

В.М. Неделько

**МАШИННОЕ ОБУЧЕНИЕ
В ВЕРОЯТНОСТНОЙ
ПОСТАНОВКЕ**





МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Механико-математический факультет

В. М. Неделько

**МАШИННОЕ ОБУЧЕНИЕ
В ВЕРОЯТНОСТНОЙ ПОСТАНОВКЕ**

Учебник

Новосибирск
2026

УДК 004.852, 519.24
ББК 16.63
Н421

Рецензенты:

д-р техн. наук, доцент В. Б. Бериков,
канд. физ.-мат. наук, доцент С. Е. Хрущев

Неделько, В. М.

Н421 Машинное обучение в вероятностной постановке : учебник / В. М. Неделько ; Новосиб. гос. ун-т. — Новосибирск : ИПЦ НГУ, 2026. — 432 с.

ISBN 978-5-4437-1899-6

Представлен базовый курс по машинному обучению (интеллектуальному анализу данных). Рассмотрены как классические методы дискриминантного и регрессионного анализа, так и современные методы машинного обучения, в частности нейронные сети, логистическая регрессия, SVM, деревья решений, композиции. Особое внимание уделено установлению взаимосвязи разных методов, вопросам выбора подходящего метода в конкретной ситуации, а также управлению процессом обучения и оцениванию качества построенных решений. Отличительными чертами изложения являются математическая строгость в сочетании с практической направленностью, а также структурированное изложение широкого круга тем разной сложности.

Материал предназначен для студентов, изучающих машинное обучение, а также специалистов в области анализа данных.

УДК 004.852, 519.24

ББК 16.63

Рекомендовано к печати кафедрой теоретической кибернетики
механико-математического факультета НГУ
(протокол от 6 февраля 2026 г.)

ISBN 978-5-4437-1899-6

© Неделько В. М., 2026

Оглавление

Введение	12
1 Метод прецедентов	19
1.1 Метод ближайшего соседа в задаче классификации	19
1.2 Случай вещественной целевой переменной	22
1.3 Использование весовой функции	26
1.4 Характеристика метода прецедентов	29
1.4.1 Варианты метода прецедентов	29
1.4.2 Свойства метода прецедентов	30
1.5 Примеры кода	31
Контрольные вопросы	35
2 (Квази)линейные методы классификации	36
2.1 Гипотеза нормальности распределений	36
2.1.1 Одномерный случай	37
2.1.2 Многомерный случай	40
2.1.3 Оценивание параметров	42
2.1.4 Фильтрация нетипичных значений	45
2.1.5 Свойства метода	46
2.2 Дискриминант Фишера	47
2.2.1 Идея метода	47
2.2.2 Свойства метода	48
2.3 Логистическая регрессия	49
2.3.1 Идея метода	49
2.3.2 Оценивание параметров	50

2.3.3	Функция потерь	51
2.3.4	Свойства метода	51
2.4	Наивный байесовский классификатор	52
2.4.1	Случай двух классов	52
2.4.2	Случай нескольких классов	55
2.4.3	Ряд Бахадура	57
2.4.4	Свойства метода	59
2.5	Метод опорных векторов	60
2.5.1	Постановка задачи классификации	61
2.5.2	Линейно разделимая выборка	62
2.5.3	Линейно неразделимая выборка	64
2.5.4	Двойственная задача	65
2.5.5	Ядра и спрямляющие пространства	68
2.5.6	Беспризнаковая классификация	70
2.5.7	Задача безусловной оптимизации	71
2.5.8	Свойства метода	72
2.6	Характеристика линейных методов	73
2.7	Примеры кода	74
2.7.1	Визуализация решений	74
2.7.2	Кодирование категориальных переменных и наивный байесовский классификатор	77
	Контрольные вопросы	78
3	Логические методы классификации	80
3.1	Логические закономерности	80
3.1.1	Понятие закономерности	80
3.1.2	Критерии качества закономерности	83
3.1.3	Решающие списки	86
3.2	Решающие деревья	87
3.2.1	Понятие дерева решений	87
3.2.2	Построение дерева решений	87
3.2.3	Критерии ветвления	91
3.3	Выводы	93
3.4	Примеры кода	94
	Контрольные вопросы	95

4	Методы, основанные на композициях	97
4.1	Композиции решающих функций	97
4.2	Независимые подвыборки	99
4.2.1	Бэггинг	100
4.2.2	Случайный лес	100
4.2.3	Свойства	101
4.3	Бустинг	102
4.3.1	Алгоритм AdaBoost	103
4.3.2	Свойства	105
4.4	Нейронные сети	105
4.4.1	Схема нейрона	106
4.4.2	Аппроксимирующая способность сети	107
4.4.3	Обучение сети	107
4.4.4	Архитектуры нейронных сетей	110
4.4.5	Глубокое обучение	112
4.4.6	Бустинг и нейронные сети	112
4.5	Ансамбли на основе различных методов	115
4.5.1	Простое голосование	115
4.5.2	Стэкинг	117
4.6	Выводы	118
4.6.1	Преимущества ансамблей	118
4.6.2	Бустинг и случайный лес	120
4.7	Примеры кода	120
4.7.1	Подбор сложности ансамбля	120
4.7.2	Визуализация нейронной сети	126
	Контрольные вопросы	128
5	Градиентный бустинг	130
5.1	Алгоритм AdaBoost	130
5.1.1	Непосредственная оптимизация экспоненциальной функции потерь	131
5.1.2	AdaBoost как градиентный бустинг	134
5.1.3	Сходимость процесса бустинга	136
5.2	Градиентный бустинг	137
5.3	Метод XGBoost	140

5.4	Свойства бустинга	146
5.4.1	Представление бустинга	146
5.4.2	Запуск бустинга на распределениях	147
5.4.3	Модельный пример	149
5.4.4	Понятие отступа	152
5.5	Выводы	152
5.6	Примеры кода	153
	Контрольные вопросы	155
6	Оценивание качества решений	156
6.1	Постановка задачи	156
6.1.1	Точечные оценки риска	156
6.1.2	Формальная постановка	157
6.2	Контрольная выборка	162
6.2.1	Точечная оценка	162
6.2.2	Доверительный интервал	163
6.2.3	Усреднённые оценки	166
6.2.4	Случай произвольной функции потерь	168
6.2.5	Свойства оценки по контрольной выборке	173
6.3	Отложенная выборка	174
6.3.1	Определение	174
6.3.2	Несмещённость оценки hold-out	175
6.3.3	Вероятностные характеристики модели	175
6.3.4	Точность оценки hold-out	178
6.4	Скольльзящий экзамен	180
6.4.1	Определение	180
6.4.2	Несмещённость скольльзящего экзамена	182
6.4.3	Точность скольльзящего экзамена	183
6.4.4	Дисперсия скольльзящего экзамена	184
6.4.5	Пример одинаково распределённых классов	187
6.4.6	Статистические оценки	188
6.4.7	Разновидности скольльзящего экзамена	190
6.4.8	Оценка out-of-bag	194
6.4.9	Свойства оценки скольльзящего экзамена	195
6.5	Скольльзящий экзамен и дисперсионный анализ ☆	196

6.5.1	Задача оценивания зависимостей	196
6.5.2	Гипотеза однородности	197
6.5.3	Байесовская модель	200
6.5.4	Усреднённое решение	202
6.5.5	Обсуждение результатов	204
6.6	Практические рекомендации	205
6.6.1	Оценивание дисперсии	205
6.6.2	Валидация и тест	206
6.7	Выводы	207
6.8	Примеры кода	208
	Контрольные вопросы	211
7	Эмпирический риск	212
7.1	Понятие эмпирического риска	212
7.1.1	Основные понятия	212
7.1.2	Смещённость эмпирического риска	213
7.2	Число разбиений выборки	215
7.2.1	Разбиения линейными функциями	215
7.2.2	Комбинаторная размерность	217
7.3	Проверка нулевой гипотезы	218
7.3.1	Статистический критерий	218
7.3.2	Неполная делимость	221
7.4	Оценки Вапника–Червоненкиса	222
7.4.1	Доверительный интервал	222
7.4.2	Использование функции роста	225
7.4.3	Неулучшаемость оценок ☆	227
7.4.4	Точные оценки для гистограммного классификатора	232
7.5	Доверительный интервал для риска	235
7.5.1	Качество точечных оценок	235
7.5.2	Доверительный интервал для риска	236
7.5.3	Свойства эмпирического риска	240
7.6	Примеры кода	240
	Контрольные вопросы	242

8	Задача машинного обучения	243
8.1	Критерии качества	243
8.1.1	Целевая функция потерь	243
8.1.2	Критерии точности классификации	245
8.1.3	Кривая ошибок	248
8.1.4	Функция потерь для оценки вероятности	251
8.1.5	Функция потерь для непрерывной целевой переменной	253
8.2	Теория статистической устойчивости	254
8.2.1	Разложение ошибки	254
8.2.2	Меры адекватности и устойчивости	255
8.2.3	Сложность модели	257
8.2.4	Процесс обучения	258
8.3	Смещение и разброс	261
8.3.1	Bias-variance decomposition	261
8.3.2	Универсальное разложение	262
8.3.3	Логарифмическая функция потерь	263
8.3.4	Разложение для kNN	265
8.3.5	Свойства разложения	267
8.4	Общая постановка задачи машинного обучения	269
8.4.1	Модели	269
8.4.2	Замечания о терминах	270
8.4.3	Задача построения решающей функции	271
8.4.4	Связь с задачей проверки гипотез	272
8.4.5	Классификация методов машинного обучения	273
8.5	Сравнение методов построения решающих функций	275
8.5.1	«Полигон» — 1980-е	275
8.5.2	Архив UCI	276
8.5.3	«Полигон» — 2000-е	276
8.5.4	Платформа Kaggle	277
8.5.5	Синтетические данные	277
8.6	Понятие оптимальности метода	278
8.6.1	Проблема оптимальности методов построения решающих функций	278
8.6.2	Минимаксный подход к оцениванию качества	279

8.6.3	Выбор классов распределений	280
8.7	Некоторые задачи машинного обучения	281
8.8	Выводы	282
8.9	Примеры кода	283
	Контрольные вопросы	286
9	Применение теории вероятностей	287
9.1	Вероятностная интерпретация	287
9.2	Вероятность как математическое понятие	289
9.2.1	Аксиоматическое определение	290
9.2.2	Закон больших чисел	291
9.2.3	Частотное определение вероятности	292
9.2.4	Условное распределение	293
9.3	Вероятность как эмпирическое понятие	298
9.3.1	Эмпирические факты	298
9.3.2	Достоверность и практическая достоверность	301
9.4	Случайность как незнание	303
9.4.1	Парадокс конвертов	303
9.4.2	Байесовский подход	309
9.5	Применимость вероятностных моделей	316
9.5.1	Статистическая устойчивость	316
9.5.2	Проблема объективного сравнения подходов	320
9.5.3	«Измерение» вероятности	321
9.6	Связанные понятия	323
9.6.1	Случайность и Колмогоровская сложность .	323
9.6.2	Детерминированность и случайность	325
9.7	Обсуждение	327
9.7.1	Выводы	327
9.7.2	Замечания	330
10	Некоторые задачи анализа данных	332
10.1	Методы глобальной оптимизации	332
10.1.1	Задача поиска глобального экстремума . . .	333
10.1.2	Формальная постановка	334
10.1.3	Теорема о бесплатном завтраке	335

