
6. Почему при мультиколлинеарности веса модели неустойчивы, а прогнозы на обучающих объектах — нет?
7. Почему модель полиномиальной регрессии называют линейной?

Задачи

1. По точкам $(0;1)$, $(1;1)$, $(2;4)$ найдите параметры прямой МНК по формулам (3.1) и (3.2) и проверьте, что сумма остатков равна нулю.
2. Для той же выборки запишите $\mathbf{F}$, $\mathbf{F}^{\mathrm{T}}\mathbf{F}$, $\mathbf{F}^{\mathrm{T}}\mathbf{y}$ и решите нормальную систему.
3. Признак f_2 равен f_1, измеренному в других единицах: $f_2(x) = 100 f_1(x)$. Покажите, что матрица $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ для этих двух признаков необратима. Какое решение даст псевдообратная матрица — с весом при одном признаке или при обоих?
4. Через сколько точек с различными x всегда можно точно провести многочлен степени d? Чему тогда равен эмпирический риск на обучающей выборке?

4. Градиентный спуск

В гл. 3 параметры линейной регрессии, минимизирующие эмпирический риск, были получены точно — как решение нормальной системы уравнений. Такое решение в явном виде существует далеко не всегда, и в этой главе описан универсальный способ минимизации функционала качества — градиентный спуск: двигаться к минимуму небольшими шагами, на каждом шаге в направлении, в котором функция убывает быстрее всего.

4.1. Зачем нужны итерации

Формула $\theta^* = (\mathbf{F}^\mathrm{T}\mathbf{F})^{-1}\mathbf{F}^\mathrm{T}\mathbf{y}$ годится не всегда. Во-первых, она дорогая: при n признаках решение нормальной системы требует порядка n^3 операций, а для вычисления $\mathbf{F}^\mathrm{T}\mathbf{F}$ вся выборка должна быть доступна сразу. При сотнях тысяч признаков (например, слов в текстах) или выборке, которая не помещается в память, это непрактично. Во-вторых, для большинства моделей и функций потерь формулы для минимума нет вовсе. Например, для логистической регрессии — одной из основных моделей классификации — уравнение «градиент равен нулю» в явном виде не решается.

Градиентный спуск требует только одного: уметь вычислять градиент ∇Q в любой точке. Поэтому один и тот же алгоритм обучает как линейную и логистическую регрессию, так и нейронные сети.

4.2. Интуиция на одном параметре

Возьмём три точки $(1;1)$, $(2;2)$, $(3;3)$ и модель без свободного члена $g(x, \theta_1) = \theta_1 x$. Эмпирический риск

$$Q(\theta_1) = \frac{1}{3}\sum_{i=1}^{3}(\theta_1 x_i - y_i)^2.$$

У всех трёх точек $y_i = x_i$, поэтому $\theta_1 x_i - y_i = \theta_1 x_i - x_i = x_i(\theta_1 - 1)$, и

$$Q(\theta_1) = \frac{1}{3}\sum_{i=1}^{3} x_i^2(\theta_1 - 1)^2 = \frac{1+4+9}{3}(\theta_1 - 1)^2 = \frac{14}{3}(\theta_1 - 1)^2$$

— парабола с минимумом в точке $\theta_1 = 1$ (рис. 4.1). Представим, что формулы минимума мы не знаем, но в любой точке можем вычислить производную

$$Q'(\theta_1) = \frac{28}{3}(\theta_1 - 1).$$

Знак производной показывает, куда двигаться. Если $Q' > 0$, функция растёт вправо, и минимум левее — θ_1 надо уменьшить. Если $Q' < 0$, минимум правее. Величина $|Q'|$ говорит о крутизне: вдали от минимума склон крутой, вблизи — пологий. Отсюда правило шага:

$$\theta_1 := \theta_1 - \eta\, Q'(\theta_1),$$

где $\eta > 0$ — *скорость обучения* (learning rate). Шаг автоматически направлен к минимуму и уменьшается по мере приближения к нему.

Пример. Начнём с $\theta_1 = 0$, где $Q'(0) = -28/3 \approx -9{,}33$. При $\eta = 0{,}05$ первый шаг даёт $\theta_1 = 0 + 0{,}05 \cdot 9{,}33 = 0{,}467$. Далее $Q'(0{,}467) = -4{,}98$ и $\theta_1 = 0{,}467 + 0{,}05 \cdot 4{,}98 = 0{,}716$; третий шаг даёт 0,848. Расстояние до минимума каждый раз умножается на одно и то же число: $\theta_1 - 1 := (1 - 28\eta/3)(\theta_1 - 1)$, здесь $1 - 28 \cdot 0{,}05/3 \approx 0{,}53$ (рис. 4.1, а).

При $\eta = 0{,}2$ множитель равен $1 - 28 \cdot 0{,}2/3 \approx -0{,}87$: отрицательный, но по модулю меньше единицы. Точка перескакивает через минимум (0; 1,867; 0,249; 1,651; …) и приближается к нему медленно, колеблясь (рис. 4.1, б). При $\eta = 0{,}25$ множитель $-1{,}33$ по модулю больше единицы: каждый шаг уводит дальше от минимума, процесс *расходится* (рис. 4.1, в). Спуск сходится, когда $|1 - \eta Q''| < 1$, то есть при $\eta < 2/Q''$; здесь $Q'' = 28/3$ и граница равна 0,21. Чем круче парабола, тем меньше допустимый шаг.

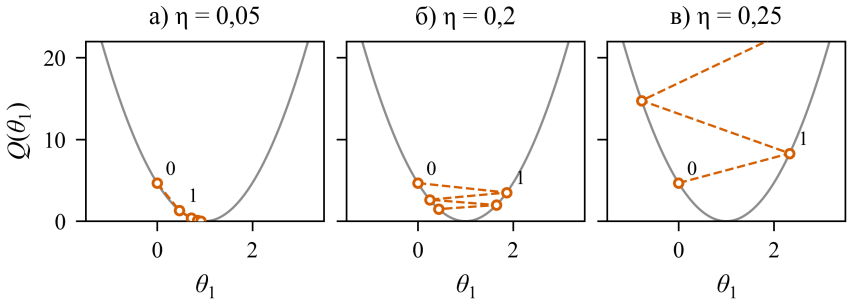


Рис. 4.1. Градиентный спуск по одному параметру: четыре шага из точки $\theta_1 = 0$

4.3. Алгоритм

При нескольких параметрах производная заменяется градиентом — вектором частных производных $\nabla Q(\theta) = (\partial Q/\partial\theta_0, \dots, \partial Q/\partial\theta_n)$. Градиент направлен в сторону наискорейшего роста функции, а его длина — крутизна склона в этом направлении. Значит, антиградиент $-\nabla Q$ указывает направление наискорейшего убывания. *Градиентный спуск* повторяет шаг

$$\theta^{(t+1)} = \theta^{(t)} - \eta\,\nabla Q\big(\theta^{(t)}\big), \tag{4.1}$$

где $t = 0, 1, 2, \dots$ — номер шага (итерации), а $\theta^{(0)}$ — начальное приближение.

Читаем формулу. Для каждого параметра θ_j шаг выполняется по своей частной производной: $\theta_j^{(t+1)} = \theta_j^{(t)} - \eta\,\partial Q/\partial\theta_j$. Все производные вычисляются в *одной и той же* точке $\theta^{(t)}$, и только потом обновляются все параметры. Если обновить θ_0 и сразу использовать новое значение при вычислении производной по θ_1, получится другой алгоритм, не градиентный спуск. В коде это значит: сначала вычислить весь вектор градиента, затем изменить вектор параметров.

Когда остановиться. Обычно используют одно или несколько условий:

- градиент стал мал: $\|\nabla Q\|_2 < \delta$ — точка близка к минимуму, где градиент равен нулю;
- за шаг почти не изменились Q или параметры;
- достигнуто наибольшее допустимое число шагов.

Выбор скорости обучения. Пример разд. 4.2 показывает три режима. Если η слишком мала, спуск сходится, но требует очень многих шагов. Если η велика, параметры колеблются вокруг минимума; если ещё больше — Q растёт с каждым шагом, и вскоре наступает переполнение. Практический способ — попробовать несколько значений через порядок (0,001; 0,01; 0,1; 1) и посмотреть на *кривую обучения* — график Q от номера шага. Кривая должна монотонно убывать; рост Q — верный признак слишком большой η.

В некоторых книгах эмпирический риск записывают с множителем $1/(2\ell)$ вместо $1/\ell$: двойка от производной квадрата сокращается. Точка минимума от этого не меняется, но градиент уменьшается вдвое, поэтому одно и то же значение η даёт разные шаги. Сравнивая скорости обучения из разных источников, нужно учитывать, как там записан Q.

Локальные минимумы. Эмпирический риск линейной регрессии — «чаша» с единственным минимумом, и спуск при подходящем η при-

ходит в него из любой начальной точки. У других моделей функция Q может иметь несколько локальных минимумов; тогда результат зависит от начального приближения, и спуск запускают несколько раз из разных случайных точек (*мультистарт*).

4.4. Градиентный спуск для линейной регрессии

Для линейной модели с квадратичной потерей градиент найден в гл. 3, формула (3.3). Подставим его в (4.1):

$$\theta^{(t+1)} = \theta^{(t)} - \frac{2\eta}{\ell}\,\mathbf{F}^{\mathrm{T}}\bigl(\mathbf{F}\theta^{(t)} - \mathbf{y}\bigr). \tag{4.2}$$

Покомпонентно:

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \frac{2\eta}{\ell}\sum_{i=1}^{\ell}\bigl(g(x_i,\theta^{(t)}) - y_i\bigr)f_j(x_i), \quad j = 0,\dots,n.$$

Читаем формулу. Шаг для веса θ_j — сумма по объектам произведений остатка на значение j-го признака. Если модель на объекте завышает ответ (остаток положителен), а признак положителен, вес уменьшается, и прогноз на этом объекте снижается. Объекты с большими остатками влияют на шаг сильнее. Когда остатки «уравновешены» — ортогональны всем столбцам $\mathbf{F}$, — шаг равен нулю: это и есть нормальная система.

Одна итерация требует двух умножений матрицы на вектор, порядка ℓn операций, — гораздо меньше, чем n^3 для точного решения при большом n. В NumPy формула (4.2) записывается одной строкой (листинг 4.1). Цикл по объектам и признакам на Python работал бы в сотни раз медленнее: операции над массивами NumPy выполняются скомпилированными библиотеками линейной алгебры (BLAS, LAPACK). Такую запись называют *векторизованной*.

Листинг 4.1. Градиентный спуск для линейной регрессии

```
import numpy as np

def gradient_descent(F, y, eta=0.1, steps=1000, tol=1e-6):
    ell, p = F.shape                    # F — со столбцом f0 = 1
    theta = np.zeros(p)
    for t in range(steps):
        grad = 2 / ell * F.T @ (F @ theta - y)
        theta = theta - eta * grad      # все параметры сразу
        if np.linalg.norm(grad) < tol:
            break
    return theta
```

Пример. Для трёх точек (1; 1), (2; 3), (3; 2) из примера гл. 3 минимум Q находится в точке $\theta^* = (1;\ 0{,}5)$. На рис. 4.2, а показаны 25 шагов спуска с $\eta = 0{,}15$. Линии уровня — вытянутые эллипсы. Антиградиент перпендикулярен линии уровня и направлен не в центр, а поперёк эллипса; траектория сначала колеблется поперёк «оврага», а затем медленно ползёт вдоль его дна. После 25 шагов до минимума остаётся около 40 % начального расстояния.

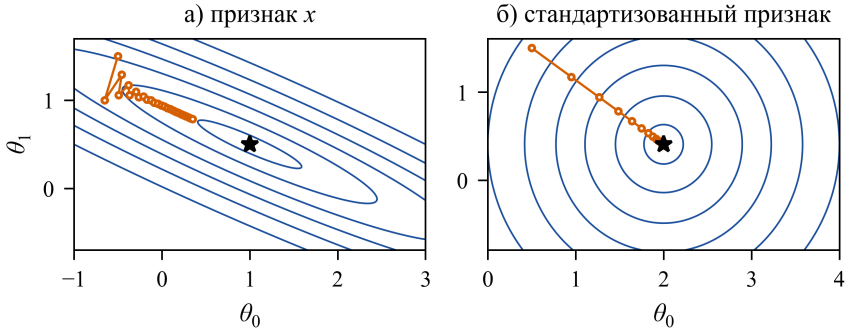


Рис. 4.2. Траектория градиентного спуска (25 шагов, $\eta = 0{,}15$): а — исходный признак; б — стандартизованный

Причина — разная крутизна Q по разным направлениям. Вдоль крутого направления допустимый шаг мал (иначе — расходимость, как в разд. 4.2), а вдоль пологого такой шаг слишком мал для быстрого продвижения. Отношение крутизны по самому крутому и самому пологому направлениям — число обусловленности $\kappa(\mathbf{F}^{\mathrm{T}}\mathbf{F})$ из гл. 3; здесь оно около 46. Чем оно больше, тем больше шагов нужно спуску, причём число шагов растёт примерно пропорционально κ.

4.5. Стохастический и мини-пакетный спуск

Градиент в (4.2) — сумма по всем ℓ объектам. При миллионах объектов каждый шаг требует прохода по всей выборке, а шагов нужны сотни. *Стохастический градиентный спуск* (stochastic gradient descent, SGD) делает шаг по градиенту потери на одном случайно выбранном объекте:

$$\theta := \theta - \eta\,\nabla\mathcal{L}_i(\theta), \qquad \nabla\mathcal{L}_i(\theta) = 2\big(\langle\theta, \mathbf{x}_i\rangle - y_i\big)\mathbf{x}_i,$$

где $\mathcal{L}_i(\theta) = (\langle\theta, \mathbf{x}_i\rangle - y_i)^2$ — квадратичная потеря на объекте x_i.

Читаем формулу. Метод рассматривает один объект и сдвигает веса

так, чтобы ошибка на нём уменьшилась: шаг пропорционален остатку и направлен вдоль признакового описания объекта. Если объект выбран случайно и равновероятно, градиент потери на нём в среднем равен градиенту Q: Q — среднее потерь, а производная среднего — среднее производных. Отдельный шаг может и увеличить Q, но в среднем шаги ведут к минимуму, а каждый стоит в ℓ раз дешевле.

На практике объект не выбирают заново случайно на каждом шаге: выборку перемешивают и проходят по порядку. Один проход по всей выборке называют *эпохой*. Компромисс между полным и стохастическим градиентом — *мини-пакетный* спуск: шаг по среднему градиенту на пакете (mini-batch; в литературе часто встречается калька «мини-батч») из B объектов, обычно от десятков до сотен. Усреднение по пакету уменьшает шум, а вычисления над пакетом векторизуются так же, как в листинге 4.1.

На рис. 4.3 сравниваются три способа на синтетической задаче с $\ell = 1000$ объектов и пятью признаками. За эпоху полный градиент делает один шаг, мини-пакетный — 50, стохастический — 1000. Поэтому за первую эпоху стохастический и мини-пакетный спуск почти доходят до минимума, а полному градиенту нужно несколько эпох. Зато кривые стохастических методов «шумят»: при постоянной η параметры не останавливаются в минимуме, а блуждают вокруг него, потому что градиент на одном объекте не обращается в нуль даже в точке минимума Q.

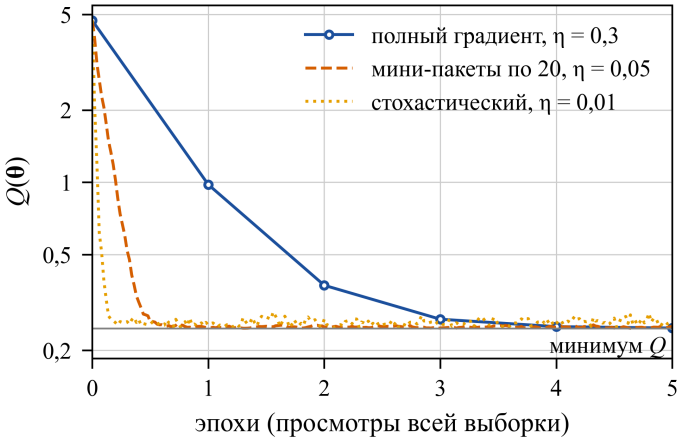


Рис. 4.3. Эмпирический риск в зависимости от числа эпох для трёх вариантов спуска (синтетические данные)

Убывающая скорость обучения. Чтобы блуждание затухало, η уменьшают с номером шага t. Стохастический спуск для выпуклой функции сходится к минимуму, если выполнены *условия Роббинса — Монро*:

$$\eta_t \to 0, \qquad \sum_{t=1}^{\infty} \eta_t = \infty, \qquad \sum_{t=1}^{\infty} \eta_t^2 < \infty.$$

Первое и третье условия гасят шум: шаги становятся всё мельче. Второе не даёт шагам уменьшиться слишком быстро: их сумма бесконечна, поэтому из любой начальной точки можно дойти до минимума. Подходит, например, $\eta_t = \eta_0/t$: гармонический ряд $\sum 1/t$ расходится, а ряд $\sum 1/t^2$ сходится.

Оценка Q на лету. Вычислять Q по всей выборке после каждого шага SGD дорого — это лишает метод смысла. Поэтому Q оценивают по потерям на тех объектах, по которым уже сделаны шаги, *экспоненциальным скользящим средним*:

$$\bar{Q} := \beta \mathcal{L}_i + (1 - \beta)\, \bar{Q},$$

где $\mathcal{L}_i$ — потеря на текущем объекте, вычисленная до шага, а $\beta \in (0, 1)$ — скорость забывания. Новое значение входит в оценку с весом β, предыдущее — с весом $\beta(1-\beta)$, ещё более раннее — $\beta(1-\beta)^2$ и т. д. При $\beta = 0{,}01$ оценка примерно отражает среднюю потерю на последней сотне объектов. По $\bar{Q}$ строят кривую обучения и проверяют условие остановки.

Стохастический спуск позволяет обучать модель на данных, которые поступают потоком, и доучивать её на новых объектах без повторного обучения на всей выборке. Платой служит набор настроек — скорость обучения и закон её убывания, размер пакета, число эпох, — которые подбирают экспериментально.

4.6. Практические приёмы

Масштабирование признаков. Пример на рис. 4.2 показывает, почему перед градиентным спуском признаки стандартизуют (гл. 2). Если признаки измерены в разных единицах (площадь в квадратных метрах, этаж, доля), Q по одним весам гораздо круче, чем по другим, линии уровня вытянуты, а число обусловленности велико. После стандартизации признака из примера столбцы $\mathbf{F}$ становятся ортогональными, линии уровня — окружностями (рис. 4.2, б), $\varkappa = 1$, и антиградиент направлен прямо в минимум: за 25 шагов с тем же $\eta = 0{,}15$ спуск приходит в ми-

нимум с точностью до 10^{-4}. На практике столбцы не бывают в точности ортогональными, но стандартизация обычно уменьшает κ на порядки. Решение, найденное по стандартизованным признакам, затем пересчитывают в исходные единицы или, что проще, применяют модель вместе с тем же преобразованием признаков (конвейер scikit-learn).

Начальное приближение. Для линейной регрессии подходит $\theta^{(0)} = \mathbf{0}$: функция выпукла, и от начальной точки зависит лишь число шагов. Можно взять и небольшие случайные значения или решение, полученное на небольшой подвыборке. Для невыпуклых функций потерь, например в нейронных сетях, начальное приближение влияет на результат, и там его выбирают случайно по особым правилам.

scikit-learn. Стохастический спуск для линейной регрессии реализован в классе `SGDRegressor` (листинг 4.2). По умолчанию он добавляет к Q небольшой штраф за большие веса (регуляризацию) и уменьшает скорость обучения с номером шага; параметр `penalty=None` отключает штраф. Масштабирование признаков выполняется в том же конвейере.

Листинг 4.2. Стохастический градиентный спуск в scikit-learn

```
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

model = make_pipeline(
    StandardScaler(),
    SGDRegressor(penalty=None, eta0=0.01, max_iter=1000,
                 tol=1e-4, random_state=0))
model.fit(X_train, y_train)
```

Для небольших задач линейной регрессии, где признаков не больше нескольких тысяч, быстрее и точнее решить нормальную систему (`LinearRegression`, гл. 3). Градиентный спуск нужен, когда данных много, когда они поступают потоком, или когда формулы для минимума нет.

Вопросы для самоконтроля

1. В каких случаях градиентный спуск предпочтительнее решения нормальной системы?
2. Почему шаг градиентного спуска делают против градиента, а не вдоль него?
3. Что произойдёт при слишком малой и при слишком большой скорости обучения? Как это увидеть на кривой обучения?
4. Почему все параметры нужно обновлять одновременно?
5. Почему градиент потери на одном случайном объекте в среднем совпадает с градиентом эмпирического риска?