10.1.4	Алгоритм СПА	338
10.2	Кластерный анализ	339
10.2.1	Метод k -средних	340
10.2.2	Оценивание смеси распределений	340
10.2.3	Иерархическая кластеризация	341
10.3	Анализ временных рядов	342
10.4	Линейная регрессия	343
10.4.1	Постановка задачи	344
10.4.2	Сингулярное разложение	345
10.4.3	Регуляризация	346
10.4.4	Задача прогнозирования	348
10.5	Использование экспертных знаний	350
10.5.1	Постановка задачи	350
10.5.2	Байесовский подход	352
10.5.3	Статистическая интерпретация экспертной информации	352
10.5.4	Примеры	353
10.6	Некоторые задачи и приёмы решения	354
10.6.1	Пример Штейна	354
10.6.2	Прогнозирование нескольких переменных	357
10.6.3	Проблема target leaks	358
10.6.4	Заполнение пропусков	362
10.6.5	Обобщение задачи построения регрессии	363
10.6.6	Кодирование категориальных переменных	364
10.6.7	Визуализация многомерных данных	365
10.6.8	Рекомендации по решению задач анализа данных	368
10.6.9	Вопросы на собеседованиях	370
	Контрольные вопросы	376
11	Проблемы искусственного интеллекта	378
11.1	Большие языковые модели	379
11.1.1	Тестирование возможностей	379
11.1.2	Перспективы использования	394
11.1.3	Перспективы развития	398

11.2 Историческая ретроспектива ИИ	400
11.2.1 Краткий обзор	400
11.2.2 Экспертные системы	402
11.3 Естественный интеллект	405
11.3.1 Интеллект и сознание	405
11.3.2 Виды познания	407
11.4 Примеры кода	408
Заключение	411
Glossarium	413
Литература	427

Введение

В эпоху, когда машинным обучением не занимается только ленивый... Пожалуй, такая формулировка не будет соответствовать реальности. В реальности многие инструменты машинного обучения ориентированы как раз на «ленивого» пользователя: это и полностью сконфигурированные предварительно обученные нейросети, и инструменты для автоматического подбора параметров, и даже инструменты для автоматического выбора метода обучения (модели). Поэтому точнее будет сформулировать тезис следующим образом. В условиях, когда машинным обучением занимается даже ленивый, нет необходимости обосновывать актуальность этого научного направления, поскольку она очевидна.

Возможно, текущие ожидания в этой области даже несколько (а может быть, и значительно) завышены. Однако реальная полезность данных инструментов не вызывает сомнений, равно как и их востребованность в обозримом будущем.

Следует уточнить терминологию. Мы будем использовать как синонимы следующие термины: «машинное обучение», «интеллектуальный анализ данных», «теория статистических решений», «data mining», «распознавание образов». Также будем применять широкий термин «искусственный интеллект», относя к этой области любые вопросы, связанные с автоматизацией интеллектуальной деятельности, т. е. с передачей компьютеру интеллектуальных функций человека. Представленный материал в первую очередь относится к машинному обучению, но некоторые вопросы будут касаться как раз более широкой области.

Можно выделить два подхода к использованию машинного обучения, которые условно назовём «инженерным» и «исследовательским».

Первый подход предполагает использование готовых «рецептов» для построения моделей машинного обучения. Для этого необходимы широкий кругозор, владение последними актуальными инструментами и знание большого числа шаблонов для решения разных практических задач. При этом не требуется глубокого понимания используемых методов и приёмов. По большому счёту достичь такого понимания для всех актуальных инструментов физически нереально ввиду их большого количества и постоянного обновления. Подобная квалификация широко востребована. Вместе с тем такие специалисты испытывают затруднения при решении нестандартных задач, требующих доработки или модификации известных методов.

Чтобы решать нестандартные задачи или разрабатывать новые методы анализа данных, требуется хорошо разобратся в их принципах и тонкостях работы. При этом не получится ограничиться только одной областью машинного обучения (например, только нейросетевыми методами). Нейронные сети являются основным инструментом при решении многих задач анализа данных (хотя и далеко не всех, например на табличных данных нейросети, как правило, уступают бустингу), но инструментом, сложным для анализа и неподходящим для изучения принципов машинного обучения. Чтобы научиться понимать сложные нейросетевые модели, целесообразно потренироваться на более простых моделях.

Если ставить целью получение не только инженерной, но и исследовательской квалификации, то целесообразно системно изучать основные методы и подходы машинного обучения, от простых к более сложным. Именно такую цель преследует данный курс. Большое внимание будет уделено выявлению неочевидных связей между различными методами, а также фундаментальным проблемам машинного обучения. Необходимо разобратся с математическим фундаментом, на котором основывается область искусственного интеллекта.

Следует отметить отличия данного учебника от классических курсов (например, [11; 41]), а также большого количества открытых курсов и пособий по машинному обучению.

Во-первых, существенно отличаются структура и изложение материала. Это связано с тем, что в анализе данных существуют разные научные школы, и если математические факты для всех одинаковы, то их интерпретация и практика применения в разных школах могут различаться. Разумеется, автор не считает своё изложение более «правильным» по сравнению с другими. Материал излагается так, как было бы понятнее самому автору. Предполагается, что читателю в любом случае полезно рассмотреть материал с разных точек зрения.

Во-вторых, здесь, несмотря на относительно небольшой объём, есть материал, который редко встречается в других источниках.

В-третьих, учебник предполагает широкую целевую аудиторию и предназначен как для начального изучения предмета, так и для углубленного. Некоторые темы, возможно, представляют интерес и для специалистов по анализу данных. Такая универсальность достигается за счёт того, что темы выбираются не по их сложности, а по важности.

В-четвёртых, изложение сочетает математическую строгость с практической направленностью. Автор имеет существенный опыт решения реальных задач как в рамках прикладных проектов, так и конкурсов (грандмастер Kaggle).

В книге рассматриваются вопросы, как простые, так и сложные. При этом для сложных вопросов предварительно даётся некоторое краткое изложение основных идей, чтобы при первом чтении можно было пропустить сложные детали, и это не помешало пониманию дальнейшего материала.

Разделы повышенной сложности отмечены ☆.

Имеющиеся учебники по машинному обучению во многих случаях попадают в крайности. Одна часть старается предельно упростить изложение, что приводит к примитивизации и вульгарным представлениям. Другая часть сводит изложение к объяснению терминологии, правил и рецептов решения задач. Третьи ориенти-

рованы на предельную формализацию, рассказывают множество интересных математических фактов, но теряют связь с практическими задачами.

Идея данного учебника — осветить актуальные на практике вопросы математически строго.

Автор старался излагать материал максимально просто и наглядно. Но не всё можно упростить, не исказив смысл. Также не всегда возможно двигаться от простого к сложному. Иногда трудный фрагмент логически должен находиться во вполне определённом месте, и переносить его — значит нарушить логику и связность. При первом чтении будет понятно далеко не всё. Но непонятные фрагменты всё равно как-то отложатся и помогут в формировании полной картины. К этим фрагментам можно будет потом вернуться. Для связности восприятия при первом прочтении достаточно, если будет понятна только половина материала.

Логика изложения построена так, чтобы, двигаясь, от частного к общему, максимально быстро выстроить некоторый «каркас знаний», т. е. получить цельное представление о предмете. А имея минимальную цельную картину, уже можно встраивать в неё любые новые знания. По замыслу автора такая картина должна сложиться к восьмой главе. Все темы, которые не являются необходимыми для формирования минимально полного представления, рассматриваются позже. Пример такой темы — линейная регрессия.

Для понимания предмета читателю потребуется знание основ теории вероятностей в объёме, например, пособий [34; 43; 44], а также математического анализа в объёме типовых университетских курсов.

На самом деле, требуются не столько знания из дисциплин, сколько некоторый навык усвоения материала. Получить пользу от прочтения данной книги может даже школьник (с олимпиадным опытом). Но более эффективным будет изучение после 3-го курса университета.

Автор будет рад получить комментарии и замечания (на адрес: nedelko@math.nsc.ru).

На первый взгляд, в математике не может быть разных точек зрения, поскольку если теорема доказана, то уже не получится с ней не согласиться. Однако разные точки зрения возможны в интерпретации утверждений. Например, теорема Гёделя о неполноте. Иногда её интерпретируют как довод в пользу того, что в математике существуют утверждения, которые нельзя ни доказать, ни опровергнуть. Но из самой теоремы подобного не следует.

В машинном обучении довольно много подобных примеров, когда некоторые строгие утверждения используются для аргументации некоторых гипотез, которые лишь косвенно связаны с доказанными утверждениями. В данном материале мы будем особо акцентировать внимание на таких «общеизвестных фактах», которые на самом деле требуют аккуратной интерпретации.

Особо отметим следующее. Если в монографии результат приводится без ссылок, то это значит, что результат принадлежит автору. Если в учебнике результат приводится без ссылки — значит, автор считает результат общеизвестным. Здесь именно так: все собственные результаты автора приведены со ссылкой; всё, что дано без ссылки, предполагается общеизвестным.

Автор настоятельно рекомендует (до или после чтения данного учебника) ознакомиться с [11]. Это классический курс по машинному обучению. Он включает конспекты лекций, презентации и видеозаписи лекций. Кроме того, на ресурсе `machinelearning` имеется много полезных статей данного автора, в том числе рекомендации по ведению научной работы для студентов.

Также рекомендуется (попытаться) изучить учебник [41]. Но это только в качестве следующего шага, поскольку он труднее для понимания.

Для дальнейшего чтения можно порекомендовать [18]. Статьи этого автора (который длительное время имел самый высокий рейтинг на `kaggle`) полезны как источник практических рекомендаций и приёмов для решения реальных задач [2].

Для расширения кругозора полезно знакомство с классическими источниками [1; 4; 21].

При этом автор настоятельно не рекомендует начинать изуче-

ние предмета с online-учебников и интернет-курсов (даже очень хороших). Они зачастую представляют собой набор статей разных авторов, поэтому по ним трудно получить цельное представление о материале. В части подобных источников многие вопросы освещаются поверхностно.

В текущих реалиях невозможно представить себе эффективное обучение чему-либо без использования интернет-ресурсов. Для изучения предмета можно рекомендовать следующие источники.

<http://kaggle.com/> — ресурс, который нужно использовать в обязательном порядке. Первоначально это была просто платформа для проведения конкурсов по анализу данных. Однако очень быстро стал понятен её образовательный потенциал, а также значение как «полигона» для сравнения различных методов анализа данных, который позволяет оценить их реальную полезность. Главную образовательную ценность на этом ресурсе имеют законченные примеры решения реальных задач лучшими специалистами, представленные в виде полностью рабочего кода с пояснениями.

<https://archive.ics.uci.edu/> — UCI (University of California, Irvine) Machine Learning Repository. Наиболее известная (до появления kaggle) коллекция задач машинного обучения, содержащая несколько сотен наборов данных из различных прикладных задач. На задачи этого репозитория принято ссылаться в академических публикациях, где они используются в качестве реальных данных. Использование одинаковых данных разными исследователями делает получаемые результаты воспроизводимыми.

<http://www.machinelearning.ru/> — информационный русскоязычный ресурс по анализу данных. Содержит статьи разного уровня по качеству (важно смотреть на авторство статьи), но большое число статей полезно в качестве источника справочной информации.

<https://education.yandex.ru/handbook/ml> — достаточно качественный справочник по машинному обучению. Не реко-

мендуется для первоначального знакомства с предметом, однако полезен для дополнительного обучения.

В классической математической статистике существует два основных подхода: подход, основанный на правдоподобию (использует статистические критерии и доверительные вероятности), и байесовский подход.