6. Зачем при стохастическом спуске уменьшают скорость обучения? Какие условия накладывают на последовательность η_t?
7. Как масштаб признаков влияет на скорость сходимости?

Задачи

1. Для модели $g(x) = \theta_1 x$ и точек $(1;1)$, $(2;2)$, $(3;3)$ сделайте два шага градиентного спуска из $\theta_1 = 2$ при $\eta = 0{,}1$.
2. При каких η градиентный спуск для функции $Q(\theta) = a\theta^2$, $a > 0$, сходится? При каком η он приходит в минимум за один шаг?
3. Для точек $(1;1)$, $(2;3)$, $(3;2)$ и модели $\theta_0 + \theta_1 x$ сделайте один шаг градиентного спуска из $\boldsymbol{\theta} = (0;\,0)$ при $\eta = 0{,}1$.
4. Сделайте один шаг стохастического спуска для той же модели из $\boldsymbol{\theta} = (1;\,0{,}5)$ по объекту $(2;3)$ при $\eta = 0{,}1$. Почему параметры изменились, хотя $(1;\,0{,}5)$ — минимум Q?

5. Переобучение и валидация

Метод наименьших квадратов находит параметры, при которых ошибка на обучающей выборке минимальна. Но модель строят ради прогнозов на новых объектах, и важна ошибка именно на них. Малая ошибка на обучающей выборке ещё не гарантирует малой ошибки на новых объектах. В этой главе показано, когда ошибка на обучающей выборке и ошибка на новых объектах расходятся, как оценить вторую из них и как по этой оценке выбрать модель.

5.1. Недообучение и переобучение

Рассмотрим пример К. Рунге. Функция $f(x) = 1/(1 + 25x^2)$ задана таблично в $\ell = 15$ равноотстоящих точках отрезка $[-2; 2]$; это обучающая выборка. Аппроксимируем функцию многочленами степени d от 0 до 14, находя коэффициенты методом наименьших квадратов. Чтобы проверить качество каждого многочлена, нужны объекты, которых метод обучения не видел. Поэтому зададим ещё *контрольную выборку* — значения функции в 14 точках, лежащих посередине между обучающими. При поиске коэффициентов многочлена контрольные точки не используются: по ним только вычисляется ошибка уже обученной модели — так же, как модель будет ошибаться на новых объектах при применении. Шума в данных нет: ответы — точные значения функции. Пример показывает, как ведут себя ошибки на обучающей и контрольной выборках при усложнении модели.

Многочлен второй степени (рис. 5.1, а) слишком прост: он не может повторить острый пик функции и ошибается почти везде. Ошибка велика и на обучающей, и на контрольной выборке ($Q \approx 0{,}045$ и $0{,}031$). Это *недообучение*: модель не улавливает закономерность.

С ростом степени обе ошибки сначала уменьшаются (рис. 5.1, б). Многочлен восьмой степени описывает функцию заметно лучше, ошибка на контроле минимальна, $Q \approx 0{,}009$. Дальше ошибка на обучающей выборке продолжает падать, а на контрольной — растёт. При $d = 14$ многочлен проходит через все 15 точек и ошибка на обучающей выборке равна нулю, но между точками, особенно у краёв отрезка, многочлен колеблется с огромной амплитудой, и ошибка на контроле достигает 61. Это *переобучение*: модель подогнана под обучающие объекты настолько точно, что на остальных объектах ошибается сильнее, чем более простая модель.

На графике заметны ещё две особенности. Ошибки не меняются при переходе от чётной степени к следующей нечётной: функция f чётная, и коэффициенты при нечётных степенях получаются нулевыми. При ма-

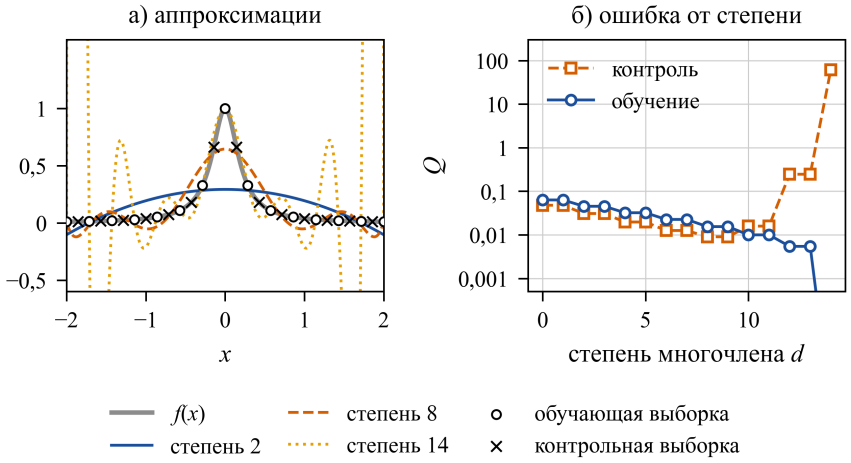


Рис. 5.1. Пример Рунге: а — многочлены трёх степеней; б — среднеквадратичная ошибка на обучающей и контрольной выборках

лых степенях ошибка на контроле чуть меньше, чем на обучении: наибольшая ошибка простых моделей — на вершине пика, в точке $x = 0$, а она входит в обучающую выборку.

Из примера следует, что ошибка на обучающей выборке с усложнением модели только уменьшается и поэтому не годится для выбора модели. Лучшую модель указывает ошибка на объектах, не участвовавших в обучении. Переобучение возникло здесь даже без шума — из-за лишней гибкости модели; при шумных ответах оно наступает ещё раньше, потому что гибкая модель повторяет и шум.

5.2. Обобщающая способность

Способность модели давать хорошие прогнозы на новых объектах называют *обобщающей способностью*. Её мера — *риск*, средняя потеря на всех возможных объектах (гл. 1). Риск вычислить нельзя, и ошибка на обучающей выборке — плохая замена ему. Она похожа на оценку за экзамен, вопросы которого студент знал заранее: метод обучения видел ответы на всех обучающих объектах и подогнал под них параметры, поэтому высокая точность на этих объектах ещё не говорит о качестве на других. Чем больше у модели параметров по сравнению с числом объектов, тем свободнее такая подгонка и тем сильнее ошибка на обучающей выборке расходится с риском. Для многочлена степени $d = \ell - 1$ в примере

Рунге ошибка на обучающей выборке равна нулю, а на контрольной она огромна.

Переобучению способствуют:

- малое число объектов по сравнению с числом параметров;
- лишние или почти линейно зависимые признаки — как показано в гл. 3, при мультиколлинеарности веса подстраиваются под мелкие погрешности данных;
- шум в ответах, который гибкая модель принимает за закономерность.

Избавиться от переобучения полностью нельзя, но его можно уменьшить: собрать больше данных, выбрать модель попроще, ограничить величину параметров. Во всех случаях нужна надёжная оценка ошибки на новых данных — без неё нельзя понять, помогли ли эти меры. Схемы такой оценки описаны в разд. 5.4, а ограничение величины параметров (регуляризация) — в гл. 6.

5.3. Смещение и разброс

Откуда берётся ошибка модели на новом объекте? Для квадратичной потери её можно разложить на три слагаемых. Пусть ответ $y = \varphi(x) + \varepsilon$, где шум ε имеет среднее 0 и дисперсию σ^2. Обучающая выборка случайна: если собрать её заново, ответы будут другими из-за шума, и метод обучения выдаст другую модель $g(x)$. Обозначим через $\mathbb{E}$ математическое ожидание (среднее) по всем возможным обучающим выборкам и по шуму в ответе на новом объекте, а через $\bar{g}(x) = \mathbb{E}\big(g(x)\big)$ — *среднюю модель*, прогноз в точке x, усреднённый по всем выборкам. Тогда для квадратичной ошибки в точке x

$$\mathbb{E}\Big(\big(y - g(x)\big)^2\Big) = \underbrace{\big(\varphi(x) - \bar{g}(x)\big)^2}_{\text{смещение}^2} + \underbrace{\mathbb{E}\Big(\big(g(x) - \bar{g}(x)\big)^2\Big)}_{\text{разброс}} + \underbrace{\sigma^2}_{\text{шум}}. \qquad (5.1)$$

Вывод формулы. Запишем ошибку как сумму трёх частей: $y - g = \varepsilon + (\varphi - \bar{g}) + (\bar{g} - g)$. При возведении в квадрат появляются квадраты частей и попарные произведения. Произведения обращаются в нуль при усреднении: шум ε на новом объекте не зависит от обучающей выборки и имеет среднее 0, а отклонение $\bar{g} - g$ имеет среднее 0 по определению средней модели. Остаются три квадрата.

Читаем формулу. *Смещение* — насколько средняя модель отличается от истинной зависимости. Оно велико, когда модель слишком проста: прямая не повторит синусоиду, на скольких выборках её ни обучай. *Разброс* (дисперсия) — насколько модели, обученные на разных выборках,

отличаются от средней модели. Он велик, когда модель гибкая и подстраивается под случайные особенности конкретной выборки. *Шум* — ошибка, которую не устранит никакая модель: часть ответа не определяется признаками. Недообучение — это большое смещение, переобучение — большой разброс.

Пример. Чтобы увидеть смещение и разброс, обучим модели на многих выборках. Пусть $\varphi(x) = \sin 2\pi x$, выборка — 15 равноотстоящих точек отрезка $[0; 1]$ с шумом $\sigma = 0{,}3$. Сгенерируем 4000 выборок, на каждой построим многочлен степени d методом наименьших квадратов и вычислим слагаемые (5.1), усреднив их по отрезку (табл. 5.1). На рис. 5.2 показаны 20 из этих моделей.

Таблица 5.1. Смещение и разброс многочленов разных степеней ($\sigma^2 = 0{,}09$)

Степень d	Смещение^2	Разброс	Шум	Ожидаемая ошибка
1	0,210	0,011	0,09	0,311
3	0,006	0,021	0,09	0,116
11	0,000	0,154	0,09	0,244

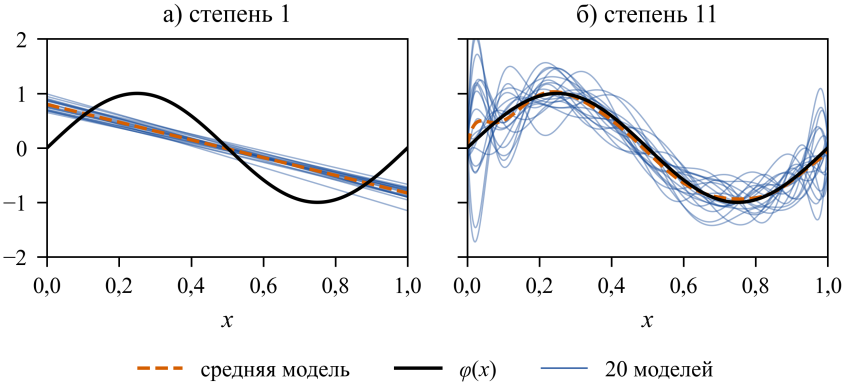


Рис. 5.2. Многочлены, обученные на 20 выборках с разным шумом: а — большое смещение; б — большой разброс

Прямые ($d = 1$) почти совпадают друг с другом — разброс мал, — но все одинаково далеки от синусоиды: смещение велико. Многочлены степени 11 в среднем повторяют синусоиду — смещение почти нулевое, — но каждый по-своему колеблется вслед за шумом своей выборки: разброс велик. Многочлен третьей степени — компромисс, его ожидаемая

ошибка наименьшая. С ростом сложности модели смещение убывает, а разброс растёт, поэтому ошибка на новых данных как функция сложности сначала падает, а потом растёт — как на рис. 5.1, б.

На практике слагаемые (5.1) не вычисляют: истинная зависимость неизвестна, а выборка одна. Разложение объясняет, как действуют меры против переобучения: больше данных — меньше разброс при том же смещении; более простая модель или ограничение параметров — меньше разброс ценой большего смещения.

5.4. Схемы валидации

Оценивать модель на тех же объектах, на которых она обучена, нельзя. Поэтому часть объектов откладывают и в обучении не используют. Все схемы ниже — способы такого разбиения.

Отложенная выборка (hold-out). Выборку делят на обучающую X^ℓ и контрольную (тестовую) X^k, обычно в отношении от 70 : 30 до 80 : 20. Модель обучают на X^ℓ, а оценкой риска служит эмпирический риск на X^k:

$$\mathrm{HO}(\mathcal{M}, X^\ell, X^k) = Q\big(\mathcal{M}(X^\ell), X^k\big).$$

Схема проста, но оценка зависит от того, какие объекты случайно попали в контроль, а часть данных не участвует в обучении.

Кросс-валидация (скользящий контроль, cross-validation). Выборку делят на k примерно равных блоков $X_1, \ldots, X_k$ (здесь k — число блоков, а не размер тестовой выборки X^k). Каждый блок по очереди служит контрольным, а модель обучается на остальных $k-1$ блоках (рис. 5.3, а). Оценка — средняя ошибка по всем блокам:

$$\mathrm{CV}(\mathcal{M}, X^\ell) = \frac{1}{k}\sum_{s=1}^{k} Q\big(\mathcal{M}(X^\ell \setminus X_s), X_s\big),$$

где $X^\ell \setminus X_s$ — выборка без блока X_s. Обычно берут $k = 5$ или 10; отсюда название схемы — k-кратная кросс-валидация (k-fold). Каждый объект ровно один раз оказывается в контроле, поэтому оценка устойчивее, чем у отложенной выборки; плата — k обучений вместо одного. Кроме среднего, полезно смотреть на разброс ошибок по блокам: если две модели различаются меньше, чем меняется ошибка от блока к блоку, считать одну из них лучшей нельзя.

Крайний случай кросс-валидации — *контроль по отдельным объектам* (leave-one-out, LOO): $k = \ell$, каждый блок состоит из одного объекта.

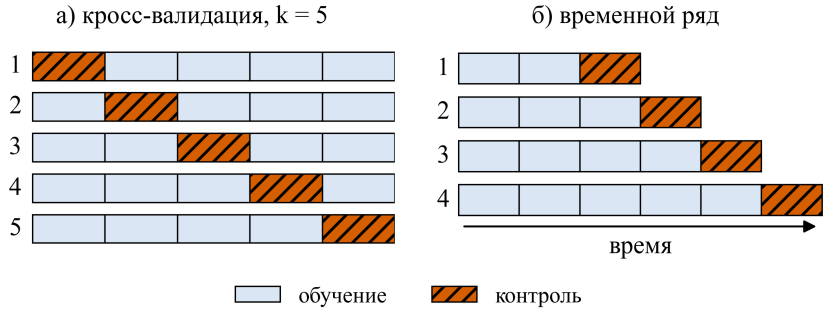


Рис. 5.3. Схемы разбиения выборки: а — кросс-валидация с пятью блоками; б — валидация для временного ряда

В среднем такая оценка почти совпадает с ошибкой модели, обученной на всей выборке, но требует ℓ обучений и применяется на небольших выборках.

Стратификация. В задачах классификации блоки формируют так, чтобы доля каждого класса в них была такой же, как во всей выборке. Иначе при редком классе в какой-то блок могут не попасть его объекты, и ошибка на этом блоке ничего не скажет о качестве распознавания редкого класса.

Временные ряды. Если данные упорядочены во времени (продажи, нагрузка на сервис, история изменений кода), случайное разбиение даёт утечку: модель обучается на более поздних данных и проверяется на более ранних, то есть «знает будущее». Поэтому контрольный блок всегда берут позже обучающих (рис. 5.3, б): модель обучают на данных до некоторого момента и проверяют на следующем периоде, затем сдвигают границу. Так воспроизводится реальное применение: прогноз строится только по прошлому.

Обучающая, валидационная и тестовая выборки. Если по ошибке на контроле выбирают модель или гиперпараметры (например, степень многочлена), эта ошибка тоже становится заниженной: из многих вариантов выбран тот, которому повезло на контрольных объектах. Поэтому выборку делят на три части. На *обучающей* находят параметры. На *валидационной* (или кросс-валидацией внутри обучающей части) сравнивают модели и подбирают гиперпараметры. *Тестовую* откладывают до конца и используют один раз — для итоговой оценки выбранной модели. Если после взгляда на тестовую ошибку что-то меняют в модели, тестовая выборка превращается в валидационную, и ошибка на ней то-

же становится заниженной.

5.5. Метрики качества регрессии

Эмпирический риск с квадратичной потерей — лишь одна из мер качества. Для регрессии используют несколько *метрик*; все они вычисляются по ответам y_i и прогнозам $\hat{y}_i = g(x_i, \theta)$ на объектах выборки, по которой оценивают модель (обычно контрольной); число этих объектов здесь обозначено ℓ.

Среднеквадратичная ошибка (mean squared error) и корень из неё:

$$\text{MSE} = \frac{1}{\ell}\sum_{i=1}^{\ell}(\hat{y}_i - y_i)^2, \qquad \text{RMSE} = \sqrt{\text{MSE}}.$$

MSE совпадает с эмпирическим риском МНК, дифференцируема и удобна для оптимизации, но сильно штрафует большие ошибки и чувствительна к выбросам. RMSE измеряется в тех же единицах, что и ответ (миллионы рублей, миллисекунды), поэтому её легче интерпретировать.

Средняя абсолютная ошибка (mean absolute error) и *средняя абсолютная процентная ошибка* (mean absolute percentage error):

$$\text{MAE} = \frac{1}{\ell}\sum_{i=1}^{\ell}|\hat{y}_i - y_i|, \qquad \text{MAPE} = \frac{100\,\%}{\ell}\sum_{i=1}^{\ell}\left|\frac{\hat{y}_i - y_i}{y_i}\right|.$$

MAE штрафует все ошибки пропорционально их величине и устойчивее к выбросам. MAPE выражает ошибку в процентах и не зависит от единиц измерения: ошибка 1 млн руб. для квартиры за 5 млн серьёзнее, чем для квартиры за 50 млн. Но при ответах, близких к нулю, MAPE теряет смысл — знаменатель почти нулевой.

Коэффициент детерминации сравнивает модель с простейшей — константой, равной среднему ответу $\bar{y}$:

$$R^2 = 1 - \frac{\sum\limits_{i=1}^{\ell}(\hat{y}_i - y_i)^2}{\sum\limits_{i=1}^{\ell}(y_i - \bar{y})^2}.$$

Читаем формулу. Числитель дроби — сумма квадратов ошибок модели, знаменатель — сумма квадратов ошибок константной модели $\hat{y} = \bar{y}$. Если модель безошибочна, $R^2 = 1$; если она не лучше среднего, $R^2 = 0$.

На обучающей выборке для линейной регрессии со свободным членом $0 \le R^2 \le 1$: константа — частный случай линейной модели, и МНК не может найти решение хуже неё. На контрольной выборке R^2 бывает отрицательным: модель может ошибаться сильнее, чем среднее контрольных ответов.

Пример. Покажем, что R^2 на контроле может быть отрицательным. В гл. 3 по точкам $(1;1)$, $(2;3)$, $(3;2)$ построена модель $g(x) = 1 + 0{,}5x$. На этой обучающей выборке $\bar{y} = 2$, сумма квадратов ошибок модели равна $0{,}25 + 1 + 0{,}25 = 1{,}5$, константы — $1 + 1 + 0 = 2$, и $R^2 = 1 - 1{,}5/2 = 0{,}25$. Проверим модель на четырёх новых объектах $(0;2)$, $(1;2{,}5)$, $(2;1{,}5)$, $(4;2)$. Прогнозы — 1; 1,5; 2; 3, ошибки — -1; -1; 0,5; 1, сумма квадратов — 3,25. Среднее контрольных ответов равно 2, сумма квадратов отклонений от него — $0 + 0{,}25 + 0{,}25 + 0 = 0{,}5$. Получаем $R^2 = 1 - 3{,}25/0{,}5 = -5{,}5$: на новых объектах ответы почти не меняются, и константа 2 предсказала бы их гораздо точнее, чем модель с наклоном.

Отсюда практическое правило: прежде чем строить сложную модель, вычисляют качество *базовой модели* (baseline) — прогноза средним или медианой ответа. Модель, которая на контрольной выборке не лучше базовой, бесполезна, как бы хорошо она ни выглядела на обучающей.

Информационные критерии. Иногда модели сравнивают без отложенной выборки — по ошибке на обучающей выборке со штрафом за число параметров. Для линейной регрессии с нормально распределённым шумом *информационные критерии* Акаике (AIC) и Шварца (BIC) с точностью до слагаемого, одинакового для всех моделей, записываются так:

$$\mathrm{AIC} = \ell \ln \mathrm{MSE} + 2(n+1), \qquad \mathrm{BIC} = \ell \ln \mathrm{MSE} + (n+1)\ln \ell,$$

где MSE вычислена на обучающей выборке, а $n+1$ — число параметров модели. Меньшее значение критерия лучше. Первое слагаемое убывает с усложнением модели, второе — растёт. BIC при $\ell > 7$ (когда $\ln \ell > 2$) штрафует сложность сильнее и выбирает более простые модели.

5.6. Подбор гиперпараметров

Степень многочлена, набор признаков, скорость обучения градиентного спуска — гиперпараметры: их не находит метод обучения, а задаёт человек. Подбирают их перебором: для каждого набора значений обучают модель и оценивают её кросс-валидацией на обучающей части выборки, а затем выбирают набор с наименьшей ошибкой.

Поиск по сетке (grid search) перебирает все сочетания значений из заданных списков: например, степени 1, ..., 10 и три способа масштабирования — 30 вариантов. Число вариантов растёт как произведение длин списков, и при нескольких гиперпараметрах перебор становится долгим. *Случайный поиск* (random search) проверяет заданное число случайных сочетаний. Обычно лишь немногие гиперпараметры сильно влияют на качество; при том же числе проверок случайный поиск пробует больше разных значений каждого из них и поэтому часто находит хорошее сочетание быстрее сетки [8]. Непрерывные гиперпараметры, такие как скорость обучения, перебирают в логарифмической шкале: 0,001; 0,01; 0,1.

В scikit-learn перебор с кросс-валидацией выполняет класс `GridSearchCV` (листинг 5.1); случайный поиск — `RandomizedSearchCV`. Всё, что вычисляется по данным, включая стандартизацию, должно входить в конвейер: иначе статистики будут посчитаны с участием контрольных блоков, и оценка окажется завышенной. В scikit-learn метрики устроены так, что большее значение лучше, поэтому MSE задаётся со знаком минус (`neg_mean_squared_error`).

Листинг 5.1. Выбор степени многочлена кросс-валидацией и итоговая оценка на тестовой выборке

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler

X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, test_size=0.2, random_state=0)
model = make_pipeline(StandardScaler(), PolynomialFeatures(),
                      LinearRegression())
search = GridSearchCV(
    model, {"polynomialfeatures__degree": range(1, 11)},
    cv=5, scoring="neg_mean_squared_error")
search.fit(X_tr, y_tr)       # кросс-валидация на обучающей части
print(search.best_params_)
# тестовая выборка — один раз, в конце
y_pred = search.predict(X_te)
print(mean_squared_error(y_te, y_pred))
```

После выбора гиперпараметров `GridSearchCV` заново обучает лучшую модель на всей обучающей части; тестовая выборка в выборе не участвует, поэтому ошибка на ней не занижена выбором модели.

Вопросы для самоконтроля

1. Чем недообучение отличается от переобучения? Как каждое из них проявляется в ошибках на обучающей и контрольной выборках?
2. Почему ошибка на обучающей выборке не годится для выбора степени многочлена?
3. Что такое смещение и разброс? Как они меняются с ростом сложности модели?
4. Почему увеличение обучающей выборки уменьшает переобучение?
5. Чем кросс-валидация лучше отложенной выборки? Чем хуже?
6. Почему для временных рядов нельзя разбивать выборку случайно?
7. Зачем нужна тестовая выборка, если модель уже выбрана по валидационной?
8. Может ли R^2 быть отрицательным? Что это означает?

Задачи

1. Ответы на трёх контрольных объектах равны 2, 4, 6, прогнозы модели — 3, 4, 8. Вычислите MSE, RMSE, MAE, MAPE и R^2.
2. Выборка из 1000 объектов, 10-кратная кросс-валидация. Сколько раз обучается модель и на скольких объектах каждый раз? Сколько обучений потребует LOO?
3. Две модели на одной выборке из $\ell = 100$ объектов: у первой 3 параметра и MSE $= 2{,}0$, у второй 10 параметров и MSE $= 1{,}8$. Какую выберет AIC? BIC?
4. Модель предсказания нагрузки на сервис по дням обучили и проверили 5-кратной кросс-валидацией со случайным разбиением, ошибка мала. После внедрения ошибка оказалась втрое больше. Назовите вероятную причину.

6. Регуляризация

В гл. 3 и 5 переобучение проявлялось по-разному: при почти совпадающих признаках решение МНК менялось от малейшего изменения ответов, а многочлен высокой степени колебался между точками. В этой главе описано средство борьбы с таким переобучением — *регуляризация*: к эмпирическому риску добавляют штраф за величину весов.

6.1. Штраф за большие веса

Вспомним пример с почти совпадающими признаками из гл. 3. Изменение ответа y_3 всего на 0,1 (с 6,1 до 6,0 или до 6,2) переводило решение из $(1;\ 1)$ в $(0;\ 2)$ или $(2;\ 0)$, хотя все три решения почти одинаково точно описывали обучающую выборку. Среди моделей с почти равной ошибкой МНК выбирает случайно, под влиянием шума. Естественно добавить к ошибке требование, которое сделает выбор однозначным: при прочих равных предпочитать модель с меньшими весами.

Регуляризованный функционал качества имеет вид

$$Q_\lambda(\theta) = \|\mathbf{F}\theta - \mathbf{y}\|_2^2 + \lambda\, R(\theta),$$

где первое слагаемое — сумма квадратов остатков, $R(\theta)$ — штраф, растущий с величиной весов, а $\lambda \geq 0$ — *коэффициент регуляризации*. В этой главе сумма квадратов записана без множителя $1/\ell$, как в документации scikit-learn: множитель лишь меняет масштаб λ, а точка минимума остаётся прежней.

Читаем формулу. Слагаемые предъявляют к весам противоположные требования: первое — точно описать обучающую выборку, второе — оставаться малыми. Коэффициент λ задаёт, что важнее. При $\lambda = 0$ получается обычный МНК; при очень большом λ все веса стремятся к нулю, и модель предсказывает константу. Между этими крайностями — компромисс между смещением и разбросом (гл. 5): штраф уменьшает разброс модели ценой некоторого смещения. Коэффициент λ — гиперпараметр: метод обучения его не находит. Его выбирают перебором, как степень многочлена в гл. 5: для каждого значения из сетки оценивают ошибку модели кросс-валидацией и берут значение с наименьшей ошибкой.

Свободный член θ_0 не штрафуют: он задаёт средний уровень ответа и с переобучением не связан. Если бы его штрафовали, результат зависел бы от начала отсчёта ответов: после вычитания из всех цен 5 млн руб. прогнозы уменьшились бы не ровно на 5 млн. Поэтому суммы в штрафах ниже начинаются с $j = 1$. На практике признаки стандартизуют, а из

ответов вычитают их среднее $\bar{y}$; тогда $\theta_0 = \bar{y}$, а остальные веса находят по матрице $\mathbf{F}$ без столбца из единиц.

6.2. Гребневая регрессия

Если штраф — квадрат евклидовой нормы вектора весов, получается *гребневая регрессия* (ridge regression), или регуляризация Тихонова:

$$Q_\lambda(\theta) = \|\mathbf{F}\theta - \mathbf{y}\|_2^2 + \lambda\|\theta\|_2^2 = \sum_{i=1}^{\ell}\bigl(\langle\theta, \mathbf{x}_i\rangle - y_i\bigr)^2 + \lambda\sum_{j=1}^{n}\theta_j^2.$$

Градиент первого слагаемого найден в гл. 3 (без множителя $1/\ell$ он равен $2\mathbf{F}^{\mathrm{T}}(\mathbf{F}\theta - \mathbf{y})$), градиент второго равен $2\lambda\theta$. Приравняв сумму к нулю, получаем

$$(\mathbf{F}^{\mathrm{T}}\mathbf{F} + \lambda\mathbf{I})\theta = \mathbf{F}^{\mathrm{T}}\mathbf{y}, \qquad \theta_\lambda^* = (\mathbf{F}^{\mathrm{T}}\mathbf{F} + \lambda\mathbf{I})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y}, \tag{6.1}$$

где $\mathbf{I}$ — единичная матрица.

Читаем формулу. Отличие от нормальной системы МНК — слагаемое $\lambda\mathbf{I}$: к каждому диагональному элементу матрицы $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ прибавляется λ. Отсюда название: на диагонали вырастает «гребень». Матрица $\mathbf{F}^{\mathrm{T}}\mathbf{F} + \lambda\mathbf{I}$ при $\lambda > 0$ всегда обратима — даже если признаки линейно зависимы и решение МНК не единственно.

Пример. Проверим, как гребень действует на пример из гл. 3: $\mathbf{F}$ со строками (1; 1), (2; 2), (3; 3,1) и ответы $\mathbf{y} = (2; 4; 6{,}1)$. При $\lambda = 0{,}1$

$$\mathbf{F}^{\mathrm{T}}\mathbf{F} + 0{,}1\mathbf{I} = \begin{pmatrix} 14{,}1 & 14{,}3 \\ 14{,}3 & 14{,}71 \end{pmatrix}, \quad \mathbf{F}^{\mathrm{T}}\mathbf{y} = \begin{pmatrix} 28{,}3 \\ 28{,}91 \end{pmatrix}.$$

Определитель этой матрицы равен $14{,}1 \cdot 14{,}71 - 14{,}3^2 = 2{,}921$ — почти в 60 раз больше, чем 0,05 без гребня. Решение:

$$\theta_\lambda^* = \frac{1}{2{,}921}\begin{pmatrix} 14{,}71 \cdot 28{,}3 - 14{,}3 \cdot 28{,}91 \\ -14{,}3 \cdot 28{,}3 + 14{,}1 \cdot 28{,}91 \end{pmatrix} = \frac{1}{2{,}921}\begin{pmatrix} 2{,}880 \\ 2{,}941 \end{pmatrix} \approx \begin{pmatrix} 0{,}99 \\ 1{,}01 \end{pmatrix}.$$

Повторим расчёт для $y_3 = 6{,}0$ и $y_3 = 6{,}2$ и для $\lambda = 1$ (табл. 6.1).

Таблица 6.1. Решения МНК и гребневой регрессии при трёх значениях y_3

Величина	МНК, $\lambda = 0$	$\lambda = 0{,}1$	$\lambda = 1$
$y_3 = 6{,}0$	(2; 0)	(0,99; 0,98)	(0,95; 0,96)

Величина	МНК, $\lambda = 0$	$\lambda = 0{,}1$	$\lambda = 1$
$y_3 = 6{,}1$	(1; 1)	(0,99; 1,01)	(0,96; 0,98)
$y_3 = 6{,}2$	(0; 2)	(0,98; 1,03)	(0,96; 0,99)
Прогноз для объекта (2; 3)	от 4 до 6	от 4,92 до 5,06	от 4,79 до 4,89

Таблица показывает три эффекта. Решение стало устойчивым: при изменении y_3 на 0,1 веса меняются на сотые доли, а прогноз для нового объекта — на 3 % вместо 50 %. Гребень выбрал из всех почти равноценных решений то, в котором вес поровну разделён между двумя одинаковыми признаками, — это решение с наименьшей нормой. Наконец, с ростом λ растёт ошибка на обучающей выборке (при $y_3 = 6{,}1$ MSE равна 0 для МНК, 0,0002 при $\lambda = 0{,}1$ и 0,022 при $\lambda = 1$), и прогнозы систематически занижаются: при $\lambda = 1$ модель предсказывает для третьего объекта 5,89 вместо 6,1. Это смещение — цена устойчивости, поэтому λ не должен быть слишком большим.

Сингулярное разложение. Подставив $\mathbf{F} = \mathbf{V}\mathbf{D}\mathbf{U}^{\mathrm{T}}$ (гл. 3) в (6.1), получим

$$\theta_\lambda^* = \sum_{j=1}^{n} \frac{s_j}{s_j^2 + \lambda}\, \mathbf{u}_j\, (\mathbf{v}_j^{\mathrm{T}}\mathbf{y}).$$

В решении МНК на месте дроби стоял множитель $1/s_j$, который при малом сингулярном числе s_j становился огромным. Теперь при $s_j^2 \gg \lambda$ дробь почти равна $1/s_j$ — направления, хорошо определённые данными, гребень почти не трогает. А при $s_j^2 \ll \lambda$ дробь близка к s_j/λ и мала — неустойчивые направления подавляются. В примере $s_2 \approx 0{,}042$: множитель МНК $1/s_2 \approx 24$, а при $\lambda = 0{,}1$ он равен $0{,}042/(0{,}0017 + 0{,}1) \approx 0{,}41$. Число обусловленности матрицы системы уменьшается с $s_1^2/s_2^2 \approx 16\,400$ до $(s_1^2 + \lambda)/(s_2^2 + \lambda) \approx 280$.

6.3. LASSO

Если штраф — сумма модулей весов, получается метод LASSO (least absolute shrinkage and selection operator):

$$Q_\lambda(\theta) = \|\mathbf{F}\theta - \mathbf{y}\|_2^2 + \lambda\|\theta\|_1 = \sum_{i=1}^{\ell} \big(\langle \theta, \mathbf{x}_i\rangle - y_i\big)^2 + \lambda \sum_{j=1}^{n} |\theta_j|.$$

Модуль не дифференцируем в нуле, и формулы решения, как (6.1), нет;

минимум ищут численно. Зато у LASSO есть свойство, которого нет у гребневой регрессии: при достаточно большом λ часть весов становится *в точности* нулевой. Признаки с нулевыми весами модель не использует, поэтому LASSO одновременно обучает модель и отбирает признаки.

Почему веса зануляются. Рассмотрим простейший случай: столбцы $\mathbf{F}$ ортонормированы, $\mathbf{F}^{\mathrm{T}}\mathbf{F} = \mathbf{I}$. Тогда решение МНК — вектор $\mathbf{a} = \mathbf{F}^{\mathrm{T}}\mathbf{y}$, а сумма квадратов остатков раскладывается на независимые слагаемые: $\|\mathbf{F}\theta - \mathbf{y}\|_2^2 = \sum_j (\theta_j - a_j)^2 + \text{const}$. Каждый вес находится отдельно. Для гребневой регрессии минимум $(\theta_j - a_j)^2 + \lambda\theta_j^2$ даёт

$$\theta_j = \frac{a_j}{1+\lambda}$$

— вес МНК уменьшается в одно и то же число раз и нулём не становится. Для LASSO при $\theta_j > 0$ производная функции $(\theta_j - a_j)^2 + \lambda|\theta_j|$ равна $2(\theta_j - a_j) + \lambda$, она обращается в нуль при $\theta_j = a_j - \lambda/2$, но только если $a_j > \lambda/2$; случай $\theta_j < 0$ симметричен. Если же $|a_j| \le \lambda/2$, минимум достигается в нуле. Итого

$$\theta_j = \operatorname{sign}(a_j)\max\bigl(|a_j| - \lambda/2,\ 0\bigr).$$

Читаем формулы. Гребневая регрессия *сжимает* все веса пропорционально, LASSO *сдвигает* их к нулю на постоянную величину $\lambda/2$, а веса меньше $\lambda/2$ по модулю обнуляет. Причина — в производной штрафа. Производная $\lambda\theta_j^2$ равна $2\lambda\theta_j$ и вблизи нуля исчезает: штраф почти не мешает малому весу оставаться ненулевым. Производная $\lambda|\theta_j|$ по модулю всегда равна λ: штраф тянет вес к нулю с постоянной силой, и если данные «тянут» его в другую сторону слабее, вес становится нулевым.

6.4. Геометрический смысл

Требование «предпочитать малые веса» можно выразить двумя способами. *Мягкое ограничение* — штраф: веса могут быть любыми, но за большие веса приходится платить, и минимизируется Q_λ. *Жёсткое ограничение* — веса обязаны лежать в заданной области, а внутри неё ищется минимум суммы квадратов остатков:

$$\|\mathbf{F}\theta - \mathbf{y}\|_2^2 \to \min_{\theta} \quad \text{при условии} \quad \|\theta\|_2^2 \le t^2$$

для гребневой регрессии и при условии $\|\theta\|_1 \le t$ для LASSO. Число t — радиус допустимой области. Две постановки эквивалентны: каждому $\lambda > 0$ соответствует такое t, что решения задач совпадают, и наоборот;

чем больше λ, тем меньше t.

Почему постановки эквивалентны. Рассмотрим гребневую регрессию. Если решение МНК лежит внутри круга радиуса t, ограничение ничего не меняет, и это соответствует $\lambda = 0$. Иначе решение задачи с ограничением лежит на границе круга, в точке, где линия уровня суммы квадратов остатков касается окружности. Сумма квадратов убывает в сторону решения МНК, то есть за пределы круга, и в точке касания её антиградиент направлен по радиусу наружу. Значит, градиенты суммы квадратов и штрафа $\|\theta\|_2^2$ (он направлен по радиусу наружу) противоположны: $\nabla\|\mathbf{F}\theta - \mathbf{y}\|_2^2 = -\lambda\nabla\|\theta\|_2^2$ при некотором $\lambda > 0$. Но это равенство и есть условие минимума Q_λ: градиент суммы $\|\mathbf{F}\theta - \mathbf{y}\|_2^2 + \lambda\|\theta\|_2^2$ равен нулю. Решение задачи с ограничением — минимум функционала со штрафом при этом λ. Число λ называют *множителем Лагранжа*, а сам приём — методом множителей Лагранжа.

Как штраф сдвигает решение, проще всего увидеть на модели с одним признаком и одним весом: $g(x, \theta) = \theta f(x)$, без свободного члена (рис. 6.1). Пусть МНК даёт для неё вес $\theta = 2$, а сумма квадратов остатков с точностью до постоянного слагаемого равна $(\theta - 2)^2$ — так получается, когда столбец признака имеет единичную длину, как в случае ортонормированных признаков из разд. 6.3. Штраф — ещё одна функция θ с минимумом в нуле, и минимум функционала Q_λ — суммы квадратов остатков и штрафа — оказывается между нулём и решением МНК. Для гребневой регрессии он равен $2/(1 + \lambda)$: 1 при $\lambda = 1$, 0,5 при $\lambda = 3$, — и в нуль не попадает ни при каком λ. Для LASSO минимум равен $\max(2 - \lambda/2,\ 0)$: 1 при $\lambda = 2$ и ровно 0 уже при $\lambda = 4$, потому что излом $|\theta|$ в нуле удерживает там минимум Q_λ.

Постановка с жёстким ограничением наглядна и для двух весов (рис. 6.2). Сумма квадратов остатков теперь зависит от двух весов, и её график — поверхность над плоскостью (θ_1, θ_2), своего рода чаша. Чтобы изобразить её на плоскости, рисуют *линии уровня* — кривые, вдоль которых сумма постоянна, как горизонтали на топографической карте. У суммы квадратов остатков линии уровня — эллипсы с центром в решении МНК (гл. 3). Допустимая область для гребневой регрессии — круг, для LASSO — ромб с вершинами на осях. Решение — точка допустимой области, лежащая на самом «низком» эллипсе, то есть точка, где расширяющийся от центра эллипс впервые касается области. Круг гладкий, и эллипс касается его в произвольной точке, где обе координаты, как правило, ненулевые. У ромба есть углы, выступающие вдоль осей, и эллипс чаще всего касается ромба именно в углу — там, где одна из координат равна нулю (на рис. 6.2, б $\theta_2 = 0$). При многих

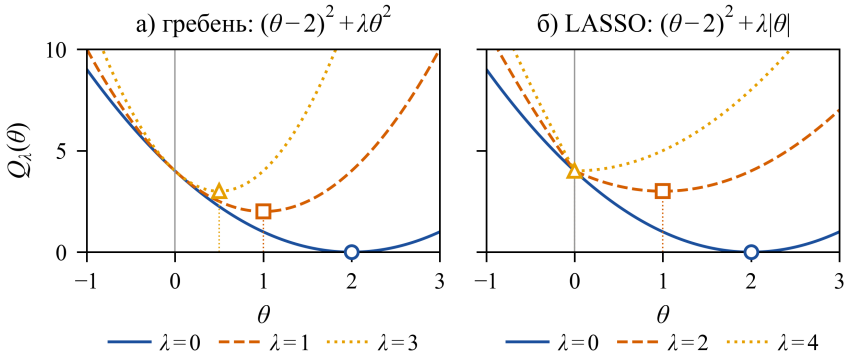


Рис. 6.1. Мягкое ограничение для модели с одним весом θ: минимум суммы квадратов остатков и штрафа сдвигается к нулю с ростом λ; а — гребневая регрессия; б — LASSO

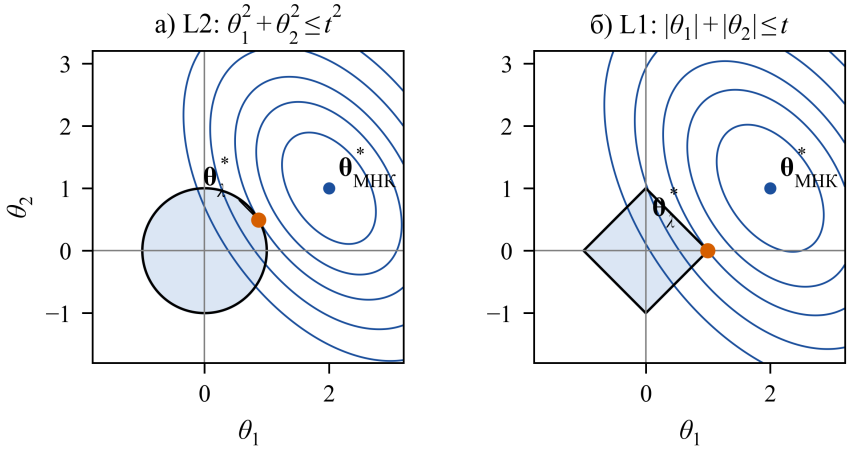


Рис. 6.2. Решение МНК с ограничением: а — на круг (гребневая регрессия); б — на ромб (LASSO)

признаках у многомерного «ромба» много вершин и рёбер, лежащих в координатных плоскостях, поэтому нулевыми оказываются сразу несколько весов.

6.5. Регуляризация на практике

Масштабирование признаков обязательно. Штраф одинаков для всех весов, а величина веса зависит от единиц измерения признака: если площадь выразить не в квадратных метрах, а в сотнях квадратных метров, её вес вырастет в 100 раз, и штраф на него — в 10^4 раз (для гребневой регрессии). Без стандартизации регуляризация наказывает признаки неравномерно, в зависимости от выбора единиц измерения. Поэтому перед регуляризацией признаки стандартизуют (гл. 2).

Пути регуляризации. Чтобы увидеть, как штраф меняет модель, строят графики весов в зависимости от λ. На рис. 6.3 — синтетические данные о квартирах с пятью стандартизованными признаками: площадь, жилая площадь (почти копия площади, корреляция 0,98), расстояние до метро, этаж и случайный признак; цена зависит только от площади и расстояния до метро. МНК (λ близко к нулю) даёт жилой площади отрицательный вес $-0{,}78$, хотя с ценой она связана положительно, — знакомое по гл. 3 проявление мультиколлинеарности. В гребневой регрессии с ростом λ вес жилой площади становится положительным, и веса двух почти одинаковых признаков сближаются: гребень делит вклад между ними. В LASSO вес жилой площади при $\lambda \approx 10$ обнуляется первым, затем обнуляются веса этажа и случайного признака, потом расстояния до метро; дольше всех остаётся площадь. LASSO выбирает из двух одинаковых признаков один.

Выбор λ. Коэффициент регуляризации перебирают по сетке в логарифмической шкале, например $10^{-3}, 10^{-2}, \dots, 10^{3}$, оценивают ошибку при каждом значении кросс-валидацией (гл. 5) и выбирают значение с наименьшей ошибкой. Слишком малое λ оставляет переобучение, слишком большое приводит к недообучению; ошибка на валидации как функция λ обычно имеет минимум посередине. В scikit-learn для этого есть классы `RidgeCV` и `LassoCV` (листинг 6.1). Следует учесть, что масштаб коэффициента в разных классах разный: в `Ridge` параметр `alpha` совпадает с λ, а в `Lasso` сумма квадратов остатков делится на 2ℓ, так что $\lambda = 2\ell \cdot$ `alpha`.

Листинг 6.1. Выбор коэффициента регуляризации по оценке кросс-валидации