Для понимания проще байесовский подход, вероятно, как раз с этим связано то, что в большинстве книг по машинному обучению используют именно его. И большая часть этих книг условно заслуживает прочтения [37]. Но в некоторых источниках байесовский подход продвигается агрессивно [30] и с использованием не вполне корректных примеров (часть из них разберём в подразделе¹ 9.4.2).

В данном учебнике байесовский подход не используется (за исключением единственного примера). Автор считает, что машинное обучение целесообразно сначала изучать именно так, поскольку, если привыкнуть пользоваться предположениями (лежащими в основе байесовского подхода), то потом крайне сложно понять, как обходиться без них.

При этом байесовский подход — безусловно полезный инструмент (автор сам его использует [29]), но он требует дополнительных предположений, и это выглядит как недостаток.

В названии курса говорится о машинном обучении в вероятностной постановке. Нужно оговорить, что такое название вовсе не означает каких-либо ограничений на круг рассматриваемых вопросов и методов. Мы будем рассматривать в том числе и вероятностные методы и подходы. А вероятностная постановка упомянута скорее как намерение по возможности использовать вероятностную интерпретацию задач и результатов. Иными словами, речь будет идти о «самом обычном» машинном обучении.

¹По правилам русской типографики главы делятся на параграфы, а параграфы — на пункты. В англоязычной традиции главы состоят из разделов и подразделов, и этот вариант получает всё большее распространение в русскоязычной литературе. Мы используем компромиссный вариант: главы делятся на параграфы, а параграфы — на подразделы.

Глава 1

Метод прецедентов

Метод прецедентов является, пожалуй, наиболее простым методом машинного обучения и наиболее интуитивным. Фактически он отражает естественную человеческую стратегию поведения: для принятия решения в новой ситуации найти похожие случаи из прошлого опыта и действовать по аналогии.

Метод прецедентов является на самом деле не конкретным методом, а некоторым подходом (классом методов), в рамках которого существует ряд методов, по-разному реализующих идею прецедентов, или аналогии.

Данный класс методов также называют *метрическими методами* построения решающих функций. При этом методы вовсе не требуют, чтобы функция (не)похожести объектов была именно метрикой.

1.1 Метод ближайшего соседа в задаче классификации

Для знакомства с методом прецедентов рассмотрим одну простую практическую задачу машинного обучения, а именно задачу «Iris» из репозитория UCI.

Задача состоит в том, чтобы построить правило, позволяющее

по внешним признакам различать три вида (сорта) ирисов (цветы):

- 1 – Iris Setosa,
- 2 – Iris Versicolour,
- 3 – Iris Virginica.

Вид ириса обозначим переменной Y , которая будет принимать значения 1, 2, 3.

В качестве измеряемых характеристик используются:

- X_1 – длина чашелистика (sepal length),
- X_2 – длина лепестка (petal length).

Таким образом, требуется по измеренным значениям переменных X_1 и X_2 определить вид ириса.

Результаты измерений указанных характеристик для 24 экземпляров растений трёх видов приведены в табл. 1.1. Здесь i обозначает номер объекта (экземпляра) и в верхней позиции обозначает не степень, а верхний индекс, x_1^i – значения переменной X_1 , x_2^i – значения переменной X_2 , y^i – значения переменной Y .

Заметим, что в исходной таблице данных, доступной в репозитории, используются четыре характеристики, а число объектов составляет 150, однако для удобства мы рассматриваем сокращённую таблицу.

Для наглядности изобразим имеющиеся данные на диаграмме (рис. 1.1). Требуется, проанализировав представленные данные (будем называть их выборкой), научиться распознавать сорт ириса по внешним признакам (X_1 и X_2). Математически это означает, что необходимо построить функцию f , сопоставляющую любой паре (x_1, x_2) значений переменных X_1, X_2 некоторое значение y переменной Y .

Такая функция называется *решающей функцией* или *решающим правилом*

$$f : X_1 \times X_2 \rightarrow Y,$$

где $f(x_1, x_2) \in Y$, $x_1 \in X_1$, $x_2 \in X_2$. Здесь и далее мы обозначаем одинаково переменную и множество значений, которые она

Таблица 1.1. Данные по ирисам

i	x_1^i	x_2^i	y^i	i	x_1^i	x_2^i	y^i	i	x_1^i	x_2^i	y^i
1	5,1	1,4	1	9	7,0	4,7	2	17	6,3	6,0	3
2	4,9	1,4	1	10	6,4	4,5	2	18	5,8	5,1	3
3	4,7	1,3	1	11	6,9	4,9	2	19	7,1	5,9	3
4	4,6	1,5	1	12	5,5	4,0	2	20	6,3	5,6	3
5	5,0	1,4	1	13	6,5	4,6	2	21	6,5	5,8	3
6	5,4	1,7	1	14	5,7	4,5	2	22	7,6	6,6	3
7	4,6	1,4	1	15	6,3	4,7	2	23	4,9	4,5	3
8	5,0	1,5	1	16	4,9	3,3	2	24	7,3	6,3	3

принимает.

Идея метода прецедентов заключается в следующем. Для объекта, который нужно классифицировать, находятся наиболее похожие на него объекты, для которых целевой признак известен.

Простейшая модификация метода заключается в том, что распознаваемый объект относится к тому же классу, к которому принадлежит объект, на который он более всего похож. Такой метод носит название *метода ближайшего соседа*.

В качестве «меры (не)похожести» можно использовать расстояние ρ между точками, представляющими объекты в признаковом пространстве. Например, для точки $(6,7,5,6)$, которая обозначена на рис. 1.1 крестиком, ближайшей точкой выборки будет обведённая жёлтым квадратом точка $x^{21} = (6,5,5,8)$, которая принадлежит третьему классу ($y = 3$), поэтому точке $(6,7,5,6)$ также припишем третий класс, т. е. $f(6,7,5,6) = 3$.

Введём формальное обозначение для выборки

$$S = ((x_1^i, x_2^i, y^i) \mid i = 1, \dots, N) = ((x^i, y^i) \mid i = 1, \dots, N),$$

где $x^i = (x_1^i, x_2^i)$, N – количество объектов (объём выборки).

Здесь мы просто ввели буквенные обозначения для таблицы данных и её содержимого.

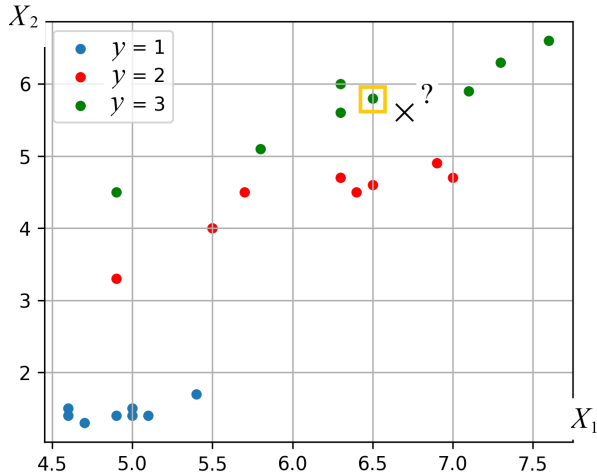


Рис. 1.1. Визуализация данных из табл. 1.1

Теперь решающую функцию, получаемую методом ближайшего соседа, можно записать формально

$$f(x) = y^{i_x}, \quad i_x = \arg \min_i \rho(x, x^i),$$

где $\arg \min$ – аргумент минимума, т. е. в данном случае это индекс i , при котором достигается минимум расстояния.

Существует модификация метода, когда вместо одной точки находятся несколько ближайших точек выборки и приписывается класс, к которому принадлежит большинство из них. Это метод *k* ближайших соседей (или kNN).

1.2 Случай вещественной целевой переменной

В предыдущем разделе переменная Y принимала конечный набор значений, на которых не был определён порядок (так как

на сортах ирисов не определено отношение «больше-меньше»). В таких случаях задача построения решающей функции называется задачей *распознавания образов* или задачей *классификации* (с «учителем»).

Если целевая переменная Y является вещественной, то говорят о задаче восстановления зависимостей. Задачу построения решающей функции при вещественной Y можно также называть задачей регрессионного анализа, хотя это будет не совсем точно, поскольку регрессионный анализ предполагает, что мы оцениваем функцию регрессии, т. е. функцию условного математического ожидания. Однако функция регрессии является оптимальным решением, если в качестве критерия используется средний квадрат отклонения (MSE). Если, например, критерием является средний модуль отклонения (MAE), то оптимальным решением будет не регрессия, а условная медиана.

Кроме того, классическая постановка регрессионного анализа подразумевает, что значения переменных, кроме целевой, не являются случайными. Мы же полагаем все переменные случайными величинами.

Однако термин «восстановление зависимостей» тоже не является точным, поскольку подразумевает, что есть некоторая «истинная» зависимость, которую нужно восстановить. Вместе с тем в реальных задачах чаще всего никакой «объективной» зависимости нет, есть только статистическая (вероятностная) связь (зависимость) между переменными. При этом функциональную зависимость (решающую функцию) мы не восстанавливаем, а конструируем. Поэтому будем пользоваться термином *построение решающей функции*.

В качестве примера задачи построения решающей функции для вещественной (непрерывной) целевой переменной рассмотрим задачу «Wine Quality» из репозитория UCI. База данных для этой задачи описывает несколько тысяч образцов вин, которые характеризуются одиннадцатью переменными, соответствующими измерениям различных химических показателей. Целевая переменная Y принимает целые значения от 3 до 8 и является экспертной

Таблица 1.2. Данные по задаче оценивания качества вин

i	x_1^i	x_2^i	y^i	i	x_1^i	x_2^i	y^i	i	x_1^i	x_2^i	y^i
1	1,02	0,02	3	11	0,70	0,13	5	21	0,36	0,34	7
2	0,44	0,42	3	12	0,59	0,01	5	22	0,30	0,41	7
3	0,61	0,49	3	13	0,49	0,49	5	23	0,59	0,00	7
4	0,76	0,02	3	14	0,82	0,29	5	24	0,34	0,40	7
5	1,19	0,00	3	15	0,69	0,49	5	25	0,59	0,06	7
6	0,33	0,32	4	16	0,58	0,56	6	26	0,35	0,53	8
7	0,58	0,00	4	17	0,56	0,24	6	27	0,62	0,67	8
8	0,92	0,27	4	18	0,39	0,47	6	28	0,49	0,03	8
9	0,48	0,20	4	19	0,49	0,10	6	29	0,32	0,45	8
10	1,13	0,09	4	20	0,51	0,13	6	30	0,30	0,56	8

оценкой качества вина.

Поскольку дробные оценки качества не противоречат содержательному смыслу задачи, будем считать переменную Y вещественной.

Как и в предыдущем случае, уменьшим размерность задачи. Выберем две переменные:

X_1 – концентрация летучих кислот (volatile acidity),

X_2 – концентрация лимонной кислоты (citric acid).

Из всего множества объектов выберем по пять образцов для каждого значения качества. Заметим, что после такого сокращения данных решение, которое будет получено, нельзя переносить на исходную задачу, фактически мы сформулировали новую учебную задачу на основе данных UCI.

Сформированная выборка приведена в табл. 1.2 и визуализирована на рис. 1.2, где цифры рядом с точками отражают значения целевой переменной.