```
import numpy as np
from sklearn.linear_model import LassoCV, RidgeCV
```

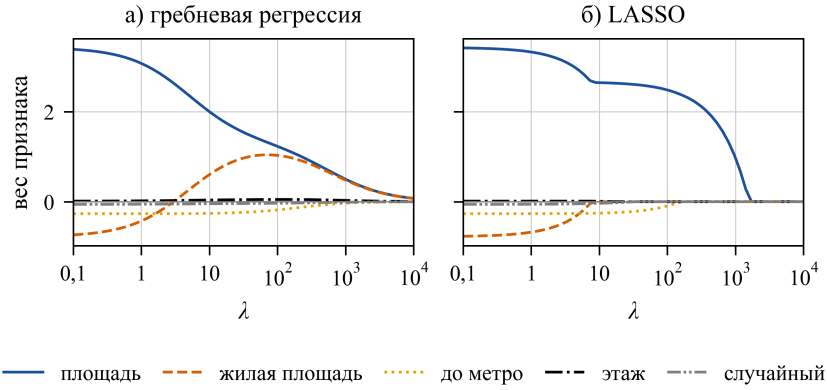


Рис. 6.3. Веса признаков в зависимости от коэффициента регуляризации (синтетические данные)

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

ridge = make_pipeline(StandardScaler(),
                      RidgeCV(alphas=np.logspace(-3, 3, 13)))
lasso = make_pipeline(StandardScaler(), LassoCV(cv=5))
ridge.fit(X_train, y_train)
lasso.fit(X_train, y_train)
print(ridge[-1].alpha_)         # выбранный коэффициент
# сколько признаков осталось в модели LASSO
print(np.sum(lasso[-1].coef_ != 0))
```

Эластичная сеть. LASSO из нескольких сильно коррелированных признаков оставляет один, причём какой именно — зависит от случайностей выборки. Если нужно сохранить все признаки, используют *эластичную сеть* (elastic net) со штрафом $\lambda_1\|\theta\|_1 + \lambda_2\|\theta\|_2^2$: L1-часть отбирает признаки, L2-часть распределяет вес между коррелированными признаками, как в гребневой регрессии. В scikit-learn это класс `ElasticNet`.

Регуляризация в градиентном спуске. Если параметры ищут градиентным спуском (гл. 4), к градиенту добавляется производная штрафа. Для гребневой регрессии она равна $2\lambda\theta$, и шаг принимает вид

$$\theta := (1 - 2\eta\lambda)\,\theta - \eta\,\nabla\|\mathbf{F}\theta - \mathbf{y}\|_2^2.$$

Перед обычным шагом все веса умножаются на число чуть меньше единицы, поэтому такую регуляризацию называют *сокращением весов*

(weight decay). На каждом шаге сокращение тянет все веса к нулю, а градиент ошибки удерживает те из них, которые уменьшают ошибку на обучающей выборке. Вес признака, который почти не влияет на ошибку, ничто не удерживает, и он убывает как геометрическая прогрессия со знаменателем $1 - 2\eta\lambda$. Этот приём используется и в моделях, для которых формулы решения нет, — от логистической регрессии до нейронных сетей.

Вопросы для самоконтроля

1. Какую задачу решает регуляризация? Что происходит при $\lambda = 0$ и при очень большом λ?
2. Почему свободный член θ_0 не штрафуют?
3. Почему матрица $\mathbf{F}^{\mathrm{T}}\mathbf{F} + \lambda\mathbf{I}$ обратима при любом $\lambda > 0$?
4. Как гребневая регрессия распределяет вес между двумя одинаковыми признаками? Как это делает LASSO?
5. Объясните, почему LASSO может обнулять веса, а гребневая регрессия — нет.
6. Почему перед регуляризацией признаки нужно стандартизовать?
7. Как выбирают коэффициент регуляризации?

Задачи

1. Для примера из табл. 6.1 найдите решение гребневой регрессии при $\lambda = 1$ и $\mathbf{y} = (2; 4; 6{,}1)$.
2. Признаки ортонормированы, решение МНК $\mathbf{a} = (3; -0{,}5; 1{,}2)$. Найдите веса гребневой регрессии и LASSO при $\lambda = 2$.
3. Один признак умножили на 10 (перешли к другим единицам), остальные не меняли. Как изменятся вес этого признака в МНК и в гребневой регрессии?

7. Логистическая регрессия

В предыдущих главах решалась задача регрессии. Теперь перейдём к классификации с двумя классами, $Y = \{-1, +1\}$: есть ли в модуле программы ошибка, вернёт ли заёмщик кредит, злокачественна ли опухоль. Логистическая регрессия — линейная модель, которая не только относит объект к классу, но и оценивает вероятность этого класса. Её обучение сводится к минимизации эмпирического риска с особой функцией потерь, и в этой главе показано, откуда эта функция берётся.

7.1. Почему не подходит линейная регрессия

Можно попробовать обучить на метках ± 1 обычную линейную регрессию и относить объект к классу $+1$, если прогноз положителен. На рис. 7.1 так классифицируются опухоли по размеру: четыре доброкачественные ($y = -1$) размером от 1 до 2,5 см и четыре злокачественные ($y = +1$) размером от 3,5 до 5 см. Прямая $g(x) = \theta_0 + \theta_1 x$, найденная методом наименьших квадратов (гл. 3), обращается в нуль при $x = 3$ см; это и есть порог классификации, и он разделяет классы без ошибок.

Добавим ещё одну злокачественную опухоль размером 14 см. Классифицировать её легко, но прежняя прямая дала бы для неё прогноз около $+5$, и МНК, приближая прогноз к метке $+1$, уменьшает наклон прямой. Порог сдвигается до 3,53 см, и опухоль размером 3,5 см теперь ошибочно считается доброкачественной. Квадратичная потеря штрафует модель за то, что на очевидном объекте прогноз «слишком правильный», и ради этого жертвует объектами у границы. Для классификации нужна функция потерь, которая не наказывает уверенные верные ответы.

7.2. Линейный классификатор

Линейный классификатор для двух классов (гл. 1) имеет вид

$$g(x, \theta) = \operatorname{sign} h(x, \theta), \qquad h(x, \theta) = \langle \theta, \mathbf{x} \rangle = \sum_{j=0}^{n} \theta_j f_j(x),$$

где h — *разделяющая функция*. Объекты, для которых $h(x, \theta) = 0$, образуют *разделяющую поверхность*: по одну её сторону классификатор отвечает $+1$, по другую — -1. Для линейной h вид поверхности зависит от числа признаков. Если признаков два, каждый объект — точка на плоскости, и разделяющая поверхность — прямая на этой плоскости. Если

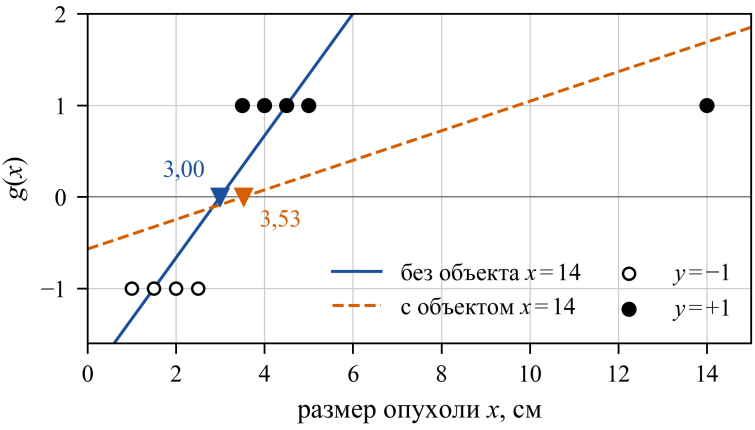


Рис. 7.1. Линейная регрессия на метках классов: далёкий объект сдвигает порог

признаков три, пространство признаков трёхмерное, и поверхность — плоскость. При n признаках это гиперплоскость — «плоскость» размерности $n-1$ в n-мерном пространстве признаков.

Пример. Пусть у объекта два признака, x_1 и x_2, и $\theta = (-3;\ 1;\ 1)$. Тогда $h = -3 + x_1 + x_2$, и разделяющая поверхность — прямая $x_1 + x_2 = 3$ (рис. 7.2, а). Объект $(1;\ 1)$ даёт $h = -1$ и относится к классу -1, объект $(2;\ 2)$ даёт $h = 1$ и относится к классу $+1$. Величина $|h|$ пропорциональна расстоянию от объекта до прямой: она показывает, насколько уверенно классификатор относит объект к классу.

Модель линейна по параметрам, но не обязана быть линейной по исходным признакам — как и в полиномиальной регрессии из гл. 3. Если классы разделены окружностью (рис. 7.2, б), возьмём признаки $f_1 = x_1^2$ и $f_2 = x_2^2$. При $\theta = (-1;\ 1;\ 1)$ получаем $h = -1 + x_1^2 + x_2^2$, и разделяющая поверхность — окружность $x_1^2 + x_2^2 = 1$.

7.3. Отступ и функции потерь

Отступом объекта x_i называют величину

$$M_i(\theta) = y_i\, h(x_i, \theta).$$

Отступ положителен, если знак h совпадает с меткой, то есть классификатор отвечает верно, и отрицателен при ошибке. Его модуль — та же уверенность $|h|$: большой положительный отступ означает уверенный вер-

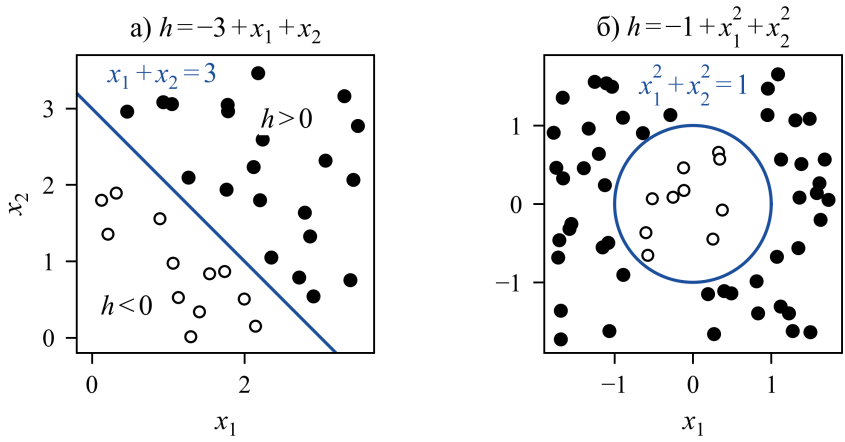


Рис. 7.2. Разделяющие поверхности: а — прямая; б — окружность, полученная линейной моделью от квадратов признаков

ный ответ, большой отрицательный — грубую ошибку. Благодаря обозначению классов ± 1 одна формула описывает оба класса.

Индикатор ошибки из гл. 1 через отступ записывается как *пороговая функция потерь* $[M_i < 0]$, а эмпирический риск — как доля ошибок. Минимизировать её напрямую трудно: функция ступенчатая, её производная почти везде равна нулю, и градиентный спуск (гл. 4) не знает, куда двигаться. Поэтому пороговую функцию заменяют непрерывной функцией $\mathcal{L}(M)$ без скачка, которая лежит не ниже неё (рис. 7.3). Тогда сумма $\mathcal{L}(M_i)$ — верхняя оценка числа ошибок, и, уменьшая её, мы уменьшаем и число ошибок.

Три верхние оценки на рис. 7.3 — логистическая, кусочно-линейная и экспоненциальная — убывают с ростом отступа (кусочно-линейная — до $M = 1$, дальше она равна нулю). В отличие от пороговой функции, они штрафуют не только ошибки ($M < 0$), но и неуверенные верные ответы вблизи границы. При большом положительном отступе потеря близка к нулю — именно этого не хватало квадратичной потере в разд. 7.1. Логистическая функция $\ln(1 + e^{-M})$ (на рисунке — с основанием логарифма 2, чтобы она проходила через точку (0; 1); от основания зависит только множитель) приводит к логистической регрессии; кусочно-линейная $\max(0, 1 - M)$ — к методу опорных векторов; экспоненциальная e^{-M} используется в бустинге. Выбор логистической функции кажется произвольным, но у неё есть вероятностный смысл, который раскрыт

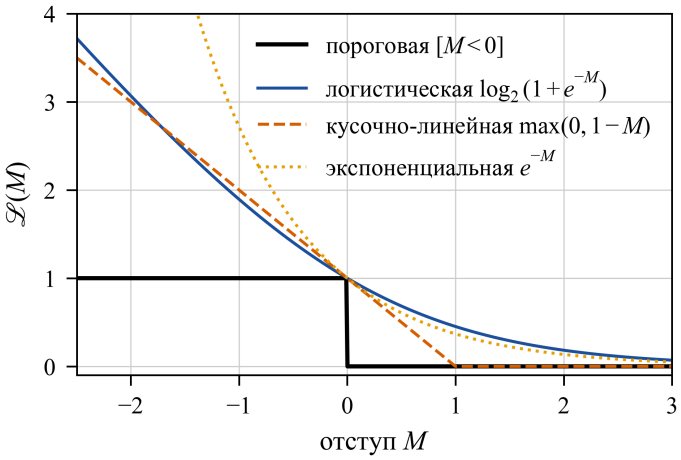


Рис. 7.3. Пороговая функция потерь и её верхние оценки как функции отступа

в двух следующих разделах.

7.4. Сигмоида и вероятность класса

Сигмоида (логистическая функция)

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

переводит любое число в интервал (0; 1): $\sigma(0) = 0{,}5$, $\sigma(z) \to 1$ при $z \to +\infty$ и $\sigma(z) \to 0$ при $z \to -\infty$ (рис. 7.4). У неё два полезных свойства, которые проверяются прямой подстановкой:

$$\sigma(-z) = 1 - \sigma(z), \qquad \sigma'(z) = \sigma(z)\big(1-\sigma(z)\big).$$

Логистическая регрессия представляет вероятность класса $+1$ как сигмоиду от разделяющей функции:

$$P(y = +1 \mid x) = \sigma\big(h(x, \theta)\big). \tag{7.1}$$

Чем дальше объект от разделяющей поверхности в сторону класса $+1$, тем ближе вероятность к единице; на самой поверхности она равна 0,5. Вероятность класса -1 по первому свойству сигмоиды равна $1 - \sigma(h) = \sigma(-h)$. Обе формулы объединяются в одну:

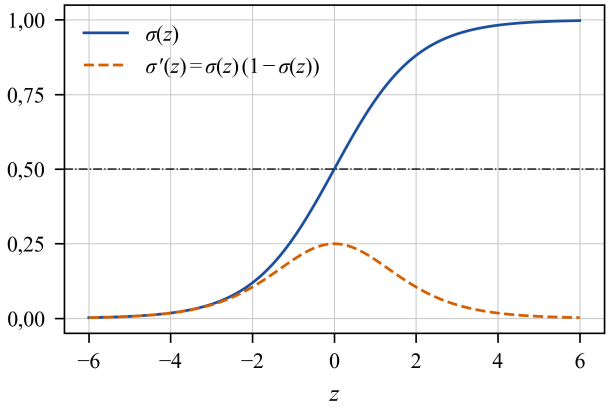


Рис. 7.4. Сигмоида и её производная; штрихпунктир — уровень 0,5, который сигмоида проходит при $z = 0$

$$P(y \mid x) = \sigma\big(y\, h(x, \theta)\big) = \sigma(M), \quad y \in \{-1, +1\}. \tag{7.2}$$

Вероятность правильного ответа — сигмоида от отступа.

Метки 0 и 1. В scikit-learn и большинстве учебников классы обозначают $\tilde{y} \in \{0, 1\}$, где $\tilde{y} = (y + 1)/2$. Вероятность класса 1, которую модель предсказывает для объекта x, обозначим $\hat{p}$:

$$\hat{p} = P(\tilde{y} = 1 \mid x) = \sigma\big(h(x, \theta)\big).$$

Это та же вероятность, что в (7.1): событие $\tilde{y} = 1$ совпадает с событием $y = +1$. Знак «крышка» над буквой, как в $\hat{y}$, напоминает, что это прогноз модели — её оценка вероятности, а не сама вероятность; для i-го объекта пишем $\hat{p}_i$. Вероятность наблюдаемой метки

$$P(\tilde{y} \mid x) = \hat{p}^{\tilde{y}}(1 - \hat{p})^{1-\tilde{y}} \tag{7.3}$$

(при $\tilde{y} = 1$ остаётся $\hat{p}$, а при $\tilde{y} = 0$ остаётся $1 - \hat{p}$). Это та же модель, что (7.2), в другой записи. Чтобы получить класс, вероятность сравнивают с порогом: при пороге 0,5 класс 1 выбирается, если $\hat{p} \geq 0{,}5$, то есть $h \geq 0$, — это прежний классификатор $\operatorname{sign} h$. Если ошибки разных типов имеют разную цену, порог сдвигают.

Шансы и смысл коэффициентов. *Шансы* класса — отношение $\hat{p}/(1 - \hat{p})$: шансы 3 означают, что класс 1 втрое вероятнее класса 0. Подставим $\hat{p} = \sigma(h) = 1/(1 + e^{-h})$. Тогда $1 - \hat{p} = e^{-h}/(1 + e^{-h})$, знаменатели

при делении сокращаются, и шансы равны

$$\frac{\hat{p}}{1-\hat{p}} = \frac{1}{e^{-h}} = e^{h}.$$

Логарифмируя, получаем

$$\ln \frac{\hat{p}}{1-\hat{p}} = h(x, \theta) = \theta_0 + \theta_1 f_1(x) + \ldots + \theta_n f_n(x).$$

Логарифм шансов называют *логитом*; логистическая регрессия — линейная модель логита. Отсюда смысл коэффициентов: увеличение признака f_j на единицу при прочих равных прибавляет к логиту θ_j, то есть умножает шансы на e^{θ_j}. В примере с $\theta = (-3;\ 1;\ 1)$ объект $(1;\ 1)$ имеет $h = -1$; вероятность класса 1 для него равна $\sigma(-1) \approx 0{,}27$, а шансы — $e^{-1} \approx 0{,}37$; увеличение x_1 на единицу умножает шансы на $e \approx 2{,}72$, и у объекта $(2;\ 1)$ шансы равны 1, а вероятность — 0,5.

7.5. Максимум правдоподобия

Как подобрать параметры? Естественное требование: модель должна приписывать наблюдаемым в выборке меткам высокую вероятность. Если объекты выборки независимы, вероятность получить все метки сразу равна произведению вероятностей отдельных меток. Эту величину как функцию параметров называют *правдоподобием* и обозначают L (от англ. likelihood):

$$L(\theta) = \prod_{i=1}^{\ell} P(y_i \mid x_i, \theta).$$

Правдоподобие тоже оценивает, насколько параметры согласуются с выборкой, но, в отличие от функционала качества Q (гл. 1), его не минимизируют, а максимизируют; ниже одна задача сведена к другой. *Принцип максимума правдоподобия* — выбрать параметры, при которых правдоподобие наибольшее. Произведение многих чисел меньше единицы неудобно (оно быстро становится неразличимо малым), поэтому максимизируют логарифм — он превращает произведение в сумму и не меняет точку максимума. Максимизация $\ln L$ равносильна минимизации $-\ln L$. Подставив (7.2), получаем

$$-\ln L(\theta) = -\sum_{i=1}^{\ell} \ln \sigma(M_i) = \sum_{i=1}^{\ell} \ln\bigl(1 + e^{-M_i}\bigr). \tag{7.4}$$

Читаем формулу. Слева — минус логарифм правдоподобия, справа — сумма логистических потерь из разд. 7.3. Значит, $Q(\theta) = -\frac{1}{\ell}\ln L(\theta)$ — эмпирический риск с логистической функцией потерь, и максимизация правдоподобия и минимизация Q — одна и та же задача. Логистическая функция потерь не выбрана произвольно: она равна минус логарифму вероятности, которую модель приписывает правильному ответу. Уверенный верный ответ ($\sigma(M) \approx 1$) почти ничего не стоит, уверенная ошибка ($\sigma(M) \approx 0$) стоит очень дорого.

Каждое слагаемое суммы (7.4) — потеря на одном объекте, минус логарифм вероятности его метки: $-\ln P(y_i \mid x_i)$. Если записать эту вероятность через метки $\tilde{y} \in \{0, 1\}$ по формуле (7.3) и прологарифмировать, получится потеря, которую называют *бинарной кросс-энтропией* (log-loss):

$$\mathcal{L} = -\ln P(\tilde{y} \mid x) = -\big[\tilde{y}\ln\hat{p} + (1-\tilde{y})\ln(1-\hat{p})\big], \quad \hat{p} = \sigma(h).$$

Это та же логистическая потеря $\ln(1+e^{-M})$, только записанная через $\tilde{y}$ и $\hat{p}$; её среднее по выборке — тот же функционал $Q(\theta)$. Из двух слагаемых ненулевое только одно: при $\tilde{y} = 1$ потеря равна $-\ln\hat{p}$, а при $\tilde{y} = 0$ она равна $-\ln(1-\hat{p})$.

Тот же принцип объясняет и метод наименьших квадратов: если в регрессии шум нормально распределён, $y = g(x, \theta) + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$, то минус логарифм вероятности ответа равен $(y-g)^2/(2\sigma^2)$ плюс константа, и максимум правдоподобия совпадает с минимумом суммы квадратов.

Пример. Сравним по правдоподобию две модели $h = \theta_0 + \theta_1 x$ на трёх объектах с одним признаком: $x = 1, 2, 3$ и метками $-1, +1, +1$. Модель $\theta = (-3;\ 2)$ даёт $h = -1;\ 1;\ 3$ и отступы $1;\ 1;\ 3$. Вероятности правильных ответов — $\sigma(1) = 0{,}731;\ 0{,}731;\ \sigma(3) = 0{,}953$, правдоподобие $L = 0{,}731 \cdot 0{,}731 \cdot 0{,}953 \approx 0{,}509$, а $-\ln L = 0{,}313 + 0{,}313 + 0{,}049 = 0{,}675$. Модель $\theta = (-1{,}5;\ 1)$ с той же разделяющей точкой $x = 1{,}5$ даёт отступы $0{,}5;\ 0{,}5;\ 1{,}5$, правдоподобие $0{,}317$ и $-\ln L = 1{,}15$. Обе модели классифицируют все объекты верно, но первая увереннее, и правдоподобие её предпочитает.

Продолжим пример: модель $\theta = (-6;\ 4)$ — та же граница, веса вдвое больше — даёт $-\ln L = 0{,}256$, ещё лучше. Если классы в выборке разделимы, умножение весов на любое число больше единицы увеличивает все отступы и правдоподобие, и минимум (7.4) не достигается: веса растут бесконечно. Это крайний случай переобучения, и защищает от него регуляризация (разд. 7.6).

Градиент. Решения в явном виде у задачи нет, её решают градиентным спуском или более быстрыми методами второго порядка. Вычислим градиент потери на одном объекте в записи с метками $\tilde{y} \in \{0, 1\}$. По правилу дифференцирования сложной функции и свойству $\sigma' = \sigma(1-\sigma)$:

$$\frac{\partial \mathcal{L}}{\partial h} = -\tilde{y}\,\frac{\sigma(1-\sigma)}{\sigma} + (1-\tilde{y})\,\frac{\sigma(1-\sigma)}{1-\sigma} = \hat{p} - \tilde{y},$$

а так как $\partial h/\partial \theta = \mathbf{x}$, градиент равен $(\hat{p} - \tilde{y})\,\mathbf{x}$. Для средней потери по выборке

$$\nabla Q(\theta) = \frac{1}{\ell}\mathbf{F}^{\mathrm{T}}(\hat{\mathbf{p}} - \tilde{\mathbf{y}}),$$

где $\hat{\mathbf{p}}$ — вектор вероятностей $\sigma(h(x_i, \theta))$. Формула той же формы, что градиент квадратичной потери $\frac{2}{\ell}\mathbf{F}^{\mathrm{T}}(\mathbf{F}\theta - \mathbf{y})$ из гл. 3: столбцы признаков, умноженные на «ошибки» модели, только ошибка теперь — разность между предсказанной вероятностью и меткой. Функция (7.4) выпукла, поэтому при подходящей регуляризации у неё один минимум, и спуск находит его из любой начальной точки.

7.6. Вероятностный смысл регуляризации

Регуляризацию из гл. 6 тоже можно получить из вероятностных соображений. Будем считать параметры случайными и до наблюдения данных зададим их *априорное распределение* с плотностью $p(\theta)$ — представление о том, какие значения параметров правдоподобны. Обычно веса считают скорее малыми, чем большими. Выбирают параметры, при которых велико произведение правдоподобия данных и априорной плотности; логарифмируя, получаем задачу

$$-\ln L(\theta) - \ln p(\theta) \to \min_{\theta}.$$

Это *принцип максимума апостериорной вероятности* (maximum a posteriori, MAP): максимизируется апостериорная вероятность параметров при данной выборке, а минимизируемый функционал — её минус логарифм. Первое слагаемое — эмпирический риск, второе играет роль штрафа.

Пусть веса $\theta_1, \ldots, \theta_n$ независимы и распределены нормально со средним 0 и дисперсией C: $p(\theta_j) \propto \exp(-\theta_j^2/(2C))$. Тогда $-\ln p(\theta) = \|\theta\|_2^2/(2C) + \text{const}$ — штраф гребневой регрессии с $\lambda = 1/(2C)$. Если веса распределены по закону Лапласа, $p(\theta_j) \propto \exp(-|\theta_j|/C)$, получается штраф для LASSO $\|\theta\|_1/C$. Распределение Лапласа сильнее сосредоточено около нуля, чем нормальное, и это согласуется с тем, что LASSO обнуляет веса.

Читаем формулы. Коэффициент регуляризации обратен дисперсии

априорного распределения: малая дисперсия C означает твёрдую уверенность, что веса близки к нулю, — сильная регуляризация; большая C — регуляризация слабая. В scikit-learn логистическая регрессия параметризована именно так: параметр `C` класса `LogisticRegression` — обратный коэффициент регуляризации (по умолчанию $C = 1$ со штрафом L2), и при уменьшении `C` регуляризация усиливается.

7.7. Несколько классов

Если классов $K > 2$, $Y = \{1, \dots, K\}$, есть два подхода. *Один против всех* (one-vs-rest): обучают K бинарных классификаторов, k-й отделяет класс k от остальных, и объект относят к классу, чей классификатор дал наибольшую вероятность. *Многоклассовая логистическая регрессия* заводит для каждого класса свою разделяющую функцию $h_k(x) = \langle \theta_k, \mathbf{x} \rangle$ и переводит их в вероятности функцией *softmax*:

$$P(y = k \mid x) = \frac{e^{h_k(x)}}{\sum_{c=1}^{K} e^{h_c(x)}}, \quad k = 1, \dots, K.$$

Экспонента делает все величины положительными, а после деления на знаменатель они в сумме дают единицу, как и положено вероятностям. Класс с наибольшей h_k получает наибольшую вероятность. Параметры находят тем же максимумом правдоподобия: потеря на объекте равна $-\ln P(y_i \mid x_i)$ — многоклассовая кросс-энтропия. При $K = 2$ softmax превращается в сигмоиду от разности $h_1 - h_2$.

В scikit-learn оба случая обрабатывает класс `LogisticRegression` (листинг 7.1). Метод `predict` возвращает класс, `predict_proba` — вероятности классов, а экспоненты коэффициентов `coef_` показывают, во сколько раз меняются шансы при увеличении стандартизованного признака на одно стандартное отклонение.