Задача заключается в построении решающей функции, которая по значениям химических показателей позволяет оценить ка-

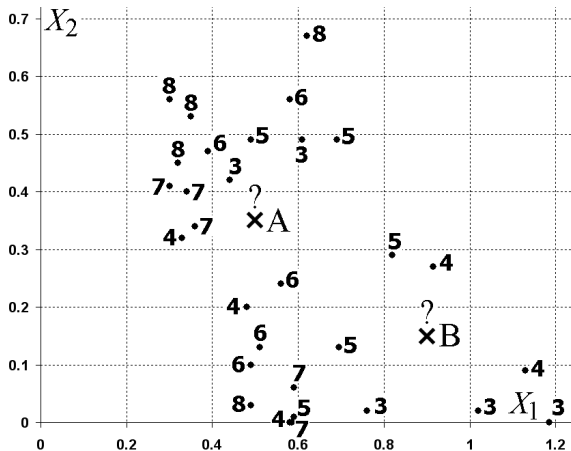


Рис. 1.2. Визуализация данных из табл. 1.2

чество вина.

Пусть, например, мы получили образец, для которого $x_1 = 0,5$, $x_2 = 0,35$. Он изображён точкой A на рис. 1.2. Что можно сказать о его качестве? Наиболее похожим из выборки является объект $i = 2$ со значениями $(0,44, 0,42)$, для которого $y = 3$. Таким образом, по методу ближайшего соседа оценка качества для объекта A также составит 3.

Однако заметим, что недалеко от A находятся и точки, для которых $y = 7$. И при незначительном смещении точки A оценка по методу ближайшего соседа может сильно измениться.

Подобное скачкообразное изменение решающей функции представляется неоправданным, поскольку разумно считать, что качество продукта в данном случае является непрерывной функцией характеристик.

Гладкость решающей функции можно обеспечить, если построить её, например, в виде

$$f(x) = \frac{1}{w} \sum_{i=1}^N y^i w_i, \quad w = \sum_{i=1}^N w_i, \quad w_i = \varphi(\rho(x, x^i)),$$

где $\rho(x, x^i)$ – расстояние между заданной точкой и объектом выборки, а $\varphi(\cdot)$ – некоторая весовая функция.

В качестве весовой функции можно использовать любую неотрицательную монотонно убывающую функцию, например функцию Коши:

$$\varphi_h(z) = \frac{1}{1 + \left(\frac{z}{h}\right)^2},$$

где h – параметр, задающий скорость убывания функции. Будем называть его «шириной окна».

На содержательном уровне идея метода в том, что решение в каждой точке есть взвешенное среднее по выборочным значениям, причём, чем ближе выборочная точка к данной, тем с большим весом она учитывается.

Для примера вычислим оценки качества в точках A и B . При $h = 0,1$ в точке A имеем $f(0,5, 0,35) \approx 5,49$, для точки B получаем $f(0,9, 0,15) \approx 4,7$.

Данный результат выглядит достаточно разумным. Действительно, согласно выборке, качество вина выше при меньшей концентрации летучих кислот (в том числе уксусной), и большей концентрации лимонной кислоты.

На практике функция Коши в роли весовой функции имеет существенный недостаток (медленное убывание) и используется редко. Но её можно подходящим образом модифицировать, например вычтя некоторую константу и «занулив» значения, ставшие отрицательными.

1.3 Использование весовой функции

Опишем идею использования весов в более общем виде.

В случае непрерывной целевой переменной решающая функция строится в виде

$$f(x) = \frac{1}{w} \sum_{i=1}^N y^i w_i, \quad w = \sum_{i=1}^N w_i, \quad w_i = \varphi\left(\frac{\rho(x, x^i)}{h}\right), \quad (1.1)$$

где $\rho(x, x^i)$ – «расстояние» (не обязательно метрика) между заданной точкой и объектом выборки, а $\varphi(\cdot)$ – весовая функция.

Выражение (1.1) называется *формулой Надарая-Ватсона* и играет важную роль в непараметрической статистике.

Весовую функцию мы также будем называть *оконной функцией*, а также *функцией ядра* (или ядерной функцией, или просто ядром). С некоторой оговоркой её можно называть потенциальной функцией.

Заметим, что может показаться более привычным называть ядром функцию двух аргументов (именно такие ядра в интегральных преобразованиях или методе опорных векторов). Если

$$K(x, x^i) = \varphi\left(\frac{\rho(x, x^i)}{h}\right),$$

то $K(\cdot, \cdot)$ будет именно ядром, а $\varphi(\cdot)$ точнее было бы назвать весовой функцией. Но следуя сложившейся традиции, $\varphi(\cdot)$ также будем называть ядром.

В задаче классификации (для категориальной целевой переменной) мы также можем строить решение в виде (1.1), но только в случае двух классов. При этом для меток классов желательно выбрать $Y = \{0, 1\}$. В результате мы получим в качестве прогноза вещественное число, округлив которое до целого, получим метку класса.

В случае нескольких классов нужно сделать модификацию

$$f_\omega(x) = \frac{1}{w} \sum_{i=1}^N I(y^i = \omega) \cdot w_i, \quad (1.2)$$

$$w = \sum_{i=1}^N w_i, \quad w_i = \varphi\left(\frac{\rho(x, x^i)}{h}\right),$$

$I(\cdot)$ – индикаторная функция (превращает логическое выражение в 0 или 1, т. е. в терминах языков программирования это просто приведение булевого типа к целому).

Если специальным образом нормировать выражение (1.2), то получим *оценку Парзена–Розенблатта* для условной плотности.

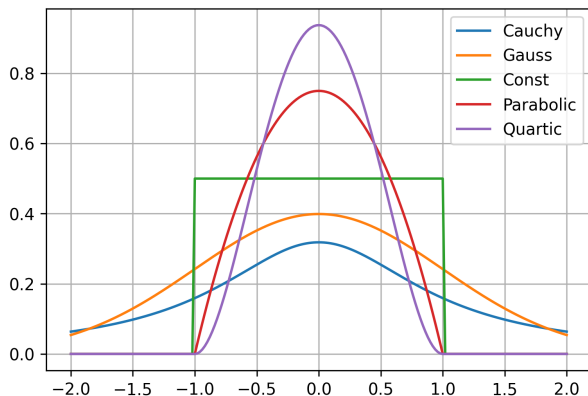


Рис. 1.3. Примеры ядер: Cauchy – плотность Коши, Gauss – нормальная плотность, Const – прямоугольное ядро, Parabolic – ядро Епанечникова, Quartic – биквадратное

Решающая функция в методе Парзена будет выглядеть как

$$f(x) = \arg \max_{\omega} f_{\omega}(x).$$

Недостатком окна фиксированной ширины являются большие колебания числа выборочных точек, попадающих в окрестность. Чтобы решить эту проблему, можно h сделать функцией от местоположения. Можно задать $h(x, S)$, например, таким образом, чтобы в окрестность любой точки x попадало одно и то же число объектов выборки.

Выбор ядра в методе прецедентов является эвристикой. Можно выбирать ядро перебором вариантов, максимизируя точность на скользящем экзамене.

На рис. 1.3 приведены некоторые функции ядра.

В непараметрической статистике [40] существуют некоторые формальные критерии для оценки ядер. В частности, вводится понятие эффективности, в соответствии с которым «лучшим» ядром оказывается ядро Епанечникова

$$\varphi_E(u) = \frac{3}{4} (1 - u^2) \cdot I(|u| \leq 1).$$

Однако из этого не следует, что это ядро даёт наиболее точные решения в задачах машинного обучения и является лучшим выбором.

Также можно в качестве ядер рассмотреть варианты весовых функций, используемых в оконном преобразовании Фурье.

1.4 Характеристика метода прецедентов

1.4.1 Варианты метода прецедентов

Можно выделить следующие основные разновидности метода прецедентов:

- ✧ k ближайших соседей (прогноз на основе объектов, ближайших к исследуемому);
- ✧ парзеновское окно (прогноз на основе объектов, попавших в окрестность исследуемого объекта);
- ✧ сглаживание ядром (вклад объекта в прогноз есть некоторая функция расстояния).

В методе парзеновского окна фиксируется не число соседей, а форма и размер «окна», т.е. окрестности, из которой берутся соседи.

Сглаживание с использованием ядер есть наиболее общий вариант метода прецедентов. При этом функция ядра может быть как функцией непосредственно расстояния, так и функцией ранга по степени близости (т.е. функция номера «соседа» в порядке близости). Тогда методы k NN и парзеновского окна соответствуют ядру в виде пороговой функции.

В методе прецедентов присутствует гиперпараметр: число соседей либо ширина окна.

Значение этого параметра может задаваться эвристически (интуитивно), но лучше его подобрать с помощью процедуры скользящего экзамена (глава 6).

Заметим, что для метода прецедентов проведение скользящего экзамена почти не требует дополнительных вычислительных за-

трат. Достаточно найти для каждого объекта обучающей выборки ближайших соседей в количестве, на единицу большем, чем предполагается k . Среди найденных соседей обязательно будет сам объект (поскольку расстояние до самого себя нулевое), который мы при скользящем контроле исключаем (не используем для прогноза).

1.4.2 Свойства метода прецедентов

Метод прецедентов, несмотря на естественность и очевидность, на практике работает относительно плохо, хотя есть исключения.

Недостатки метода:

- ✧ равное использование всех переменных, неустойчивость к «шумовым переменным», особенно при большой размерности;
- ✧ неиспользование возможной независимости переменных.

Основным недостатком метода является его малая пригодность для пространств большой размерности. При большом числе переменных понятие близости между точками становится малосодержательным и не отражает интуицию.

Представим, что в некоторой задаче есть переменная, по которой два класса безошибочно разделяются. И есть ещё 99 переменных, на которых распределения обоих классов почти совпадают (такие переменные называют «шумовыми»). В такой задаче k NN даст очень низкую точность, поскольку на расстояние между объектами влияют все переменные, и объекты, близкие по информативной переменной, весьма вероятно будут далёкими во всём пространстве.

Существуют методы, существенно более эффективные (на большинстве практических задач).

Тем не менее метод k NN вполне можно использовать в некоторых случаях.

Во-первых, это задачи небольшой размерности, где расстояние «работает».

Во-вторых, существуют методы отбора информативных переменных, которые иногда позволяют уменьшить размерность задачи.

В-третьих, метод kNN иногда используют на случайных подпространствах небольшой размерности, чтобы сформировать новые признаки в методе стэкинга (позже его обсудим).

И в-четвёртых, существует важный подход, называемый *metric learning*. В этом подходе нейросеть обучается специально таким образом, чтобы сформировать признаковое описание (*embedding*) объекта именно так, чтобы объекты одного класса (или близкие по целевой переменной) были близкими в этом пространстве. Фактически само пространство строится именно так, чтобы kNN работал, и он в этом случае работает.

1.5 Примеры кода

Для наглядности приведём примеры кода, позволяющего визуализировать данные, упомянутые в этой главе. Код демонстрирует некоторые полезные приёмы и инструменты для работы с данными.

```
# Загружаем библиотеки:
```

```
import numpy as np
import pandas as pd
from sklearn import datasets
import seaborn as sns
```

```
# Загружаем набор данных Ирисы:
```

```
iris = datasets.load_iris()
```

```
# Формируем таблицу:
```

```
df = pd.DataFrame(iris[ 'data' ],
                  columns = iris[ 'feature_names' ])
```

```
# Добавляем столбец с целевой переменной:
```

```
df[ 'target' ] = iris[ 'target' ]
```

```
# Заменяем номера классов названиями сортов:
```

```
df[ 'target' ] = df[ 'target' ].map(lambda x :
```

```
iris ['target_names' ][x])
```

```
# Смотрим, что получилось:  
df
```

Результат выполнения кода приведён на рис. 1.4. Здесь видим полную таблицу данных Iris. Дополнительно показан результат операции вычисления моментов по классам.