Листинг 7.1. Логистическая регрессия в scikit-learn

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

model = make_pipeline(StandardScaler(),
                      LogisticRegression(C=1.0))
model.fit(X_train, y_train)
proba = model.predict_proba(X_test)[:, 1]   # P(класс 1)
y_pred = model.predict(X_test)              # порог 0,5
odds = np.exp(model[-1].coef_)              # множители шансов
```

Вопросы для самоконтроля

1. Почему линейная регрессия плохо подходит для классификации?
2. Что такое отступ? Какой знак у отступа при ошибке классификатора?
3. Почему пороговую функцию потерь заменяют гладкой?
4. Почему вероятность правильного ответа в логистической регрессии равна $\sigma(M)$?
5. Как связаны максимум правдоподобия и минимизация логистической потери?
6. Как истолковать коэффициент $\theta_j = 0{,}7$ логистической регрессии?
7. Что происходит с весами логистической регрессии, если классы разделимы, а регуляризация отсутствует?
8. Какому априорному распределению весов соответствует штраф L2? L1?

Задачи

1. Для $\theta = (-3;\ 1;\ 1)$ найдите вероятность класса $+1$ и шансы для объектов $(3;\ 1)$ и $(0;\ 1)$.
2. Вычислите бинарную кросс-энтропию для объекта с меткой $\tilde{y} = 1$ при прогнозах $\hat{p} = 0{,}9;\ 0{,}5;\ 0{,}1$.
3. Покажите, что при $K = 2$ softmax даёт $P(y = 1 \mid x) = \sigma(h_1 - h_2)$.
4. Сделайте один шаг градиентного спуска для потери на одном объекте $x = 2$, $\tilde{y} = 1$ из $\theta = (0;\ 0)$ при $\eta = 0{,}5$ (признаки — $f_0 = 1$ и x).

8. Метрики качества классификации

Логистическая регрессия обучается минимизацией логистической потери, но о качестве классификатора судят по другим величинам. Какие именно величины выбрать, зависит от задачи: врача интересует, сколько больных пациентов модель пропустила, банк — сколько надёжных заёмщиков она отвергла, тестировщика — сколько модулей с ошибками она нашла (на примере модулей программы из гл. 1 построена и эта глава). В этой главе описаны метрики качества классификации, их связь с порогом, по которому вероятность превращается в класс, и правила выбора метрики под задачу.

8.1. Функция потерь и метрика качества

Функция потерь служит для обучения: она должна быть удобна для решения задачи оптимизации — быть гладкой, с вычислимым градиентом. *Метрика качества* служит для оценки: она должна отражать то, что важно в задаче, и быть понятной людям, которые принимают решения по модели. Функция, по которой модель обучают, и метрика, по которой её оценивают, не обязаны совпадать. Например, классификатор модулей из гл. 1 можно обучить логистической регрессией, то есть минимизацией логистической потери, а оценивать по доле дефектных модулей, которые он нашёл; ту же логистическую регрессию в медицинской диагностике оценивают по доле пропущенных больных. Доля ошибок, наоборот, плохо подходит для обучения (гл. 7), но понятна как метрика. Для регрессии MSE часто служит и тем и другим.

Метрики модели — не единственный способ оценить её пользу. Модель работает внутри системы — например, системы, которая подсказывает тестировщикам, какие модули проверить первыми, — и пользу от неё измеряют на нескольких уровнях, от цели всей системы до свойств самой модели. Такую лестницу называют *иерархией метрик*. На верхнем уровне — цель бизнеса: меньше ошибок, обнаруженных пользователями, меньше затрат на тестирование. Её измеряют месяцами, и на неё влияет многое помимо модели. Ниже — *онлайн-метрики*, которые собирают с работающей системы: доля модулей, проверенных по подсказке модели, в которых действительно нашли ошибку. Ещё ниже — *офлайн-метрики*, вычисляемые до внедрения на исторических данных или по оценкам экспертов; к ним относятся все метрики этой главы. Хорошая офлайн-метрика должна быть согласована с верхними уровнями: если она растёт, а онлайн-метрики нет, метрика выбрана неудачно.

8.2. Матрица ошибок

В бинарной классификации класс $+1$ называют *положительным* — обычно это то, что нужно найти (модуль с ошибкой, болезнь, мошенническая операция), класс -1 — *отрицательным*. На каждом объекте x_i контрольной выборки сравниваются два значения: правильный ответ y_i и прогноз модели $\hat{y}_i = g(x_i, \theta)$. Каждое принимает два значения, поэтому возможны четыре исхода, и все объекты выборки распределяются по ним. Число объектов с каждым исходом записывают в таблицу 2×2 — *матрицу ошибок* (confusion matrix, табл. 8.1).

Таблица 8.1. Матрица ошибок: строка — правильный ответ, столбец — прогноз модели

	Прогноз $\hat{y} = +1$	Прогноз $\hat{y} = -1$
Ответ $y = +1$	TP — true positive, истинно положительный	FN — false negative, ложноотрицательный (пропуск)
Ответ $y = -1$	FP — false positive, ложноположительный (ложная тревога)	TN — true negative, истинно отрицательный

Обозначения читаются так: вторая буква — прогноз модели (P — positive, модель ответила $+1$; N — negative, ответила -1), первая — верен ли прогноз (T — true, верен; F — false, ошибочен). Например, FN — модель ответила -1 и ошиблась: объект в действительности положительный, модель его пропустила. FP — модель ответила $+1$ и ошиблась: подняла ложную тревогу на отрицательном объекте. В клетке таблицы стоит число таких объектов, например

$$\mathrm{TP} = \sum_{i=1}^{\ell} [y_i = +1]\,[\hat{y}_i = +1];$$

остальные три числа записываются так же. На диагонали таблицы (TP и TN) — верные ответы, вне диагонали (FN и FP) — ошибки двух видов. FP называют также ошибкой первого рода, FN — ошибкой второго рода. Каждый объект попадает ровно в одну клетку, поэтому $\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN} = \ell$.

Простейшая метрика — *доля правильных ответов* (accuracy): число верных прогнозов, делённое на число объектов. Её записывают так же, как эмпирический риск с индикаторной функцией потерь в гл. 1, только

индикатор отмечает не ошибки, а верные ответы; через клетки матрицы ошибок она равна отношению суммы диагональных клеток к сумме всех:

$$\text{Accuracy} = \frac{1}{\ell}\sum_{i=1}^{\ell}[\hat{y}_i = y_i] = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

Сумма индикаторов в числителе — число верных ответов, то есть сумма диагональных клеток TP + TN; знаменатель — объём выборки ℓ. Доля ошибок — это и есть эмпирический риск Q, который минимизировали в гл. 1, поэтому Accuracy $= 1 - Q$. Её недостаток проявляется при *несбалансированных классах*. Если модули с ошибками составляют 1 % всех модулей, классификатор, который всегда отвечает «ошибок нет», имеет долю правильных ответов 99 % и при этом бесполезен: он не находит ни одного дефектного модуля. Кроме того, доля правильных ответов считает все ошибки одинаковыми, хотя пропуск дефектного модуля и лишняя проверка исправного имеют разную цену: первый означает ошибку, которую найдут пользователи, вторая — потраченное время тестировщика.

Пример. Пусть есть десять программных модулей, для которых модель (например, логистическая регрессия) вычислила вероятность наличия в каждом из них ошибки $\hat{p}$ — здесь, как и в гл. 7, «крышка» над буквой означает прогноз модели (табл. 8.2). Модули упорядочены по убыванию $\hat{p}$; ошибки действительно нашли в четырёх из них.

Таблица 8.2. Оценки модели и истинные ответы для десяти модулей

Модуль	1	2	3	4	5	6	7	8	9	10
$\hat{p}$	0,95	0,85	0,70	0,60	0,45	0,40	0,30	0,20	0,15	0,05
y	+1	+1	−1	+1	−1	+1	−1	−1	−1	−1

При пороге 0,5 модель относит к классу +1 модули 1–4. Из них три действительно дефектные: TP = 3, FP = 1 (модуль 3). Дефектный модуль 6 пропущен: FN = 1. Остальные пять модулей верно признаны исправными: TN = 5. Матрица ошибок для этого порога приведена в табл. 8.3.

Таблица 8.3. Матрица ошибок для примера при пороге 0,5

	Прогноз $\hat{y} = +1$	Прогноз $\hat{y} = -1$
Ответ $y = +1$	TP = 3 (модули 1, 2, 4)	FN = 1 (модуль 6)

	Прогноз $\hat{y} = +1$	Прогноз $\hat{y} = -1$
Ответ $y = -1$	FP = 1 (модуль 3)	TN = 5 (модули 5, 7–10)

Доля правильных ответов — $(3 + 5)/10 = 0{,}8$. Для сравнения: классификатор «всегда -1» даёт $6/10 = 0{,}6$, хотя не находит ничего.

8.3. Точность и полнота

Доля правильных ответов смешивает ошибки обоих видов, а точность и полнота рассматривают их по отдельности. *Точность* (precision) — доля действительно положительных объектов среди тех, которые модель назвала положительными: она показывает, насколько можно доверять ответу $+1$. *Полнота* (recall) — доля положительных объектов, которые модель нашла:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

В примере при пороге 0,5 точность равна $3/4 = 0{,}75$: из четырёх модулей, отправленных на проверку, ошибки есть в трёх. Полнота тоже $3/4$: из четырёх дефектных модулей найдены три. Классификатор «всегда -1» имеет нулевую полноту: доля правильных ответов у него 0,6, а полнота сразу показывает, что он ничего не находит.

Точность и полнота обычно конкурируют. Если отправлять на проверку больше модулей (снизить порог), полнота растёт, а точность, как правило, падает: в список попадает больше исправных модулей. Чтобы сравнивать классификаторы одним числом, используют *F-меру* — среднее гармоническое точности и полноты:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \qquad F_\beta = \frac{(1 + \beta^2) \cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}.$$

Читаем формулу. Среднее гармоническое близко к меньшему из чисел: при точности 1 и полноте 0,1 среднее арифметическое равно 0,55, а $F_1 \approx 0{,}18$. Поэтому высокая F-мера требует, чтобы и точность, и полнота были высоки. Параметр β задаёт относительную важность: при $\beta > 1$ важнее полнота (F_2 — когда пропуск дороже ложной тревоги), при $\beta < 1$ — точность.

В медицине и технической диагностике используют другие названия. *Чувствительность* — то же, что полнота: доля положительных объектов, которые модель отнесла к положительному классу. *Специфичность*

— доля отрицательных объектов, которые модель отнесла к отрицательному классу, $\mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$; в примере $5/6 \approx 0{,}83$.

Несколько классов. Если классов $K > 2$, точность и полноту вычисляют для каждого класса k, считая его положительным, а остальные — отрицательными. Так для каждого класса $k = 1, \dots, K$ получаются свои числа TP_k, FP_k и FN_k, которые затем сводят в одно значение метрики одним из двух способов. При *макроусреднении* метрику вычисляют для каждого класса и берут обычное среднее:

$$\mathrm{Precision}_{\text{макро}} = \frac{1}{K}\sum_{k=1}^{K}\frac{\mathrm{TP}_k}{\mathrm{TP}_k+\mathrm{FP}_k}, \qquad \mathrm{Recall}_{\text{макро}} = \frac{1}{K}\sum_{k=1}^{K}\frac{\mathrm{TP}_k}{\mathrm{TP}_k+\mathrm{FN}_k}.$$

Все классы весят одинаково, поэтому плохое качество на редком классе существенно снижает итоговую метрику. При *микроусреднении* сначала суммируют числа по всем классам, а затем вычисляют метрику по суммам:

$$\mathrm{Precision}_{\text{микро}} = \frac{\sum_{k=1}^{K}\mathrm{TP}_k}{\sum_{k=1}^{K}(\mathrm{TP}_k+\mathrm{FP}_k)}, \qquad \mathrm{Recall}_{\text{микро}} = \frac{\sum_{k=1}^{K}\mathrm{TP}_k}{\sum_{k=1}^{K}(\mathrm{TP}_k+\mathrm{FN}_k)}.$$

Здесь одинаково весит каждый объект, результат определяют частые классы, а ошибки на малочисленных классах на итоговую метрику почти не влияют. Если каждому объекту приписан ровно один класс, микроусреднённые точность и полнота совпадают с долей правильных ответов. Каждая ошибка — одновременно FP для класса, который назвала модель, и FN для правильного класса, поэтому $\sum_k \mathrm{FP}_k = \sum_k \mathrm{FN}_k$ равно числу ошибок, а $\sum_k \mathrm{TP}_k$ — числу верных ответов.

8.4. Порог, ROC-кривая и AUC

Модель выдаёт вероятность, а не класс: чтобы получить класс, $\hat{p}$ сравнивают с порогом и относят объект к классу $+1$, если $\hat{p}$ не меньше порога. Порог 0,5 выбирают по умолчанию, но он оптимален, только если ошибки первого и второго рода одинаково дороги. Пусть пропуск дефектного модуля стоит c_+, а лишняя проверка исправного — c_-. Если отнести модуль к классу $+1$, ожидаемые потери равны $(1-\hat{p})\,c_-$ (модуль окажется исправным с вероятностью $1-\hat{p}$); если к классу -1, они равны $\hat{p}\,c_+$ (модуль окажется дефектным с вероятностью $\hat{p}$). Отвечать $+1$ не хуже, чем

-1, когда $(1-\hat{p})\,c_- \le \hat{p}\,c_+$, то есть при

$$\hat{p} \ge \frac{c_-}{c_- + c_+}.$$

Если пропуск вчетверо дороже лишней проверки, порог равен $1/(1+4) = 0{,}2$.

Вернёмся к десяти модулям из табл. 8.2 и снизим порог с 0,5 до 0,35. Значение 0,35 взято для примера: оценки модулей 6 и 7 равны 0,40 и 0,30, и любой порог больше 0,30 и не больше 0,40 даёт тот же результат. Теперь на проверку уходят модули 1–6: TP = 4, FP = 2, FN = 0, TN = 4. Полнота выросла до 1, точность упала до $4/6 \approx 0{,}67$, а F_1 выросла с 0,75 до 0,8. Каждому порогу соответствует свой классификатор. Чтобы оценить модель независимо от порога, рассматривают все пороги сразу.

ROC-кривая (receiver operating characteristic) — кривая на плоскости с осями

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \qquad \mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

TPR — доля положительных объектов, которые модель нашла (полнота), FPR — доля отрицательных объектов, на которых модель подняла ложную тревогу (единица минус специфичность). Каждая точка кривой соответствует одному порогу. При пороге выше всех оценок классификатор ничего не называет положительным, это точка (0; 0); при пороге ниже всех оценок он всё называет положительным — точка (1; 1).

Построим ROC-кривую для примера из табл. 8.2 (рис. 8.1, а). Будем снижать порог так, чтобы к классу $+1$ добавлялось по одному модулю в порядке убывания $\hat{p}$. Каждый положительный модуль поднимает кривую на 1/4 (всего положительных четыре), каждый отрицательный сдвигает её вправо на 1/6. Модули 1 и 2 положительные — два шага вверх; модуль 3 отрицательный — шаг вправо; модуль 4 — вверх; модуль 5 — вправо; модуль 6 — вверх, и полнота достигла единицы; оставшиеся четыре модуля — шаги вправо. Порогам 0,5 и 0,35 соответствуют точки (1/6; 0,75) и (1/3; 1).

Идеальная модель даёт всем положительным объектам оценки выше, чем отрицательным; её кривая сразу поднимается до TPR = 1, а затем идёт вправо. Модель, выдающая случайные оценки, в среднем даёт диагональ TPR = FPR. Мерой качества служит *площадь под ROC-кривой* — AUC (area under curve): 1 у идеальной модели, около 0,5 у случайной. В примере AUC = 0,875.

Вероятностный смысл AUC. AUC равна вероятности того, что случайно выбранный положительный объект получит от модели оценку

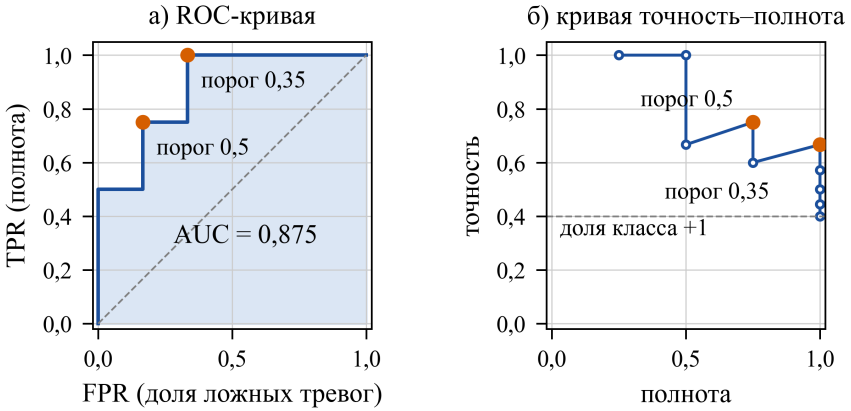


Рис. 8.1. Кривые для примера из табл. 8.2: а — ROC-кривая; б — кривая точность–полнота

выше, чем случайно выбранный отрицательный. Проверим на примере. Пар «положительный — отрицательный» $4 \cdot 6 = 24$. Модули 1 и 2 выше всех шести отрицательных — 12 верно упорядоченных пар; модуль 4 выше пяти отрицательных (всех, кроме модуля 3) — ещё 5; модуль 6 выше четырёх (модулей 7–10) — ещё 4. Итого $21/24 = 0{,}875$ — та же площадь под кривой. Почему так получается, видно, если разрезать площадь под ROC-кривой на горизонтальные полосы — по одной на каждый шаг кривой вверх, то есть на каждый положительный объект. Высота полосы 1/4, а тянется она от кривой до правого края квадрата. Её ширина равна единице минус FPR в момент шага, то есть доле отрицательных объектов, которые ещё не добавлены, — получили оценку ниже этого положительного. Для модулей 1 и 2 ширина 6/6, для модуля 4 она равна 5/6, для модуля 6 — 4/6, и площадь равна $\frac{1}{4}\left(1 + 1 + \frac{5}{6} + \frac{4}{6}\right) = \frac{21}{24}$. В общем случае каждая полоса — доля пар, в которых данный положительный объект стоит выше отрицательного, а сумма полос — доля всех верно упорядоченных пар.

Из этого смысла следуют свойства AUC. Она зависит только от порядка оценок, а не от их величины: если все вероятности возвести в квадрат, AUC не изменится. Она не зависит от порога и от соотношения классов. Поэтому AUC удобна для сравнения моделей, но ничего не говорит о том, хорошо ли откалиброваны вероятности и какой порог выбрать.

8.5. Кривая точность–полнота и логарифмическая потеря

При сильном дисбалансе классов ROC-кривая бывает обманчиво хорошей. Если исправных модулей тысячи, а дефектных десятки, даже сотня ложных тревог даёт малую FPR, и кривая выглядит почти идеальной, хотя на каждый найденный дефектный модуль приходится несколько лишних проверок. *Кривая точность–полнота* (рис. 8.1, б) показывает, сколько лишних проверок приходится на найденные дефекты: по горизонтали полнота, по вертикали точность при каждом пороге. Случайной модели соответствует горизонталь на уровне доли положительного класса (в примере 0,4). При редком классе эта горизонталь проходит низко, а кривая хорошей модели лежит намного выше неё. Площадь под кривой точность–полнота на практике оценивают *средней точностью* (average precision, в scikit-learn — `average_precision_score`): точность вычисляют в тот момент, когда в список отобранных попадает очередной положительный объект, и усредняют по всем положительным объектам. В примере положительные модули 1, 2, 4 и 6 попадают в список при точности 1; 1; 3/4 и 4/6, и средняя точность равна $(1+1+0{,}75+0{,}67)/4 \approx 0{,}85$.

Если важны сами вероятности — например, по ним рассчитывают ожидаемые затраты, — качество измеряют *логарифмической потерей* (log-loss), средней бинарной кросс-энтропией из гл. 7:

$$\mathrm{LogLoss} = -\frac{1}{\ell}\sum_{i=1}^{\ell}\bigl[\tilde{y}_i \ln \hat{p}_i + (1-\tilde{y}_i)\ln(1-\hat{p}_i)\bigr],$$

где $\tilde{y}_i \in \{0, 1\}$. В отличие от AUC, она чувствительна к величине вероятностей: уверенная ошибка стоит дорого. В примере log-loss равна 0,42, причём наибольший вклад, 1,20, даёт модуль 3: модель уверенно (с вероятностью 0,7) сочла его дефектным, а ошибок в нём нет.

8.6. Выбор метрики

Метрика выбирается до построения модели, исходя из задачи. Полезно ответить на вопросы:

- Что дороже: пропуск положительного объекта или ложная тревога? Если пропуск (дефект дойдёт до пользователей, болезнь не будет выявлена) — важна полнота, F_2 или порог, сдвинутый вниз. Если ложная тревога (рекомендация, которая раздражает пользователя) — важна точность.
- Сбалансированы ли классы? При дисбалансе доля правильных ответов неинформативна; используют точность, полноту, F-меру,

кривую точность–полнота.

- Нужен ли порог сразу? Для сравнения моделей до выбора порога подходит AUC.
- Используются ли вероятности дальше? Тогда нужна log-loss.

Значение метрики само по себе мало что говорит. Его сравнивают с *базовой моделью* (гл. 5): для классификации это предсказание самого частого класса, случайное предсказание с учётом долей классов или простое правило по одному признаку («модуль подозрителен, если в нём больше тысячи строк»). Модель, которая не превосходит такое правило, не стоит внедрения. Все метрики главы реализованы в модуле `sklearn.metrics` (листинг 8.1).

Листинг 8.1. Метрики качества классификации в scikit-learn

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import log_loss, roc_auc_score

p = model.predict_proba(X_test)[:, 1]   # P(класс 1)
y_pred = (p >= 0.35).astype(int)        # свой порог
print(confusion_matrix(y_test, y_pred))
# точность, полнота и F1 по каждому классу
print(classification_report(y_test, y_pred))
print(roc_auc_score(y_test, p), log_loss(y_test, p))
```

Метрики, не зависящие от порога (AUC, log-loss), вычисляют по вероятностям `p`, а метрики, зависящие от него, — по классам `y_pred`.

Вопросы для самоконтроля

1. Чем метрика качества отличается от функции потерь?
2. Почему доля правильных ответов обманчива при несбалансированных классах?
3. Что показывают точность и полнота? Почему они конкурируют?
4. Почему F-мера использует среднее гармоническое, а не арифметическое?
5. Как выбрать порог, если пропуск положительного объекта в 9 раз дороже ложной тревоги?
6. Что откладывают по осям ROC-кривой? Какие точки соответствуют самому высокому и самому низкому порогу?
7. Каков вероятностный смысл AUC? Изменится ли AUC, если все оценки модели умножить на 2?
8. Когда кривая точность–полнота информативнее ROC-кривой?

Задачи

1. Классификатор на 1000 объектах дал TP = 40, FP = 60, FN = 10, TN = 890. Найдите долю правильных ответов, точность, полноту, F_1 и специфичность.
2. Для примера из табл. 8.2 найдите точность, полноту и F_1 при пороге 0,65.
3. Модель выдала оценки 0,9; 0,8; 0,3 положительным объектам и 0,7; 0,2 — отрицательным. Вычислите AUC подсчётом пар.
4. Ложная тревога стоит 1 ч работы тестировщика, пропуск дефекта — 20 ч на исправление после выпуска. Какой порог вероятности выбрать?

9. Наивный байесовский классификатор

Логистическая регрессия оценивает вероятность класса по признакам объекта напрямую. Есть и другой подход: описать, как выглядят объекты каждого класса, — какие слова встречаются в спаме и какие в обычных письмах, — а затем по формуле Байеса пересчитать это описание в вероятность класса для нового объекта. Наивный байесовский классификатор — простейшая из моделей, построенных по этому принципу. Он обучается подсчётом частот, работает при огромном числе признаков и на малых выборках и служит стандартной базовой моделью для классификации текстов.

9.1. Формула Байеса

Условная вероятность $P(A \mid B)$ — вероятность события A при условии, что событие B произошло. Формула Байеса связывает вероятность класса при известном объекте с вероятностью объекта при известном классе:

$$P(y \mid x) = \frac{P(y)\, p(x \mid y)}{p(x)}, \qquad p(x) = \sum_{c \in Y} P(c)\, p(x \mid c). \tag{9.1}$$

Здесь $P(y)$ — *априорная вероятность* класса, известная до того, как мы увидели объект; $p(x \mid y)$ — *правдоподобие*, вероятность (для непрерывных признаков — плотность) встретить такой объект среди объектов класса y; $P(y \mid x)$ — *апостериорная вероятность* класса после наблюдения объекта. Знаменатель $p(x)$ одинаков для всех классов и лишь делает сумму апостериорных вероятностей равной единице.

Пример. Покажем, как формула Байеса связывает частоту класса и частоту признака. Пусть спам составляет 30 % писем, слово «выигрыш» встречается в 40 % спама и в 2 % обычных писем. Какова вероятность, что письмо со словом «выигрыш» — спам? По формуле (9.1)

$$P(\text{спам} \mid \text{«выигрыш»}) = \frac{0{,}3 \cdot 0{,}4}{0{,}3 \cdot 0{,}4 + 0{,}7 \cdot 0{,}02} = \frac{0{,}12}{0{,}134} \approx 0{,}90.$$

Слово встречается меньше чем в половине спам-писем, но сдвигает вероятность с 0,3 до 0,9: ключевое значение имеет отношение правдоподобий $0{,}4/0{,}02 = 20$, а не сама частота встречаемости 0,4.

Два подхода к классификации. Логистическая регрессия (гл. 7) моделирует апостериорную вероятность $P(y \mid x)$ напрямую; такие модели называют *дискриминативными* — они учатся отличать классы друг от

друга. *Генеративные* модели описывают, как устроены объекты каждого класса: оценивают $P(y)$ и $p(x \mid y)$, а $P(y \mid x)$ получают по формуле Байеса. Генеративная модель знает о данных больше, чем нужно для классификации (из неё можно порождать новые объекты), но платит за это допущениями о виде $p(x \mid y)$.

9.2. Оптимальный байесовский классификатор

Предположим, что распределения $P(y)$ и $p(x \mid y)$ известны точно. Если отнести объект x к классу y, классификатор ошибётся с вероятностью $1 - P(y \mid x)$. Значит, вероятность ошибки наименьшая, если выбирать класс с наибольшей апостериорной вероятностью. Знаменатель $p(x)$ в формуле (9.1) от класса не зависит, и его можно отбросить:

$$g(x) = \arg\max_{y \in Y} P(y)\, p(x \mid y). \tag{9.2}$$

Это *оптимальный байесовский классификатор*. Если ошибки первого и второго рода (гл. 8) имеют разную цену, выбирают класс с наименьшими ожидаемыми потерями; для двух классов это даёт порог $c_-/(c_- + c_+)$ на апостериорную вероятность класса $+1$ — тот же, что в гл. 8.

Его ошибка — наименьшая возможная при данных признаках. Она больше нуля, если объекты разных классов бывают одинаковыми по признакам (два письма с одним набором слов, одно — спам, другое нет); такую ошибку не устранит никакое обучение, нужны другие признаки.

На практике распределения неизвестны, и их оценивают по выборке. Вероятность класса оценить легко: $P(y) \approx \ell_y/\ell$, где ℓ_y — число объектов класса y в выборке. С правдоподобием $p(x \mid y)$ дело обстоит сложнее. Пусть у объекта n бинарных признаков, например наличие или отсутствие n слов в письме. Различных объектов 2^n, и для каждого класса нужно оценить вероятность каждого из них. При $n = 30$ это около миллиарда чисел; большинство сочетаний слов в выборке не встретится ни разу, и их вероятность оценить невозможно.

9.3. Наивное предположение

Наивный байесовский классификатор упрощает задачу допущением: *при известном классе признаки независимы*. Тогда правдоподобие объекта распадается в произведение правдоподобий отдельных признаков:

$$p(x \mid y) = \prod_{j=1}^{n} p\big(f_j(x) \mid y\big).$$

Вместо распределения на 2^n сочетаниях в каждом классе нужно оценить n одномерных распределений (для бинарных признаков — n чисел), и каждое оценивается по всем объектам класса.

Произведение многих вероятностей становится меньше наименьшего числа, представимого в компьютере: для 1000 слов с вероятностями около 0,001 оно порядка 10^{-3000}, а числа двойной точности не бывают меньше примерно 10^{-324}. Поэтому, как и в гл. 7, переходят к логарифмам:

$$g(x) = \arg\max_{y \in Y} \Big(\ln P(y) + \sum_{j=1}^{n} \ln p\big(f_j(x) \mid y\big) \Big). \tag{9.3}$$

Читаем формулу. Каждый класс набирает очки: начальный запас — логарифм его априорной вероятности, затем каждый признак вносит свой вклад $\ln p(f_j(x) \mid y)$, больший у того класса, для которого такое значение признака типично. Побеждает класс с наибольшей суммой.

Почему «наивный». Допущение почти всегда неверно: слова «выигрыш» и «бесплатно» в спаме встречаются вместе, рост и вес человека связаны, а значит допущение о независимости признаков не выполняется. Тем не менее классификатор часто работает хорошо. Для верного ответа не нужно, чтобы оценки вероятностей были точными, — нужно только, чтобы у правильного класса сумма в (9.3) была наибольшей, а это выполняется и при заметной зависимости признаков [9]. За это допущение приходится платить точностью самих вероятностей: информацию, общую для нескольких признаков, классификатор учитывает несколько раз, и его оценки получаются слишком близкими к 0 или 1.

9.4. Гауссовский наивный байес

Для непрерывного признака нужно выбрать вид распределения $p(f_j(x) \mid y)$. Обычно берут нормальное:

$$p\big(f_j(x) \mid y\big) = \frac{1}{\sqrt{2\pi}\,\sigma_{jy}} \exp\!\left(-\frac{\big(f_j(x) - \mu_{jy}\big)^2}{2\sigma_{jy}^2} \right),$$

где μ_{jy} и σ_{jy}^2 — среднее и дисперсия признака f_j в классе y. Обучение сводится к их подсчёту по объектам класса y (это оценки максимального правдоподобия из гл. 7).

Пример. Покажем, как выглядит граница между классами. Пусть признак один, класс -1 составляет 60 % выборки, и у объектов этого класса значения признака распределены нормально со средним 2 и

стандартным отклонением 1, а у объектов класса +1 (40 % выборки) — со средним 5 и стандартным отклонением 2. На рис. 9.1, а показаны произведения $P(y)\,p(x \mid y)$ — величины, которые сравнивает правило (9.2). Для объекта $x = 3$ они равны $0{,}6 \cdot 0{,}242 = 0{,}145$ и $0{,}4 \cdot 0{,}121 = 0{,}048$, и апостериорная вероятность класса +1 равна $0{,}048/(0{,}145 + 0{,}048) \approx 0{,}25$. Кривые пересекаются при $x \approx 3{,}63$: правее классификатор выбирает класс +1.

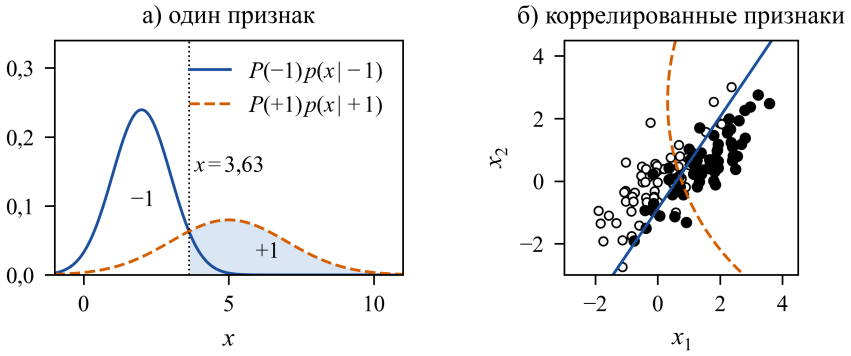


Рис. 9.1. Гауссовский наивный байес: а — произведения $P(y)\,p(x \mid y)$ для одного признака, класс +1 выбирается в закрашенной области; б — граница наивного байеса (штриховая) и логистической регрессии (сплошная) при коррелированных признаках

Граница проходит не посередине между средними (3,5): сказываются и априорные вероятности, и дисперсии. Есть и вторая точка пересечения, $x \approx -1{,}63$, где обе кривые почти нулевые: там снова выигрывает класс +1, его распределение шире и на краях убывает медленнее. Причина в том, что логарифм нормальной плотности квадратичен по признаку, и граница между классами квадратичная. Если дисперсии признака в обоих классах одинаковы, слагаемые с f_j^2 при сравнении классов сокращаются, и граница становится линейной.

9.5. Тексты и мешок слов

Чтобы классифицировать тексты, их нужно превратить в признаки. Простейший способ — *мешок слов* (bag of words, BoW): составляют словарь из n слов, встречающихся в обучающих текстах, и признаком $f_j(x)$ служит число вхождений j-го слова в текст x. Порядок слов теряется — отсюда название. Классификация текстов наивным байесом подробно разобра-

на в [1].

Мультиномиальный наивный байес считает, что текст класса y порождается так: слова выбираются независимо одно за другим, и слово j выбирается с вероятностью q_{jy}. Тогда правдоподобие текста с точностью до множителя, одинакового для всех классов, равно

$$p(x \mid y) \propto \prod_{j=1}^{n} q_{jy}^{f_j(x)} :$$

каждое вхождение слова умножает правдоподобие на его вероятность. Оценка вероятности слова — его доля среди всех слов текстов класса:

$$q_{jy} = \frac{N_{jy}}{N_y}, \tag{9.4}$$

где N_{jy} — сколько раз слово j встретилось в обучающих текстах класса y, а $N_y = \sum_j N_{jy}$ — общее число слов в них.

Бернуллиевский вариант учитывает только наличие слова, $f_j(x) \in \{0, 1\}$, и оценивает q_{jy} как долю текстов класса y, содержащих слово j. Правдоподобие $\prod_j q_{jy}^{f_j(x)}(1 - q_{jy})^{1-f_j(x)}$ учитывает и отсутствие слов: если в письме нет слова, типичного для спама, это довод против спама.

Пример: спам-фильтр. Покажем на примере, как по обучающей выборке вручную вычисляется вероятность класса, и заодно обнаружим проблему нулевых частот. Словарь — четыре слова, обучающая выборка — шесть писем (табл. 9.1), спам — класс +1.

Таблица 9.1. Обучающая выборка спам-фильтра

Письмо	Текст	Класс
1	выигрыш выигрыш бесплатно	+1
2	бесплатно выигрыш	+1
3	бесплатно встреча	+1
4	отчёт встреча	−1
5	отчёт отчёт бесплатно	−1
6	встреча отчёт	−1

$P(+1) = P(-1) = 0{,}5$. В спаме семь слов: «выигрыш» — 3 раза, «бесплатно» — 3, «отчёт» — 0, «встреча» — 1. В обычных письмах тоже семь: 0, 1, 4 и 2 раза соответственно. По формуле (9.4) вероятности слов в спаме — 3/7, 3/7, 0, 1/7, в обычных письмах — 0, 1/7, 4/7, 2/7.

Классифицируем письмо «выигрыш встреча». Для спама произведе-

ние $0{,}5 \cdot \frac{3}{7} \cdot \frac{1}{7}$, для обычного письма $0{,}5 \cdot 0 \cdot \frac{2}{7} = 0$: слово «выигрыш» ни разу не встретилось в обычных письмах, и вероятность спама получается равной 1. Письмо «бесплатно отчёт», наоборот, получает вероятность спама 0, потому что слова «отчёт» не было в спаме, — хотя «бесплатно» встречается в спаме втрое чаще. Слово, не встретившееся в обучающих текстах класса, перечёркивает вклад всех остальных слов: в сумме (9.3) его слагаемое равно $\ln 0 = -\infty$. Это *проблема нулевых частот*.

9.6. Сглаживание Лапласа

Нулевая оценка — следствие малой выборки, а не свойство класса. *Сглаживание Лапласа* прибавляет к каждому счётчику число $\alpha > 0$, как будто каждое слово словаря встретилось в каждом классе ещё α раз:

$$q_{jy} = \frac{N_{jy} + \alpha}{N_y + \alpha n}.$$

В знаменателе добавлено αn, чтобы вероятности всех n слов по-прежнему давали в сумме единицу. При $\alpha = 1$ сглаживание называют лапласовским, при $0 < \alpha < 1$ — сглаживанием Лидстона. На больших выборках добавка мало меняет частоты частых слов, а редким словам не даёт нулевых оценок.

Применим сглаживание с $\alpha = 1$ к тому же примеру ($n = 4$). Знаменатель в обоих классах $7 + 4 = 11$; вероятности слов «выигрыш», «бесплатно», «отчёт», «встреча» в спаме — 4/11, 4/11, 1/11, 2/11, в обычных письмах — 1/11, 2/11, 5/11, 3/11. Для письма «выигрыш встреча» произведения равны $0{,}5 \cdot \frac{4}{11} \cdot \frac{2}{11}$ и $0{,}5 \cdot \frac{1}{11} \cdot \frac{3}{11}$, они относятся как $8 : 3$. Вероятность спама — доля первого произведения в их сумме, то есть $8/(8+3) = 8/11 \approx 0{,}73$; совпадение со знаменателем 11 в вероятностях слов случайно. Для письма «бесплатно отчёт» произведения равны $0{,}5 \cdot \frac{4}{11} \cdot \frac{1}{11}$ и $0{,}5 \cdot \frac{2}{11} \cdot \frac{5}{11}$, они относятся как $4 : 10$, и вероятность спама $4/(4+10) = 4/14 \approx 0{,}29$. Вместо категоричных 1 и 0 получились осторожные оценки, учитывающие оба слова. Гиперпараметр α по роли похож на коэффициент регуляризации (гл. 6): чем он больше, тем сильнее оценки стянуты к равномерным, — и подбирают его так же, кросс-валидацией.

Пример: бернуллиевский вариант. Покажем, как на ответ влияет отсутствие слов. Теперь q_{jy} — доля писем класса y, содержащих слово j. Сглаживание прибавляет α к числу таких писем и 2α к числу писем класса (у признака два значения): $q_{jy} = (D_{jy} + \alpha)/(\ell_y + 2\alpha)$, где D_{jy} — число писем класса y со словом j. В табл. 9.1 «выигрыш» есть в двух письмах

спама из трёх, «бесплатно» — в трёх, «отчёт» — ни в одном, «встреча» — в одном; в обычных письмах — в 0, 1, 3 и 2. При $\alpha = 1$ вероятности слов в спаме — 3/5, 4/5, 1/5, 2/5, в обычных письмах — 1/5, 2/5, 4/5, 3/5.

Классифицируем снова письмо «выигрыш встреча». В шансы спама входят все четыре слова словаря: присутствующее — множителем $q_{j,+1}/q_{j,-1}$, отсутствующее — множителем $(1 - q_{j,+1})/(1 - q_{j,-1})$. «Выигрыш» есть: 3; «встреча» есть: 2/3; «бесплатно» нет: $(1/5)/(3/5) = 1/3$ — довод против спама, ведь в спаме это слово есть всегда; «отчёт» нет: $(4/5)/(1/5) = 4$ — довод за спам. Шансы спама $3 \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot 4 = \frac{8}{3}$, вероятность $8/11 \approx 0{,}73$ — здесь случайно та же, что у мультиномиального варианта. Для письма из одного слова «встреча» варианты расходятся. Мультиномиальный учитывает только это слово и даёт шансы $2 : 3$, вероятность спама 0,40. Бернуллиевский добавляет вклад трёх отсутствующих слов: шансы $\frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{3} \cdot 4 = \frac{4}{9}$, вероятность $4/13 \approx 0{,}31$.

Таким образом, бернуллиевский вариант оценивает письмо и по тому, каких слов в нём нет. При небольшом словаре это помогает, а при словаре из десятков тысяч слов отсутствие отдельного слова почти ничего не говорит, и мультиномиальный вариант обычно точнее [10].

9.7. Наивный байес и логистическая регрессия

Возьмём для мультиномиального наивного байеса логарифм отношения апостериорных вероятностей двух классов — логит из гл. 7. По формуле Байеса

$$\ln \frac{P(+1 \mid x)}{P(-1 \mid x)} = \ln \frac{P(+1)}{P(-1)} + \sum_{j=1}^{n} f_j(x) \ln \frac{q_{j,+1}}{q_{j,-1}}. \tag{9.5}$$

Читаем формулу. Справа — линейная функция признаков: свободный член $\theta_0 = \ln(P(+1)/P(-1))$ и веса $\theta_j = \ln(q_{j,+1}/q_{j,-1})$. Значит, $P(+1 \mid x) = \sigma(h(x, \theta))$ — модель той же формы, что логистическая регрессия. В примере веса слов «выигрыш», «бесплатно», «отчёт», «встреча» равны $\ln 4 = 1{,}39$; $\ln 2 = 0{,}69$; $\ln 0{,}2 = -1{,}61$; $\ln(2/3) = -0{,}41$, а $\theta_0 = 0$. Для текста «выигрыш встреча» $h = 1{,}39 - 0{,}41 = 0{,}98$ и $\sigma(0{,}98) \approx 0{,}73$ — тот же ответ. Логит линеен по признакам и в бернуллиевском варианте, и в гауссовском — если дисперсия каждого признака одинакова в обоих классах (иначе, как показано выше, граница квадратичная).

Различаются модели способом нахождения весов. Наивный байес оценивает вес каждого признака отдельно, по частотам в классах. Логистическая регрессия подбирает все веса совместно, максимизируя

правдоподобие ответов, и учитывает зависимость признаков. Поэтому наивный байес учитывает одну и ту же информацию дважды: если бы у слова «выигрыш» в словаре был двойник, всегда стоящий рядом с ним (например, если бы в спаме всегда писали «крупный выигрыш», а слово «крупный» отдельно не встречалось), наивный байес сложил бы оба вклада, и вероятность спама для письма «выигрыш встреча» выросла бы с 0,73 примерно до 0,91, хотя информации не прибавилось; логистическая регрессия разделила бы вес между двойниками.

На рис. 9.1, б признаки x_1 и x_2 сильно коррелированы в обоих классах (коэффициент корреляции 0,8), а классы различаются тем, что у класса $+1$ (чёрные точки) x_1 больше при том же x_2. Наивный байес считает оба признака независимыми доводами в пользу класса $+1$ и проводит границу поперёк облака точек. Логистическая регрессия нашла веса 3,0 и $-2{,}0$: при фиксированном x_1 большое x_2 говорит против класса $+1$. На 100 000 новых объектов из тех же распределений доля правильных ответов наивного байеса 0,74, логистической регрессии — 0,83.

Зато наивный байес выигрывает при малых выборках: он приближается к своей наилучшей ошибке уже при числе объектов, растущем логарифмически с числом признаков, тогда как логистической регрессии нужно число объектов порядка числа признаков; при большой выборке логистическая регрессия обычно точнее [11].

Итак, достоинства наивного байеса: обучение — один проход по выборке с подсчётом частот, без оптимизации, и дообучение на новых данных — прибавление к счётчикам; работа на малых выборках и при десятках тысяч признаков; пропущенный признак просто не участвует в сумме (9.3). Недостатки: зависимость признаков не учитывается; для непрерывных признаков приходится угадывать вид распределения; вероятности излишне уверенные — по метрикам гл. 8 AUC, зависящая только от порядка оценок, бывает высокой, а log-loss плохой. Если нужен не только класс, но и правдоподобная оценка вероятности — например, чтобы рассчитать по ней ожидаемые затраты, как в гл. 8, — оценки *калибруют*: подправляют так, чтобы среди объектов с оценкой около 0,8 к классу $+1$ действительно относилось около 80 %. В scikit-learn для этого служит класс `CalibratedClassifierCV`.

В scikit-learn описанные в этой главе виды наивного байесовского классификатора реализованы в классах `GaussianNB`, `MultinomialNB` и `BernoulliNB` модуля `sklearn.naive_bayes`; параметр `alpha` — коэффициент сглаживания (по умолчанию 1). В листинге 9.1 — спам-фильтр из примера выше; мешок слов строится с помощью класса `CountVectorizer`.