Код запускался в среде jupyter-notebook, которая предоставляет запускать код частями и видеть результат выполнения каждого фрагмента. Также важной особенностью является то, что после модификации кода нет необходимости запускать программу с начала — можно продолжить выполнение в текущем контексте.

Пожалуй, для первоначального знакомства с языком python данная среда подходит лучше всего.

На рис. 1.5 приведён результат выполнения кода

```
sns.pairplot(df, hue = 'target')
```

Показан только фрагмент рисунка, который в полном виде представляет собой 4×4 таблицу картинок и имеет слишком большой размер.

Видим, что библиотека seaborn позволяет одним вызовом получать достаточно сложные и информативные иллюстрации.

В данном учебнике примеров кода будет относительно мало, поскольку большой объём кода был бы неуместен. При этом примеры отобраны только наиболее важные с точки зрения иллюстрации работы методов или демонстрации важных конструкций языка.

Настоятельно рекомендуется запустить все предложенные примеры и понять работу кода, в том числе внося в него изменения и наблюдая результат.

```
df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
df.groupby(['target']).agg(['mean', 'std'])
```

target	sepal length (cm)		sepal width (cm)		petal length (cm)		petal width (cm)	
	mean	std	mean	std	mean	std	mean	std
setosa	5.006	0.352490	3.428	0.379064	1.462	0.173664	0.246	0.105386
versicolor	5.936	0.516171	2.770	0.313798	4.260	0.469911	1.326	0.197753
virginica	6.588	0.635880	2.974	0.322497	5.552	0.551895	2.026	0.274650

Рис. 1.4. Выполнение кода в jupyter-notebook

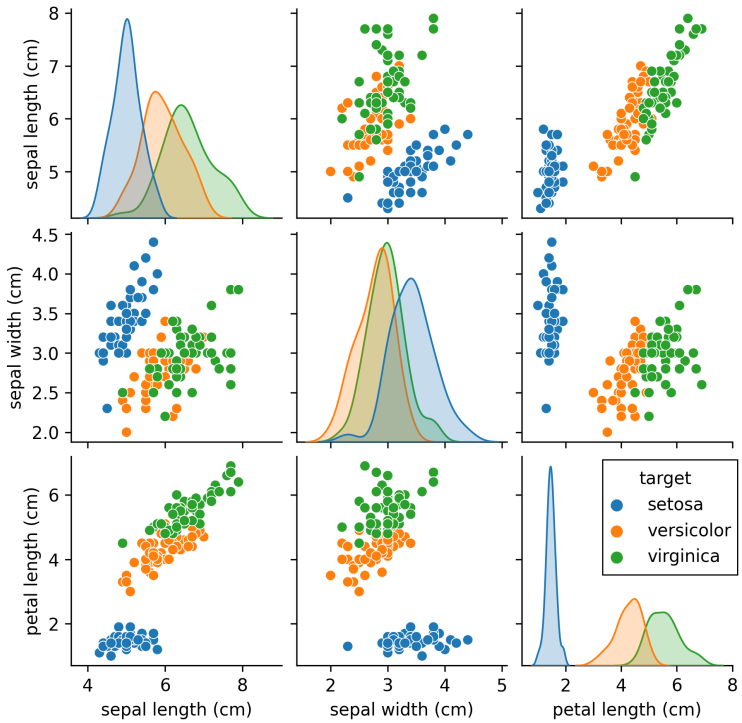


Рис. 1.5. Визуализация средствами seaborn

Контрольные вопросы

1. В чём состоит задача классификации? Что такое решающая функция?
2. На что влияет число соседей в методе kNN?
3. Каков основной недостаток метода прецедентов?
4. Зачем используются функции ядра?
5. Какова вычислительная трудоёмкость метода kNN?
6. Какова область применимости метода прецедентов?
7. Как методом kNN получить вероятностный прогноз?
8. Как оценить точность решения, полученного методом прецедентов?

Глава 2

(Квази)линейные методы классификации

В данной главе рассмотрены методы классификации, в которых решающая функция является линейной либо становится линейной в подходящем пространстве переменных.

Список методов:

- ✧ линейный и квадратичный дискриминант;
- ✧ дискриминант Фишера;
- ✧ логистическая регрессия;
- ✧ наивный байесовский классификатор;
- ✧ машина опорных векторов.

2.1 Гипотеза нормальности распределений

Нормальное распределение является, пожалуй, наиболее часто используемым в теории вероятностей и математической статистике. В задачах анализа данных обычно нет оснований предполагать нормальность распределений, однако в ряде случаев такую модель оправдано использовать.

Рассмотрим, как выглядит оптимальное решение, если распре-

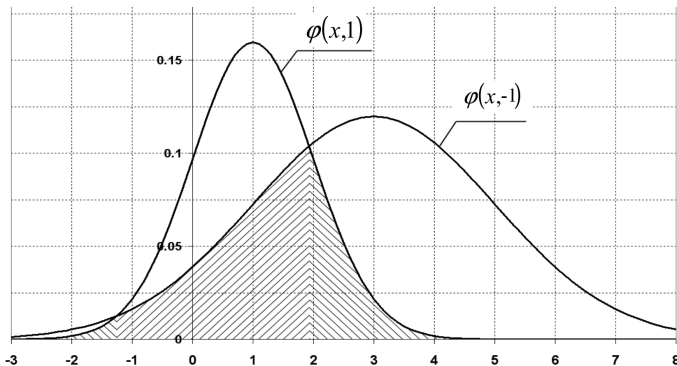


Рис. 2.1. Совместная плотность вероятности

деления известны.

2.1.1 Одномерный случай

Для простоты будем рассматривать случай двух классов, т. е. $Y = \{-1, 1\}$, и пусть имеется одна непрерывная переменная X .

Предположим, что заданы распределение $P(y) = P_y$ и условные плотности вероятности $\varphi_y(x)$, $y \in \{-1, 1\}$, причём плотности являются нормальными

$$\varphi_y(x) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{(x-\mu_y)^2}{2(\sigma_y)^2}}.$$

Через заданные величины можно выразить совместную плотность вероятности

$$\varphi(x, y) = \varphi_y(x)P(y).$$

В качестве примера на рис. 2.1 приведён график совместной плотности при $P(1) = 0,4$, $P(-1) = 0,6$, $\mu_1 = 1$, $\sigma_1 = 1$, $\mu_{-1} = 3$, $\sigma_{-1} = 2$.

Оптимальная (или байесовская) решающая функция будет выглядеть следующим образом

$$f^*(x) = \arg \max_y \varphi(x, y). \quad (2.1)$$

При $\varphi(x, 1) = \varphi(x, -1)$ функция $\arg \max$ возвращает оба значения, т.е. множество $\{-1, 1\}$. В этом случае значение байесовской решающей функции может выбираться произвольно, как 1 так и -1 , причём при практической реализации часто имеет смысл делать случайный (равновероятный) выбор между этими значениями.

Заметим, что в определении (2.1) байесовской решающей функции формула Байеса не использовалась, поэтому термин «байесовская», пожалуй, является не вполне удачным, однако он устоявшийся, и мы будем его использовать.

На рис. 2.1 компоненты совместной плотности имеют две точки пересечения: при $x = a \approx -1,26$ и при $x = b \approx 1,93$. Оптимальная решающая функция

$$f^*(x) = \begin{cases} 1, & x \in [a, b] \\ -1, & x \notin [a, b] \end{cases}.$$

Вероятность ошибочной классификации для оптимального решения есть

$$R^* = P(f^*(x)) = \int_{f^*(x)=-1} \varphi(x, 1) dx + \int_{f^*(x)=1} \varphi(x, -1) dx.$$

Для приведённого примера

$$\begin{aligned} R^* &= \int_{-\infty}^a \varphi(x, 1) dx + \int_a^b \varphi(x, -1) dx + \int_b^{\infty} \varphi(x, 1) dx = \\ &= \frac{1}{2}P(1) \left(1 + \Phi \left(\frac{a - \mu_1}{\sigma_1} \right) + 1 - \Phi \left(\frac{b - \mu_1}{\sigma_1} \right) \right) + \\ &\quad + \frac{1}{2}P(-1) \left(\Phi \left(\frac{b - \mu_{-1}}{\sigma_{-1}} \right) - \Phi \left(\frac{a - \mu_{-1}}{\sigma_{-1}} \right) \right) \approx 0,24. \end{aligned}$$

Полученное значение есть площадь заштрихованной области на рис. 2.1.

Здесь интегралы от нормальной плотности выражены через так называемую *функцию Лапласа*

$$\Phi(z) = \frac{2}{\sqrt{2\pi}} \int_0^z e^{-\frac{t^2}{2}} dt,$$

значения которой могут быть найдены в справочных таблицах либо вычислены на компьютере.

Введём *функции условной вероятности классов*

$$g_1(x) = P(y = 1 | x) = \varphi_1(x) \frac{P_1}{\varphi(x)}, \quad g_{-1}(x) = 1 - g_1(x),$$

где

$$\varphi(x) = \varphi_1(x)P_1 + \varphi_{-1}(x)P_{-1}.$$

Здесь использована формула Байеса.

Байесовская решающая функция может быть записана как

$$f^*(x) = \arg \max_y g_y(x).$$

Если максимум достигается на обоих классах, можем выбрать любой, например класс 1.

Поскольку функции $g_1(x)$ и $g_{-1}(x)$ в рассматриваемом случае положительны, их сравнение друг с другом эквивалентно сравнению их отношения с единицей, что, в свою очередь, эквивалентно сравнению логарифма отношения с нулём, поэтому байесовскую решающую функцию можно представить в виде

$$f^*(x) = \begin{cases} 1, & l(x) \geq 0 \\ -1, & l(x) < 0 \end{cases},$$

где

$$l(x) = \ln \frac{g_1(x)}{g_{-1}(x)} = \ln \varphi_1(x) - \ln \varphi_{-1}(x) + \ln \frac{P(1)}{P(-1)}.$$

Функцию $l(x)$ будем называть *разделяющей функцией*.

Мы также можем использовать стандартную функцию знака (сигнум)

$$\text{sign}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}.$$

Можем записать $f^*(x) = \text{sign}(l(x))$. В этом случае мы допускаем, что на границе классов прогноз будет 0, что можно интерпретировать как «отказ от прогноза». Это ни на что не влияет, поскольку происходит на множестве меры 0.

Подставляя нормальные плотности, получаем

$$l(x) = \frac{(x - \mu_{-1})^2}{(\sigma_{-1})^2} - \frac{(x - \mu_1)^2}{(\sigma_1)^2} + \ln \frac{\sigma_{-1}}{\sigma_1} + \ln \frac{P(1)}{P(-1)}.$$

Видим, что $l(x) = 0$ представляет собой квадратное уравнение, которое имеет либо два корня, либо один корень — в случае $\sigma_1 = \sigma_{-1}$, либо не имеет корней — если одна из компонент совместной плотности целиком лежит под другой (также возможен очевидный вырожденный случай бесконечного множества корней — при полном совпадении компонент).

Заметим, что имеет место следующее соотношение

$$g_1(x) = P(y = 1|x) = \frac{1}{1 + e^{-l(x)}} = \sigma(l(x)).$$

Здесь $\sigma(z) = \frac{1}{1+e^{-z}}$ — так называемая *логистическая функция* (иногда также называемая *сигмоидом*, или *логит-функцией*). Данная функция, как мы увидим в дальнейшем, очень часто возникает при решении задачи классификации и играет важную роль.

2.1.2 Многомерный случай

Рассмотрим теперь случай нескольких переменных X_1, \dots, X_n .

Условные плотности вероятности $\varphi_y(x)$ предполагаются нормальными и задаются параметрами μ_y и λ_y , где μ_y — вектор средних (математических ожиданий) для класса y , а λ_y — ковариационная матрица.

Оптимальная решающая функция записывается так же, как и в одномерном случае, через разделяющую функцию

$$l(x) = \ln \varphi_1(x) - \ln \varphi_{-1}(x) + \ln \frac{P(1)}{P(-1)}.$$

Решение уравнения $l(x) = 0$ задает границу между классами и называется *разделяющей поверхностью*.

Подставим в разделяющую функцию аналитические выражения для нормальных плотностей

$$\varphi_y(x) = \frac{1}{(2\pi)^{n/2} |\lambda_y|^{n/2}} e^{-\frac{1}{2} Q_y(x)},$$

где $Q_y(x) = (x - \mu_y)' (\lambda_y)^{-1} (x - \mu_y)$.

Штрих в записи квадратичной формы обозначает транспонирование вектора.

После преобразований имеем

$$2l(x) = Q_{-1}(x) - Q_1(x) + \ln |\lambda_{-1}| - \ln |\lambda_1| + 2 \ln P(1) - 2 \ln P(-1).$$

Далее преобразуем

$$\begin{aligned} Q_{-1}(x) - Q_1(x) &= \\ &= (x - \mu_{-1})' (\lambda_{-1})^{-1} (x - \mu_{-1}) - (x - \mu_1)' (\lambda_1)^{-1} (x - \mu_1) = \\ &= x' ((\lambda_{-1})^{-1} - (\lambda_1)^{-1}) x - (\mu_{-1}' (\lambda_{-1})^{-1} - \mu_1' (\lambda_1)^{-1}) x - \\ &\quad - x' ((\lambda_{-1})^{-1} \mu_{-1} - (\lambda_1)^{-1} \mu_1) + \\ &\quad + \mu_{-1}' (\lambda_{-1})^{-1} \mu_{-1} - \mu_1' (\lambda_1)^{-1} \mu_1. \end{aligned}$$

Наконец получаем

$$2l(x) = x' Ax + bx + c, \tag{2.2}$$

где

$$\begin{aligned} A &= (\lambda_{-1})^{-1} - (\lambda_1)^{-1}, \\ b &= 2\mu_1 (\lambda_1)^{-1} - 2\mu_{-1} (\lambda_{-1})^{-1}, \\ c &= \mu_{-1}' (\lambda_{-1})^{-1} \mu_{-1} - \mu_1' (\lambda_1)^{-1} \mu_1 + \ln \frac{|\lambda_{-1}|}{|\lambda_1|} + 2 \ln \frac{P_1}{P_{-1}}. \end{aligned}$$

При получении выражения для b использованы коммутативность скалярного произведения векторов и коммутативность произведения вектора на симметричную матрицу (напомним, что ковариационная матрица симметрична).

Уравнение $l(x) = 0$ в общем случае задает поверхность второго порядка, которая при $\lambda_1 = \lambda_{-1}$, т.е. при равенстве ковариационных матриц, вырождается в гиперплоскость.

Заметим, что ковариационные матрицы являются положительно определенными, ввиду чего линии (поверхности) уровня нормального распределения представляют собой эллипсоиды. При этом разделяющая поверхность может оказаться любой поверхностью второго порядка: эллипсоидом, гиперboloидом, параболоидом, конусом, цилиндром.

2.1.3 Оценивание параметров

Пусть имеется выборка, представленная прямоугольной матрицей

$$S = ((x^i, y^i) | i = 1, \dots, N), \quad x^i = (x_1^i, \dots, x_n^i).$$

Параметр N представляет собой число независимых реализаций системы случайных величин и называется *объёмом выборки*, n – число переменных. Верхний индекс указывает номер объекта, нижний индекс соответствует номеру переменной.

Сформируем множество индексов (номеров) объектов, принадлежащих первому классу: $I_1 = \{i | y^i = 1\}$. Аналогично, $I_{-1} = \{i | y^i = -1\}$. Тогда $N_y = |I_y|$ есть число объектов класса y в выборке, $N_1 + N_{-1} = N$.

Проще всего оценить априорные (безусловные) вероятности классов, а именно: $\tilde{P}_y = \frac{N_y}{N}$, где \tilde{P}_y – оценка для P_y .

Обозначим через $\tilde{\mu}_y$ оценку вектора μ_y .

Компоненты вектора средних и ковариационной матрицы для каждого класса оцениваются по формулам

$$\begin{aligned} \tilde{\mu}_y &= \frac{1}{N_y} \sum_{i \in I_y} x^i, \\ \tilde{\lambda}_y &= \frac{1}{N_y} \sum_{i \in I_y} (x^i - \tilde{\mu}_y)(x^i - \tilde{\mu}_y)'. \end{aligned} \quad (2.3)$$

где I_y – множество индексов объектов класса y из выборки.

Здесь произведение векторов вычисляется по правилу умножения матриц, причём (поскольку транспонирован правый сомножитель) умножается вектор-столбец на вектор-строку, в результате чего получается квадратная матрица оценок ковариаций.

Оценки (2.3) являются оценками максимального правдоподобия. Можно использовать несмещённую версию оценок, но на типичных размерах выборки различие пренебрежимо.

Для построения выборочной разделяющей поверхности достаточно теперь в уравнение подставить вместо параметров распределений найденные оценки.

Отдельно рассмотрим случай равных матриц ковариации.

Если известно, что распределения обоих классов имеют одинаковые матрицы ковариации, то нужно всю выборку использовать для вычисления оценки $\tilde{\lambda}$ матрицы $\lambda = \lambda_1 = \lambda_{-1}$. Оценка находится следующим образом

$$\tilde{\lambda} = \frac{1}{N} \sum_{y \in \{1, -1\}} \sum_{i \in I_y} (x^i - \tilde{\mu}_y)(x^i - \tilde{\mu}_y)'. \quad (2.4)$$

Остальные параметры оцениваются так же, как в общем случае.

В качестве примера рассмотрим задачу об ирисах, изложенную в главе 1.

Поскольку мы разобрали метод применительно к случаю двух классов, выберем для классификации два вида ирисов: Iris Versicolour и Iris Virginica.

Если предположить ковариационные матрицы классов различными, то, оценив параметры распределения, получим разделяющую кривую, которая оказалась гиперболой, график которой приведён на левой диаграмме рис. 2.2.

Если предположить равенство ковариационных матриц, то получим разделяющую прямую, график которой приведён на правой диаграмме рис. 2.2.

На практике обычно нет информации о том, являются ли ковариационные матрицы классов одинаковыми, поэтому решение о способе их оценивания приходится принимать эмпирически.

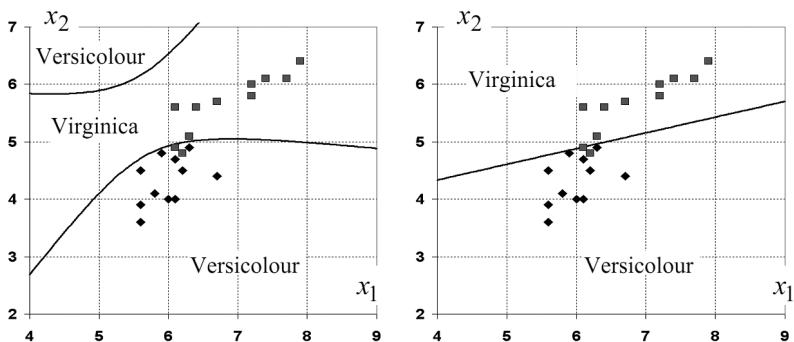


Рис. 2.2. Квадратичный (слева) и линейный (справа) дискриминант в задаче Iris

Можно, например, применить статистический критерий проверки гипотезы о равенстве ковариационных матриц. Но на практике это нецелесообразно.

Разумный вариант: при большом объеме выборки оценивать матрицы для каждого класса отдельно, при малых выборках — оценивать одну матрицу и строить линейную разделяющую поверхность.

Дело в том, что квадратичный дискриминант требует оценить порядка n^2 параметров (напомним, что n — число переменных), а линейный дискриминант — порядка n параметров. Может показаться, что линейный дискриминант тоже оценивает ковариационную матрицу (где n^2 значений), но на самом деле важно именно число степеней свободы в решении, а линейное решение определяется уравнением разделяющей гиперплоскости, т. е. имеет $n + 1$ параметров.

Здесь мы впервые сталкиваемся с эффектом, что при малых выборках следует строить более простое решающее правило, даже если это упрощение вносит заведомую погрешность. В дальнейшем такая ситуация будет частой.

Решение о выборе между линейным и квадратичным дискриминантом целесообразно принимать на основе кросс-валидации.

Таким образом, даже казалось бы в простейшем случае, когда форма распределений известна, а именно нормальна, оптимальный метод построения решающей функции по конечной выборке неизвестен. При «малых» выборках мы строим линейный дискриминант (даже если точно знаем, что ковариационные матрицы не равны), при «больших» — квадратичный. Кажется, что в подобной ситуации должны быть и промежуточные варианты.

2.1.4 Фильтрация нетипичных значений

Описанный метод сильно чувствителен к «выбросам», т.е. нетипичным значениям переменных, сильно отличающимся от нормальных. Такие значения могут быть вызваны как ошибками измерений (или ошибками переноса значений в таблицу), так и наличием нехарактерных объектов в выборке (например, растение повреждено вредителями и не выросло до обычных размеров).

Для решения проблемы выбросов существуют отработанные методы робастных оценок. Общая идея таких оценок заключается в следующем. Сначала строятся оценки параметров по всей выборке. После чего вся выборка упорядочивается по значению правдоподобия, рассчитанного по этим параметрам. Далее некоторая доля (например, 1%) наименее правдоподобных точек выборки отбрасывается, после чего параметры вновь оцениваются по оставшимся объектам. Эту процедуру можно повторять несколько раз.

Такой алгоритм является достаточно универсальным (может применяться с разными методами) и позволяет эффективно отфильтровывать нетипичные значения в данных. Следует только иметь в виду, что после такой фильтрации в оценках условных вероятностей классов нужно делать поправки. Грубо говоря, если мы удалили из выборки 1% объектов, то ни в одной точке пространства мы не должны давать для функции $g_y(x)$ оценку меньше 0,01 и больше 0,99.

2.1.5 Свойства метода

Можно отметить следующие особенности рассмотренного метода:

- ✧ основан на предположении нормальности распределений;
- ✧ чувствителен к «выбросам», но эта проблема решается;
- ✧ имеет низкую вычислительную трудоёмкость, решение в аналитическом виде;
- ✧ выбор между линейным и квадратичным решениями не однозначен.

Первую особенность, как правило, позиционируют как недостаток метода. Однако её можно считать, наоборот, достоинством. Аргументация этого следующая.

Если в какой-то задаче мы знаем, что распределения классов нормальны, это, очевидно, является некоторым обоснованием применения метода (линейного или квадратичного дискриминанта).

Если же мы не имеем оснований полагать распределения нормальными, или даже точно знаем, что они не являются нормальными, то из этого совершенно не следует, что данный метод в такой задаче работать не будет.

Это совершенно общее соображение: если что-то работает при некоторых предположениях, то это не значит, что оно не работает при нарушении предположений. Достаточность условий не подразумевает их необходимость.

Если распределения не являются нормальными, мы всё равно можем применять линейный и квадратичный дискриминант, но уже как чисто эвристический метод.

Заметим, что почти все (другие) методы машинного обучения изначально являются эвристическими. Это не значит, что они не имеют обоснования. Про обоснованность методов машинного обучения будем говорить в главе 8.