Листинг 9.1. Мультиномиальный наивный байес для текстов

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

texts = ["выигрыш выигрыш бесплатно", "бесплатно выигрыш",
         "бесплатно встреча", "отчёт встреча",
         "отчёт отчёт бесплатно", "встреча отчёт"]
y = [1, 1, 1, 0, 0, 0]                          # 1 — спам
model = make_pipeline(CountVectorizer(),
                      MultinomialNB(alpha=1.0))
model.fit(texts, y)
model.predict_proba(["выигрыш встреча"])     # [[0.27, 0.73]]
```

Вопросы для самоконтроля

1. Что такое априорная и апостериорная вероятности и правдоподобие в формуле Байеса?
2. Чем генеративная модель отличается от дискриминативной? К какому типу относятся логистическая регрессия и наивный байес?
3. Почему ошибка оптимального байесовского классификатора может быть больше нуля?
4. В чём состоит наивное предположение? Зачем оно нужно?
5. Чем мультиномиальный вариант отличается от бернуллиевского?
6. Какую проблему решает сглаживание Лапласа?
7. Почему вероятности наивного байеса бывают излишне уверенными?

Задачи

1. Болезнь встречается у 1 % людей. Тест обнаруживает её у 90 % больных и ошибочно даёт положительный результат у 5 % здоровых. Какова вероятность болезни при положительном тесте?
2. В примере со спам-фильтром (с $\alpha = 1$) найдите вероятность спама для письма «бесплатно бесплатно встреча».
3. Признак распределён нормально с дисперсией 1 в обоих классах, со средним 0 в классе -1 и 4 в классе $+1$. Найдите границу при $P(-1) = 0{,}8$, $P(+1) = 0{,}2$.
4. Объекты описаны 20 бинарными признаками, классов два. Сколько вероятностей нужно оценить для правдоподобия без допущения о независимости и с ним?

Заключение

Линейная и логистическая регрессия и наивный байесовский классификатор внешне не похожи друг на друга, но обучаются по общей схеме. Выбирают семейство моделей $g(x, \theta)$ и функцию потерь $\mathcal{L}$, а параметры находят минимизацией эмпирического риска — явной формулой, как в методе наименьших квадратов, градиентным спуском, как в логистической регрессии, или подсчётом частот, как в наивном байесовском классификаторе. Малый риск на обучающей выборке ещё не означает хорошей модели: качество проверяют на объектах, не участвовавших в обучении, с помощью отложенной выборки или кросс-валидации, а переобучение сдерживают регуляризацией. Метрику качества выбирают исходя из задачи, и она не обязана совпадать с функцией потерь.

Теория вероятностей объясняет, откуда берутся функции потерь и штрафы, которые в главах книги могли показаться выбранными произвольно. Квадратичная потеря соответствует нормальному шуму в ответах, логистическая — распределению Бернулли для меток класса; минимизация риска с такими потерями — это максимизация правдоподобия. Штрафы L2 и L1 соответствуют априорным распределениям весов — нормальному и Лапласа. Дискриминативные модели оценивают вероятность класса напрямую, генеративные — через распределения объектов в классах и формулу Байеса.

На практике решение задачи машинного обучения начинается не с модели, а с постановки: что прогнозируется, какие данные доступны в момент прогноза, как будет измеряться качество и с какой базовой моделью сравнивать. Затем данные исследуют и готовят, разбивают так, чтобы оценка качества не была завышенной, обучают простую модель и только после этого — более сложные. Линейная и логистическая регрессия, наивный байесовский классификатор остаются хорошими базовыми моделями и в задачах, где в итоге используют более мощные методы.

Понятия книги переносятся на другие модели. Кусочно-линейная функция потерь от отступа приводит к методу опорных векторов, экспоненциальная — к бустингу; логистическая регрессия с функцией softmax — это нейронная сеть из одного слоя, а мини-пакетный градиентный спуск и регуляризация весов — основа обучения глубоких сетей. Для дальнейшего изучения рекомендуются книги [6, 7] — подробное и строгое изложение статистического обучения, [2, 3] — практическое руководство по scikit-learn и нейронным сетям, [4, 5] — инженерные приёмы построения систем машинного обучения.

Ответы к задачам

Глава 1. Задача обучения по прецедентам

1. Модель ошибается на x_3 и x_5, $Q = 2/5 = 0{,}4$.
2. $\theta = \sum x_i y_i / \sum x_i^2 = 46/30 \approx 1{,}53$ — почти модель Б.

Глава 2. Данные и признаки

1. Шесть строк вида (1, площадь, $f_{\text{Б}}$, $f_{\text{В}}$, этаж, год); у квартиры № 4 год равен 2005.
2. $m = 52$, $\sigma \approx 13{,}66$; стандартизованные площади квартир № 1–6: $-1{,}32$; 0; $-0{,}81$; 1,68; 0,81; $-0{,}37$.

Глава 3. Линейная регрессия и метод наименьших квадратов

1. $\theta_1 = 1{,}5$, $\theta_0 = 0{,}5$; остатки $-0{,}5$; 1; $-0{,}5$, их сумма равна нулю.
2. $\mathbf{F}^{\mathrm{T}}\mathbf{F} = \begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix}$, $\mathbf{F}^{\mathrm{T}}\mathbf{y} = (6;\ 9)^{\mathrm{T}}$, $\boldsymbol{\theta}^* = (0{,}5;\ 1{,}5)$.
3. Определитель $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ равен нулю. Псевдообратная матрица даёт решение наименьшей длины, в котором вес получают оба признака: $\theta_2 = 100\,\theta_1$.
4. Через $d + 1$ точку; $Q = 0$.

Глава 4. Градиентный спуск

1. На каждом шаге отклонение $\theta_1 - 1$ умножается на $1 - 28 \cdot 0{,}1/3 \approx 0{,}067$: $\theta_1 \approx 1{,}067$, затем $\approx 1{,}004$.
2. При $0 < \eta < 1/a$; за один шаг — при $\eta = 1/(2a)$.
3. В точке $\boldsymbol{\theta} = (0;\ 0)$ градиент $\nabla Q = -\frac{2}{3}\mathbf{F}^{\mathrm{T}}\mathbf{y} = (-4;\ -8{,}67)$; $\boldsymbol{\theta} = (0{,}4;\ 0{,}87)$.
4. Прогноз $1 + 0{,}5 \cdot 2 = 2$, остаток $2 - 3 = -1$, градиент $2 \cdot (-1) \cdot (1;\ 2) = (-2;\ -4)$, $\boldsymbol{\theta} = (1{,}2;\ 0{,}9)$; градиент потери на одном объекте в минимуме Q не равен нулю.

Глава 5. Переобучение и валидация

1. MSE $= 5/3 \approx 1{,}67$; RMSE $\approx 1{,}29$; MAE $= 1$; MAPE $\approx 27{,}8\,\%$; $R^2 = 1 - 5/8 = 0{,}375$.
2. 10 раз по 900 объектов; LOO — 1000 обучений по 999 объектов.
3. AIC: $100 \ln 2 + 6 \approx 75{,}3$ против $100 \ln 1{,}8 + 20 \approx 78{,}8$ — первую; BIC штрафует сложность сильнее и тем более выберет первую.
4. Утечка из будущего: для временного ряда нужно разбиение, при котором контроль всегда позже обучения.

Глава 6. Регуляризация

1. $\mathbf{F}^{\mathrm{T}}\mathbf{F} + \mathbf{I} = \begin{pmatrix} 15 & 14{,}3 \\ 14{,}3 & 15{,}61 \end{pmatrix}$, определитель 29,66, $\boldsymbol{\theta}^*_\lambda \approx (0{,}956;\ 0{,}976)$.
2. Гребневая регрессия — (1; $-0{,}17$; 0,4); LASSO — (2; 0; 0,2).
3. В МНК вес уменьшится ровно в 10 раз, прогнозы не изменятся; в

гребневой регрессии штраф на этот вес станет относительно меньше, модель изменится — поэтому признаки стандартизуют.

Глава 7. Логистическая регрессия

1. Для (3; 1): $h = 1$, $\hat{p} = \sigma(1) \approx 0{,}73$, шансы $e \approx 2{,}72$; для (0; 1): $h = -2$, $\hat{p} \approx 0{,}12$, шансы $e^{-2} \approx 0{,}14$.
2. 0,105; 0,693; 2,303 — уверенная ошибка стоит в 22 раза дороже уверенного верного ответа.
3. Разделив числитель и знаменатель на e^{h_1}, получаем $1/\big(1 + e^{-(h_1 - h_2)}\big) = \sigma(h_1 - h_2)$.
4. $\hat{p} = 0{,}5$, градиент $(\hat{p} - \tilde{y})(1;\ 2) = (-0{,}5;\ -1)$, $\theta = (0{,}25;\ 0{,}5)$.

Глава 8. Метрики качества классификации

1. Доля правильных ответов 0,93; точность 0,4; полнота 0,8; $F_1 \approx 0{,}53$; специфичность $890/950 \approx 0{,}94$.
2. К классу +1 отнесены модули 1–3; TP = 2, FP = 1, FN = 2; точность 2/3, полнота 0,5, $F_1 \approx 0{,}57$.
3. Из 6 пар неверно упорядочена одна ($0{,}3 < 0{,}7$), AUC $= 5/6 \approx 0{,}83$.
4. Порог $c_-/(c_- + c_+) = 1/(1 + 20) \approx 0{,}048$: проверять модуль, если вероятность ошибки в нём не меньше 4,8 %.

Глава 9. Наивный байесовский классификатор

1. $0{,}009/(0{,}009 + 0{,}0495) \approx 0{,}15$: здоровых намного больше, и большинство положительных результатов ложные.
2. $h = 2\ln 2 + \ln(2/3) \approx 0{,}98$ (веса слов «бесплатно» и «встреча»), вероятность $8/11 \approx 0{,}73$.
3. Из $\ln 0{,}8 - x^2/2 = \ln 0{,}2 - (x - 4)^2/2$ получаем $x = 2 + \ln 4/4 \approx 2{,}35$: граница сдвинута к редкому классу.
4. $2 \cdot (2^{20} - 1) \approx 2{,}1$ млн и $2 \cdot 20 = 40$.

Список литературы

1. Mitchell, T. M. Machine Learning / T. M. Mitchell. – New York : McGraw-Hill, 1997. – 414 p.
2. Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems / A. Géron. – 3rd ed. – Sebastopol : O'Reilly Media, 2022. – 861 p.
3. Жерон, О. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow : концепции, инструменты и техники для создания интеллектуальных систем / О. Жерон ; перевод с английского Ю. Н. Артеменко. – 2-е изд. – Москва ; Санкт-Петербург : Диалектика, 2020. – 1039 с.
4. Lakshmanan, V. Machine Learning Design Patterns : Solutions to Common Challenges in Data Preparation, Model Building, and MLOps / V. Lakshmanan, S. Robinson, M. Munn. – Sebastopol : O'Reilly Media, 2020. – 408 p.
5. Лакшманан, В. Машинное обучение. Паттерны проектирования / В. Лакшманан, С. Робинсон, М. Мунн ; перевод с английского. – Санкт-Петербург : БХВ-Петербург, 2022. – 448 с.
6. Hastie, T. The Elements of Statistical Learning : Data Mining, Inference, and Prediction / T. Hastie, R. Tibshirani, J. Friedman. – 2nd ed. – New York : Springer, 2009. – 745 p.
7. Хасти, Т. Основы статистического обучения : интеллектуальный анализ данных, логический вывод и прогнозирование / Т. Хасти, Р. Тибширани, Дж. Фридман ; перевод с английского Д. А. Клюшина. – 2-е изд. – Москва ; Санкт-Петербург : Диалектика, 2020. – 768 с.
8. Bergstra, J. Random Search for Hyper-Parameter Optimization / J. Bergstra, Y. Bengio // Journal of Machine Learning Research. – 2012. – Vol. 13. – P. 281–305.
9. Domingos, P. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss / P. Domingos, M. Pazzani // Machine Learning. – 1997. – Vol. 29, no. 2–3. – P. 103–130.
10. McCallum, A. A Comparison of Event Models for Naive Bayes Text Classification / A. McCallum, K. Nigam // Learning for Text Categorization : Papers from the AAAI Workshop, Technical Report WS-98-05. – Menlo Park : AAAI Press, 1998. – P. 41–48.
11. Ng, A. Y. On Discriminative vs. Generative Classifiers : A Comparison of Logistic Regression and Naive Bayes / A. Y. Ng, M. I. Jordan // Advances in Neural Information Processing Systems 14. – Cambridge : MIT Press, 2002. – P. 841–848.

Приложение А. Математический минимум

Здесь собраны сведения из линейной алгебры, математического анализа и теории вероятностей, на которые опираются главы. Доказательства не приводятся, но каждое понятие сопровождается коротким примером.

А.1 Векторы и матрицы

Векторы. Вектор $\mathbf{a} = (a_1, \dots, a_n)^{\mathrm{T}}$ — упорядоченный набор из n чисел; по умолчанию его записывают столбцом. Векторы одной длины складывают и умножают на число покомпонентно. *Скалярное произведение*

$$\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^{\mathrm{T}}\mathbf{b} = \sum_{j=1}^{n} a_j b_j$$

равно нулю, если векторы *ортогональны* (перпендикулярны). Длину вектора измеряют *нормой*. Евклидова норма $\|\mathbf{a}\|_2 = \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle}$ — обычная длина, норма L1 $\|\mathbf{a}\|_1 = \sum_j |a_j|$ — сумма модулей компонент. Для $\mathbf{a} = (3; -4)$: $\|\mathbf{a}\|_2 = 5$, $\|\mathbf{a}\|_1 = 7$.

Матрицы. Матрица $\mathbf{A}$ размера $m \times n$ — таблица чисел a_{ij} из m строк и n столбцов. Произведение матриц $\mathbf{A}\,(m \times n)$ и $\mathbf{B}\,(n \times k)$ — матрица $m \times k$ с элементами

$$(\mathbf{AB})_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk} \ :$$

элемент на пересечении строки i и столбца k — скалярное произведение i-й строки $\mathbf{A}$ на k-й столбец $\mathbf{B}$. Число столбцов первой матрицы должно совпадать с числом строк второй, и в общем случае $\mathbf{AB} \neq \mathbf{BA}$. Вектор — матрица из одного столбца, поэтому $\mathbf{F}\theta$ — вектор, i-я компонента которого равна скалярному произведению i-й строки $\mathbf{F}$ на θ, то есть прогнозу линейной модели на i-м объекте.

Транспонирование $\mathbf{A}^{\mathrm{T}}$ меняет строки и столбцы местами; $(\mathbf{AB})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$. Матрица $\mathbf{A}$ *симметрична*, если $\mathbf{A}^{\mathrm{T}} = \mathbf{A}$; например, $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ симметрична при любой $\mathbf{F}$.

Обратная матрица. *Единичная* матрица $\mathbf{I}$ — квадратная, с единицами на диагонали и нулями вне её; $\mathbf{Ia} = \mathbf{a}$. *Обратная* к квадратной матрице $\mathbf{A}$ — матрица $\mathbf{A}^{-1}$, для которой $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$. Тогда система $\mathbf{A}\theta = \mathbf{b}$ имеет единственное решение $\theta = \mathbf{A}^{-1}\mathbf{b}$. Обратная матрица существует, только если *определитель* $\det \mathbf{A} \neq 0$, или, что то же самое, если столбцы $\mathbf{A}$ *линейно независимы* — ни один не выражается как сумма

остальных, умноженных на числа. Для матрицы 2×2

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}, \qquad \det = ad - bc.$$

Например, у матрицы со строками $(3;\ 6)$ и $(6;\ 14)$ определитель $3 \cdot 14 - 6 \cdot 6 = 6$, и обратная матрица имеет строки $(7/3;\ -1)$ и $(-1;\ 1/2)$; умножением легко проверить, что произведение равно $\mathbf{I}$.

Ортогональные матрицы и собственные значения. Квадратная матрица $\mathbf{U}$ *ортогональна*, если $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$: её столбцы имеют единичную длину и попарно ортогональны (*ортонормированы*). Умножение на такую матрицу поворачивает или отражает вектор, не меняя его длины. Число s и ненулевой вектор $\mathbf{u}$ — *собственное значение* и *собственный вектор* матрицы $\mathbf{S}$, если $\mathbf{Su} = s\,\mathbf{u}$: матрица только растягивает этот вектор в s раз. У симметричной матрицы собственные значения вещественные, а собственные векторы можно выбрать ортонормированными; у матрицы $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ все собственные значения неотрицательны.

A.2 Производная и градиент

Производная $f'(x)$ — скорость изменения функции: при малом приращении Δx значение меняется примерно на $f'(x)\,\Delta x$. Основные формулы:

$$(x^a)' = ax^{a-1}, \qquad (e^x)' = e^x, \qquad (\ln x)' = \frac{1}{x},$$

$$(u + v)' = u' + v', \qquad (uv)' = u'v + uv', \qquad \big(f(u(x))\big)' = f'(u)\,u'(x).$$

Последнее — *правило дифференцирования сложной функции*: производная внешней функции, умноженная на производную внутренней. Например, для логистической потери из гл. 7

$$\big(\ln(1 + e^{-z})\big)' = \frac{1}{1 + e^{-z}} \cdot (-e^{-z}) = -\frac{1}{e^z + 1} = \sigma(z) - 1.$$

В точке минимума или максимума гладкой функции производная равна нулю.

Частные производные и градиент. Если функция зависит от нескольких переменных, $Q(\theta_1, \ldots, \theta_n)$, её *частная производная* $\partial Q/\partial\theta_j$ — обычная производная по θ_j при фиксированных остальных переменных. Вектор частных производных называют *градиентом*:

$$\nabla Q = \left(\frac{\partial Q}{\partial \theta_1}, \dots, \frac{\partial Q}{\partial \theta_n} \right)^{\mathrm{T}}.$$

Градиент указывает направление наискорейшего роста функции, антиградиент $-\nabla Q$ — наискорейшего убывания; в точке минимума $\nabla Q = \mathbf{0}$. Например, для $Q = \theta_1^2 + 3\theta_2^2$ градиент $\nabla Q = (2\theta_1; 6\theta_2)$, в точке (1; 1) он равен (2; 6): функция быстрее всего растёт вдоль θ_2.

Для линейных и квадратичных функций векторного аргумента удобны готовые формулы (матрица $\mathbf{A}$ симметрична):

$$\nabla \langle \mathbf{a}, \theta \rangle = \mathbf{a}, \qquad \nabla \left(\theta^{\mathrm{T}} \mathbf{A} \theta \right) = 2\mathbf{A}\theta, \qquad \nabla \|\mathbf{F}\theta - \mathbf{y}\|_2^2 = 2\mathbf{F}^{\mathrm{T}}(\mathbf{F}\theta - \mathbf{y}).$$

Они аналогичны одномерным $(ax)' = a$ и $(ax^2)' = 2ax$.

Выпуклые функции. Функция *выпукла*, если отрезок, соединяющий любые две точки её графика, лежит не ниже графика. У выпуклой функции любой локальный минимум глобальный, и градиентный спуск не застревает в «ложных» минимумах. Выпуклы θ^2, $|\theta|$, e^{θ}, $\ln(1 + e^{-\theta})$; сумма выпуклых функций и выпуклая функция от линейной функции параметров тоже выпуклы. Поэтому выпуклы эмпирический риск линейной регрессии с квадратичной потерей и логистической регрессии, в том числе со штрафами L1 и L2.

А.3 Вероятность

События. Вероятность $P(A)$ события A — число от 0 до 1. Её можно понимать как долю случаев, в которых A происходит, при многократном повторении опыта; поэтому вероятности оценивают частотами в выборке. *Условная вероятность* A при условии B —

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)},$$

где $A \cap B$ — событие «произошли и A, и B». Отсюда *правило умножения* $P(A \cap B) = P(B)\,P(A \mid B)$. События *независимы*, если $P(A \cap B) = P(A)\,P(B)$, то есть знание о B не меняет вероятность A: $P(A \mid B) = P(A)$. События *условно независимы* при условии C, если $P(A \cap B \mid C) = P(A \mid C)\,P(B \mid C)$; на этом допущении построен наивный байесовский классификатор (гл. 9).

Если события $B_1, \dots, B_K$ попарно несовместны и одно из них обязательно происходит, верна *формула полной вероятности* $P(A) =$

$\sum_k P(B_k)\,P(A \mid B_k)$, а из правила умножения следует *формула Байеса*

$$P(B_k \mid A) = \frac{P(B_k)\,P(A \mid B_k)}{P(A)}.$$

Пример. Пусть 20 % модулей программы большие, и ошибки находят в 30 % больших модулей и в 5 % остальных. По формуле полной вероятности ошибку найдут в доле модулей $0{,}2 \cdot 0{,}3 + 0{,}8 \cdot 0{,}05 = 0{,}1$. Если в модуле нашли ошибку, он большой с вероятностью $0{,}06/0{,}1 = 0{,}6$ — втрое выше, чем доля больших модулей среди всех.

Случайные величины. Случайная величина ξ принимает числовые значения с некоторыми вероятностями. Если значения $x_1, x_2, \ldots$ принимаются с вероятностями $p_1, p_2, \ldots$, *математическое ожидание* (среднее значение) и *дисперсия* равны

$$\mathbb{E}(\xi) = \sum_k x_k p_k, \qquad \mathrm{D}(\xi) = \mathbb{E}\big((\xi - \mathbb{E}(\xi))^2\big) = \mathbb{E}(\xi^2) - \big(\mathbb{E}(\xi)\big)^2.$$

Дисперсия измеряет разброс значений вокруг среднего; корень из неё, $\sqrt{\mathrm{D}(\xi)}$, — *стандартное отклонение*, оно измеряется в тех же единицах, что и ξ. Свойства: $\mathbb{E}(a\xi + b) = a\,\mathbb{E}(\xi) + b$, $\mathrm{D}(a\xi + b) = a^2\,\mathrm{D}(\xi)$; ожидание суммы равно сумме ожиданий, а дисперсия суммы независимых величин — сумме дисперсий. Например, величина, равная 1 с вероятностью p и 0 с вероятностью $1 - p$ (*распределение Бернулли*, модель бинарного ответа), имеет $\mathbb{E}(\xi) = p$ и $\mathrm{D}(\xi) = p(1 - p)$.

По выборке значений $u_1, \ldots, u_\ell$ ожидание оценивают средним $\bar{u} = \frac{1}{\ell}\sum_i u_i$, дисперсию — средним квадратом отклонения $\frac{1}{\ell}\sum_i (u_i - \bar{u})^2$ (иногда делят на $\ell - 1$; при больших ℓ разница мала).

Непрерывные величины. Значения непрерывной величины описывает *плотность* $p(x) \geq 0$: вероятность попасть в интервал (a, b) равна площади под графиком плотности над этим интервалом, $\int_a^b p(x)\,dx$, а вся площадь равна единице. Само значение плотности — не вероятность и может быть больше единицы. Ожидание вычисляют как $\mathbb{E}(\xi) = \int x\,p(x)\,dx$.

Нормальное (гауссовское) распределение $N(\mu, \sigma^2)$ со средним μ и дисперсией σ^2 имеет плотность

$$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\!\left(-\frac{(x - \mu)^2}{2\sigma^2}\right).$$

Её график — симметричный «колокол» с вершиной в μ (рис. А.1). В ин-

тервал $\mu \pm \sigma$ попадает около 68 % значений, в $\mu \pm 2\sigma$ — около 95 %, в $\mu \pm 3\sigma$ — около 99,7 % (правило трёх сигм): значение, отстоящее от среднего больше чем на 3σ, — редкость, и его стоит проверить как возможный выброс.

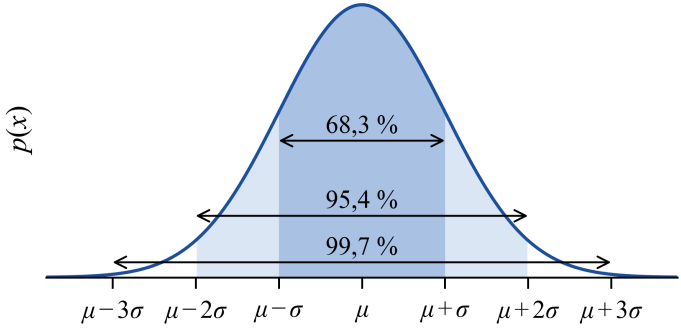


Рис. А.1. Плотность нормального распределения и доли значений в интервалах $\mu \pm k\sigma$

Нормально распределены, по крайней мере приближённо, суммы многих независимых малых случайных слагаемых, поэтому этим распределением часто описывают шум измерений. *Распределение Лапласа* с плотностью $p(x) = \frac{1}{2b} \exp(-|x-\mu|/b)$ имеет более острую вершину и более «тяжёлые» хвосты, чем нормальное.

А.4 Экспонента и логарифм

Число $e \approx 2{,}718$; функция e^x (также $\exp x$) положительна и возрастает. Натуральный логарифм $\ln x$ — обратная к ней функция, определённая при $x > 0$: $\ln e^x = x$. Основные свойства:

$$e^{a+b} = e^a e^b, \qquad \ln(ab) = \ln a + \ln b, \qquad \ln \frac{a}{b} = \ln a - \ln b, \qquad \ln a^k = k \ln a.$$

Логарифм превращает произведение в сумму — поэтому правдоподобие (произведение вероятностей) заменяют его логарифмом. Логарифм возрастает, поэтому положительная функция и её логарифм достигают максимума в одной и той же точке. Логарифмы по разным основаниям отличаются множителем: $\log_2 x = \ln x / \ln 2$.

Приложение Б. Русско-английский словарь терминов

В табл. Б.1 собраны термины, английские названия которых не совпадают с русскими дословно; они пригодятся при чтении документации scikit-learn и литературы на английском языке. В последнем столбце — номер главы, где термин определён.

Таблица Б.1. Термины машинного обучения и их английские эквиваленты

Термин	English	Гл.
апостериорная вероятность	posterior probability	9
априорная вероятность	prior probability	9
априорное распределение	prior distribution	7
базовая модель	baseline	5
бинарная кросс-энтропия	binary cross-entropy, log loss	7
гребневая регрессия	ridge regression	6
доля правильных ответов	accuracy	8
коэффициент детерминации	coefficient of determination	5
коэффициент регуляризации	regularization coefficient, regularization strength	6
кривая точность–полнота	precision–recall curve	8
логарифмическая потеря	logarithmic loss, log loss	8
максимум апостериорной вероятности	maximum a posteriori, MAP	7
максимум правдоподобия	maximum likelihood	7
матрица объекты–признаки	design matrix, feature matrix	1
матрица ошибок	confusion matrix	8
метод наименьших квадратов	least squares method	3
метрика качества	evaluation metric	8
мешок слов	bag of words, BoW	9
минимаксная нормализация	min–max scaling	2
наивный байесовский классификатор	naive Bayes classifier	9
недообучение	underfitting	5
обобщающая способность	generalization ability	5
обучающая выборка	training set	1
обучение без учителя	unsupervised learning	1
обучение с учителем	supervised learning	1
один против всех	one-vs-rest	7
отложенная выборка	hold-out set	5
отступ	margin	7
переобучение	overfitting	5
полнота	recall	8

Термин	English	Гл.
пороговая функция потерь	zero–one loss	7
признак	feature	1
признаковое описание	feature vector	1
прямое кодирование	one-hot encoding	2
разброс	variance	5
разведочный анализ данных	exploratory data analysis	2
разделяющая поверхность	decision boundary	7
разделяющая функция	discriminant function	7
сглаживание Лапласа	Laplace (additive) smoothing	9
скорость обучения	learning rate	4
смещение	bias	5
сокращение весов	weight decay	6
среднеквадратичная ошибка	mean squared error, MSE	5
средняя абсолютная ошибка	mean absolute error, MAE	5
стохастический градиентный спуск	stochastic gradient descent, SGD	4
точность	precision	8
утечка данных	data leakage	2
функционал качества	objective function, training objective	1
функция потерь	loss function	1
число обусловленности	condition number	3
шансы	odds	7
эмпирический риск	empirical risk	1