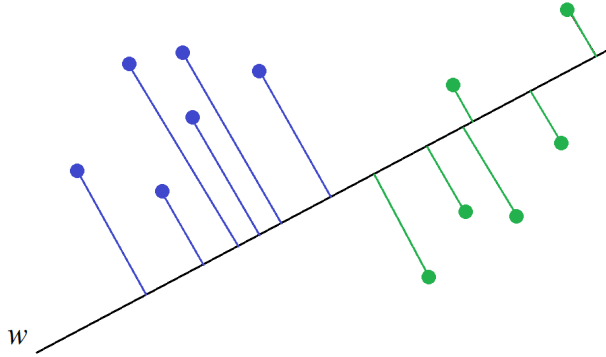


Рис. 2.3. Пример проецирования выборки на направление

2.2 Дискриминант Фишера

2.2.1 Идея метода

Идея метода заключается в выборе направления, при проецировании выборки на которое образы классов оказываются наиболее удалёнными друг от друга.

Это выражается в максимизации критерия

$$\Phi(w) = \frac{(\tilde{\mu}_{w,1} - \tilde{\mu}_{w,-1})^2}{\tilde{S}_{w,1} + \tilde{S}_{w,-1}}, \quad (2.5)$$

где $\tilde{\mu}_{w,y} = \frac{1}{N_y} \sum_{i \in I_y} wx_i$ – среднее,

а $\tilde{S}_{w,y} = \frac{1}{N_y} \sum_{i \in I_y} (wx_i - \tilde{\mu}_{w,y})^2$ – средний квадрат отклонений проекций.

Этот критерий имеет очень простой содержательный смысл: два множества точек тем дальше друг от друга, чем дальше их средние и чем меньше разброс точек.

Требуется найти оптимальное направление.

Оказывается, что в (2.5) максимум $\Phi(w)$ достигается при

$$w_{\Phi} = \tilde{S}^{-1}(\tilde{\mu}_1 - \tilde{\mu}_{-1}), \quad (2.6)$$

где $\tilde{\mu}_y$ – среднее точек выборки y -го класса, а \tilde{S}_y – выборочная ковариационная матрица y -го класса, $\tilde{S} = \tilde{S}_1 + \tilde{S}_{-1}$.

Полученное выражение неожиданно оказывается практически совпадающим с (2.2) при равных матрицах ковариаций.

Отличия незначительны, но их стоит отметить.

Дискриминант Фишера не предполагает равенства ковариационных матриц и использует усреднённую матрицу по классам. В отличие от (2.4), классы здесь дают вклад с равными весами, независимо от соотношения классов в выборке.

На самом деле это свойство можно изменить, скорректировав критерий (2.5), но это бы противоречило идее критерия. Здесь мы не предполагаем равенство ковариационных матриц, поэтому оцениваем каждую отдельно, а потом усредняем (с равным весом). А в линейном дискриминанте предполагаем равенство матриц, поэтому оцениваем одну матрицу по полной выборке.

Другое отличие в том, что дискриминант Фишера даёт только направление, но не описывает способ постановки границы между проекциями классов. Предполагается, что если задача сведена к одномерной, то разумным образом разделить классы не составит труда (например, непосредственной минимизацией числа ошибок).

2.2.2 Свойства метода

Замечания.

- ✧ Метод не требует никаких вероятностных предположений.
- ✧ Выражение для w_f очень похоже на выражение для нормали к разделяющей гиперплоскости для случая нормальных распределений с равными матрицами ковариаций.
- ✧ Метод предполагает оценивание по выборке только $n + 1$ параметров и не требователен к объёму выборки.
- ✧ Метод неустойчив к «выбросам», но позволяет использовать приём робастного оценивания.

Мы получили нетривиальный факт, что метод, выведенный

из очень сильных предположений (нормальность и равенство ковариаций) можно получить вообще без всяких предположений, только лишь на основе геометрических эвристик.

Это позволяет сделать вывод, что метод, выведенный при сильных предположениях, может оставаться пригодным и при нарушении этих предположений.

2.3 Логистическая регрессия

2.3.1 Идея метода

Рассмотрим функцию условной вероятности для класса 1

$$g(x) = P(y = 1|x) = \frac{P(1)\varphi_1(x)}{P(1)\varphi_1(x) + P(-1)\varphi_{-1}(x)} = \frac{1}{1 + e^{-l(x)}},$$

где $l(x)$ – разделяющая функция.

Подставив нормальные плотности при равных матрицах ковариаций, получим

$$g(x) = \frac{1}{1 + e^{-(wx+w_0)}} = \sigma(wx + w_0),$$

где w и w_0 есть переобозначение параметров b и c соответственно из линейного дискриминанта (2.2).

Здесь $\frac{1}{1+e^{-z}}$ – так называемая логистическая функция (иногда также называемая сигмоидом или логит-функцией).

Определить (полную) вероятностную модель данных мы можем, либо задав совместную плотность $\varphi(x, y)$, либо задав пару $\varphi_y(x)$, $P(y)$, либо задав пару $P(y|x)$, $\varphi(x)$. Все способы эквивалентны. Два последних варианта переводятся друг в друга формулой Байеса.

В методе квадратичного дискриминанта мы предполагали нормальность $\varphi_y(x)$ и затем находили $g_y(x) = P(y|x)$.

Возникает разумный вопрос: зачем делать предположения о форме $\varphi_y(x)$, если нам нужна $g_y(x)$. Почему не сделать сразу предположение о форме $g_y(x)$?

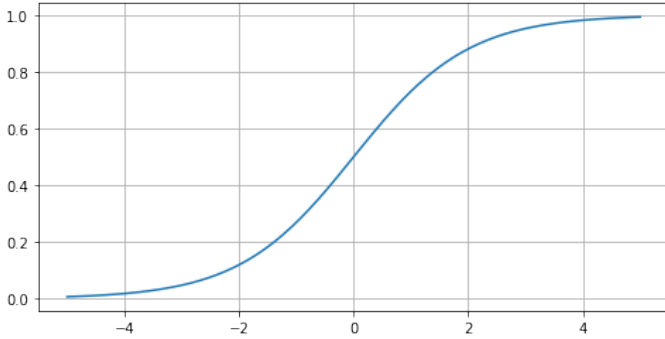


Рис. 2.4. График функции $\sigma(z)$

Оказывается, что так сделать можно и именно в этом заключается идея логистической регрессии.

При этом форма безусловной плотности $\varphi(x)$ нас не интересует, и мы о ней делать предположений не будем. В этом смысле мы будем строить неполную вероятностную модель. Эта модель ничего не говорит о вероятности появления (правдоподобии) x , но описывает условную вероятность $P(y | x)$.

Параметры такой модели оказывается возможным оценивать уже привычным методом максимального правдоподобия. Мы подбираем такие параметры, чтобы вероятность имеющейся выборки была максимальна.

2.3.2 Оценивание параметров

Метод логистической регрессии основан на оценивании функции условной вероятности моделью $\tilde{g}(x) = \sigma(\tilde{w}x + \tilde{w}_0)$, в которой \tilde{w} и \tilde{w}_0 – настраиваемые параметры.

На практике параметры модели обычно оцениваются путём максимизации критерия правдоподобия (условной вероятности выборки).

$$-K_S(\tilde{w}, \tilde{w}_0) = \sum_{i \in I_1} \ln \tilde{g}(x^i) + \sum_{i \in I_{-1}} \ln(1 - \tilde{g}(x^i)).$$

В случае линейно разделимой выборки максимум правдоподобия достигается на ступенчатой функции, которая является пределом сигмоида, когда $w^2 \rightarrow \infty$.

Формально в классе сигмоидов максимум не достигается, поэтому алгоритм численной оптимизации не сойдётся.

Чтобы обеспечить как существование решения, так и его единственность, требуется *регуляризация*. Это некоторая добавка к критерию, обеспечивающая устойчивость решения.

В данной задаче такой добавкой может быть $\varkappa(w^2)$ – некоторая монотонная функция от нормы вектора направления.

2.3.3 Функция потерь

Мы впервые встретились с методом, который сводится к минимизации аддитивной функции выборки. Дальнейшие методы именно так и будут устроены.

Метод минимизирует функцию

$$K_S(\tilde{w}, \tilde{w}_0) = \varkappa(w^2) + \sum_{i=1}^N L(\sigma(\tilde{w}x^i + \tilde{w}_0), y^i), \quad (2.7)$$

где

$$L(p, y) = \begin{cases} -\ln(p), & y=1 \\ -\ln(1-p), & y=-1 \end{cases} \quad (2.8)$$

представляет собой логарифмическую функцию потерь.

Учитывая, что $\ln \sigma(-z) = \ln(1 - \sigma(z))$, выражение (2.7) можно преобразовать к виду

$$K_S(\tilde{w}, \tilde{w}_0) = \varkappa(w^2) + \sum_{i=1}^N -\ln(\sigma(y^i \cdot (\tilde{w}x^i + \tilde{w}_0))). \quad (2.9)$$

Вид получившейся функции потерь приведён на рис. 2.7, страница 72.

2.3.4 Свойства метода

Замечания.

- ✧ Метод логистической регрессии похож на линейный дискриминант, но ослабляет вероятностные предположения.
- ✧ На практике (почти) никто вероятностные предположения не проверяет.
- ✧ По сравнению с линейным дискриминантом метод логистической регрессии более устойчив в «выбросам».
- ✧ Для повышения устойчивости требуется регуляризатор.

Хотя идейно логистическая регрессия близка к линейному дискриминанту, по свойствам она окажется больше похожа на метод опорных векторов.

2.4 Наивный байесовский классификатор

«Наивный» байесовский классификатор основан на предположении условной независимости переменных. Данное предположение редко обосновано в реальных задачах, поэтому метод и получил такое название.

Заметим, что метод использует формулу Байеса, но не байесовский подход, т. е. не предполагает распределений на параметрах или гипотезах.

2.4.1 Случай двух классов

Из формулы Байеса можем записать

$$\begin{aligned}
 g(x) &= P(y = 1|x) = \\
 &= \frac{P(dx, y = 1)}{P(dx, y = 1) + P(dx, y = -1)} = \frac{1}{1 + \frac{1-p}{p} \cdot \frac{P(dx|y=-1)}{P(dx|y=1)}},
 \end{aligned}$$

где $p = P(y = 1)$.

Данное выражение можно преобразовать к виду

$$g(x) = \sigma \left(z_0 + \ln \frac{P(dx|y = 1)}{P(dx|y = -1)} \right),$$

где введено обозначение $z_0 = \ln p - \ln(1 - p) = \sigma^{-1}(p)$.

Здесь $\sigma(z) = \frac{1}{1+e^{-z}}$ — уже знакомая нам логистическая функция (сигмоид).

Выразим правую часть

$$\ln \frac{P(dx|y = 1)}{P(dx|y = -1)} = \sigma^{-1}(g(x)) - z_0. \quad (2.10)$$

Для ряда читателей, возможно, окажется непривычной используемая здесь нотация в духе производных Радона–Никодима (производная по мере). В этом случае можно при чтении везде мысленно заменить $P(dx\dots)$ на плотность вероятности $\varphi(x\dots)$. Если непонятно, как это сделать, можно посмотреть выкладки следующего подраздела, где как раз используются плотности. Там мы выведем формулу для нескольких классов, из которых в частном случае получается то, что выведем сейчас.

Смысл использования производных по мере исключительно в том, чтобы сделать выкладки универсальными, а именно пригодными для случаев, когда пространство переменных образуют случайные величины любых типов: дискретные, (абсолютно) непрерывные, сингулярные. В противном случае для дискретных и непрерывных величин потребуются отдельные выражения.

Поскольку переменные разных типов — типичная ситуация в анализе данных, оказывается полезным изучить такие конструкции как производные по мере и интегралы Лебега и Лебега–Стилтьеса.

Независимость переменных означает, что совместное распределение раскладывается в произведение маргинальных распределений (т. е. одномерных распределений по каждой переменной). Поэтому для (условно) независимых переменных имеем

$$\frac{P(dx|y = 1)}{P(dx|y = -1)} = \prod_{j=1}^n \frac{P(dx_j|y = 1)}{P(dx_j|y = -1)}.$$

После логарифмирования получим равносильное выражение

$$\ln \frac{P(dx|y = 1)}{P(dx|y = -1)} = \sum_{j=1}^n \ln \frac{P(dx_j|y = 1)}{P(dx_j|y = -1)}.$$

Выразим обе части этого равенства через $g(x)$ и $g_j(x)$.

Аналогично формуле (2.10) мы можем вывести выражение для функций $g_j(x_j) = P(y = 1|x_j)$, а именно

$$\ln \frac{P(dx_j|y = -1)}{P(dx_j|y = 1)} = \sigma^{-1}(g_j(x)) - z_0.$$

Собрав всё вместе, получаем

$$\sigma^{-1}(g(x)) - z_0 = \sum_{j=1}^n (\sigma^{-1}(g_j(x_j)) - z_0). \quad (2.11)$$

Если ввести новые переменные $z_j = \sigma^{-1}(g_j(x_j)) - z_0$, то в пространстве этих переменных решающая функция будет иметь вид логистической регрессии

$$g(x) = \sigma \left(w_0 + \sum_{j=1}^n w_j z_j \right) \quad (2.12)$$

при $w_0 = z_0$, $w_j \equiv 1$.

Предположение о независимости переменных на самом деле редко выполняется, поэтому метод, на нём основанный, действительно может считаться наивным. Однако различные варианты частичной независимости переменных вполне типичны для реальных задач. Мы можем «ослабить» гипотезу независимости, полагая w_j в (2.12) свободными параметрами (и подбирая их самым обычным методом логистической регрессии).

Такая замена переменных (x_j на z_j) называется *target encoding* (хорошего перевода нет).

Приём target encoding, т. е. кодирование значений категориальной переменной преобразованной прогнозируемой вероятностью, является одной из стандартных техник машинного обучения. Мы можем на переменных z_j использовать не только логистическую регрессию, но и вообще любой метод машинного обучения.

2.4.2 Случай нескольких классов

В большинстве случаев мы не рассматриваем специально случай нескольких классов, поскольку для большинства методов переход от двух классов ко многим осуществляется типовыми приёмами, которые мы рассмотрим в главе, специально посвящённой общим приёмам.

Однако для рассматриваемого метода рассмотрение нескольких классов приводит к важным формулам.

Итак, цель — выразить условную вероятность заданного класса при условии конкретных значений всех переменных через условные вероятности этого класса по каждой переменной, т. е. требуется выразить $P(y = \omega|x)$ через $P(y = \omega|x_j)$, $j = 1, \dots, n$.

Введём обозначения

$$u_j^\omega = \ln P(y = \omega|x_j) - \ln P(y = \omega).$$

Для простоты выкладок предположим, что все переменные X_j — абсолютно непрерывные случайные величины, т. е., что существует (многомерная) плотность вероятности $\varphi(x)$. Для произвольных случайных величин выкладки требуют привлечения производных по мере, и читатель может провести их самостоятельно по аналогии с выкладками предыдущего подраздела.

Из формулы Байеса можем записать

$$P(y = \omega|x) = \frac{\varphi_\omega(x)P(\omega)}{\varphi(x)},$$

где $\varphi_\omega(x)$ — условная плотность вероятности, $P(\omega)$ — вероятность класса ω .

По предположению о независимости переменных

$$\varphi_\omega(x) = \prod_{j=1}^n \varphi_{\omega,j}(x_j).$$

Ещё раз применим формулу Байеса:

$$\varphi_{\omega,j}(x_j) = \frac{P(y = \omega|x_j)\varphi_j(x_j)}{P(\omega)}.$$

Соберём всё воедино

$$P(y = \omega|x) = \frac{P(\omega)}{\varphi(x)} \prod_{j=1}^n \frac{P(y = \omega|x_j)\varphi_j(x_j)}{P(\omega)}.$$

Обозначим

$$C(x) = \frac{1}{\varphi(x)} \prod_{j=1}^n \varphi_j(x_j)$$

и

$$\begin{aligned} \zeta_\omega(x) &= \ln P(\omega) + \sum_{j=1}^n (\ln P(y = \omega|x_j) - \ln P(\omega)) = \\ &= \ln P(\omega) + \sum_{j=1}^n u_j^\omega. \end{aligned}$$

В этих обозначениях

$$P(y = \omega|x) = C(x)e^{\zeta_\omega(x)}.$$

Величина $C(x)$ нам неизвестна. Её можно найти из условия, что сумма вероятностей всех классов (при любом условии) равна 1. Получаем

$$P(y = \omega|x) = \frac{e^{\zeta_\omega(x)}}{\sum_{v=1}^k e^{\zeta_v(x)}}. \quad (2.13)$$

Цель достигнута.

Выражение, что мы получили, есть не что иное как хорошо известная (например, из нейросетей) функция softmax, которая является обобщением логистической функции.

Убедимся, что в случае двух классов выражение (2.13) превращается в (2.11). Имеем

$$P(y = 1|x) = \frac{e^{\zeta_1(x)}}{e^{\zeta_1(x)} + e^{\zeta_{-1}(x)}} = \frac{1}{1 + e^{\zeta_{-1}(x) - \zeta_1(x)}} = \sigma(\zeta_1(x) - \zeta_{-1}(x)).$$

При этом

$$\zeta_1(x) - \zeta_{-1}(x) = \ln P(1) - \ln P(-1) + \sum_{j=1}^n (u_j^1 - u_j^{-1}).$$

Заметив, что $u_j^1 - u_j^{-1} = \sigma^{-1}(g_j(x_j))$, получаем искомое.

2.4.3 Ряд Бахадура

Рассмотрим интересный подход, касающийся случая бинарных (двоичных) переменных.

Бинарными называются переменные, принимающие одно из двух значений: «истина» и «ложь», которые обозначаются соответственно 1 и 0.

Рассмотрим набор переменных X_1, \dots, X_n со значениями

$$x = (x_1, \dots, x_n), \quad x_j \in \{0, 1\}, \quad x \in X = \{0, 1\}^n.$$

Если считать переменные независимыми, то для задания вероятностной модели требуется n параметров, а в общем случае модель совместного распределения имеет $2^n - 1$ параметров. Если объём выборки существенно больше 2^n , то оправдано оценивать совместное распределение, если объём выборки сравним с n , то имеет смысл использовать модель независимых переменных, даже если нет оснований ожидать, что переменные действительно независимы.

Сейчас рассмотрим модель, которую уместно использовать, когда объём выборки много больше n , но меньше 2^n . Данная модель связана с разложением совместного распределения в ряд Бахадура.

Идея разложения в ряд Бахадура состоит в том, чтобы, начиная с приближения независимых переменных, последовательно учитывать парные зависимости переменных, зависимости в тройках, четвёрках и т. д. Разложение в ряд Бахадура выглядит следующим образом

$$P(x) = P^1(x) \sum_{j=1}^n q_j(x).$$

Здесь $P^1(x)$ – значение совместной вероятности в предположении независимости переменных, т. е.

$$P^1(x) = \prod_{j=1}^n P(x_j).$$

Единица сверху — верхний индекс.

Остальные компоненты определяются как

$$q_1(x) \equiv 1, \quad q_2(x) = \sum_{j=1}^n \sum_{k=j+1}^n \rho_{jk} z_j z_k,$$

$$q_3(x) = \sum_{j=1}^n \sum_{k=j+1}^n \sum_{l=k+1}^n \rho_{jkl} z_j z_k z_l, \quad \dots$$

В этих выражениях использованы обозначения

$$z_j = \frac{x_j - p_j}{\sqrt{p_j(1 - p_j)}},$$

где $p_j = P(x_j = 1)$, а

$$\rho_{jk} = \sum_{x \in X} z_j z_k P(x), \quad \rho_{jkl} = \sum_{x \in X} z_j z_k z_l P(x), \quad \dots$$

Распределения

$$P^M(x) = P^1(x) \sum_{j=1}^M q_j(x)$$

представляют собой частичные суммы ряда Бахадура, при этом $P^2(x)$ учитывает только парные зависимости переменных, $P^3(x)$ учитывает парные зависимости и зависимости в тройках. Полная сумма ряда совпадает с исходным распределением $P(x)$.

Частичные суммы ряда Бахадура могут использоваться в качестве приближений для $P(x)$.

Чем больше членов ряда используется, тем больше параметров требуется оценить по выборке и тем выше сложность модели.

Следует выбирать сложность модели адекватной объёму выборки. Грубой рекомендацией здесь будет выбирать такую модель, чтобы число оцениваемых параметров было хотя бы в несколько раз меньше объёма выборки. Более точно выбрать сложность модели можно с помощью метода статистического моделирования.

Разложение в ряд Бахадура вполне может использоваться для решения задачи классификации в пространстве бинарных переменных.

Вместе с тем подобные задачи встречаются довольно редко, поэтому описанное разложение относительно мало распространено на практике.

Тем не менее знакомство с ним очень полезно для понимания механизмов, за счёт которых одни методы построения решающих функций оказываются эффективнее других. Мы ещё вернёмся к этому разложению при выяснении причин эффективности бустинга.

2.4.4 Свойства метода

Характеризуя «наивный» байесовский классификатор, можно сделать следующие замечания.

- ✧ Предположение независимости переменных эквивалентно аддитивности в пространстве преобразованных переменных.
- ✧ Полученное преобразование является разумным вариантом target encoding, может использоваться и для вещественных переменных.
- ✧ Наивный байесовский классификатор является частным случаем логистической регрессии с использованием target encoding.
- ✧ Предположение о (строгой) независимости переменных обычно оказывается неоправданно сильным, однако в ослабленном виде гипотеза независимости очень полезна.

Поскольку в реальных задачах предполагать независимость переменных, как правило, нет оснований, использовать «наивный» байесовский классификатор редко бывает целесообразным. Однако попробовать применить «ослабленный» вариант метода в виде логистической регрессии с target encoding — почти всегда оправдано. Если при этом коэффициенты регрессии окажутся близки к единице, то это может служить свидетельством независимости переменных (и тогда можно применить собственно наивный байесовский классификатор).

Отметим важный факт. Почти для всех методов машинного обучения имеет место правило, что чем больше переменных, тем больший объём выборки требуется для обучения (из-за чего даже появился термин «проклятие размерности»).

Исключением из этого правила является случай условно независимых переменных. Здесь всё наоборот: чем больше переменных, тем большую точность прогноза можно получить при том же объёме выборки. Это похоже на то, как привлечение множества независимых мнений позволяет сделать более точные выводы.

2.5 Метод опорных векторов

Метод был разработан ещё в 60-е годы (В. Н. Вапник и др.) и изначально назывался методом обобщённого портрета.

В 90-е годы метод был усовершенствован и стал называться машиной опорных векторов (support vector machine, SVM).

Как и дискриминант Фишера, метод опорных векторов основан на поиске такого направления в пространстве переменных, при проецировании на которое объекты разных классов оказались бы в наибольшей степени отделены друг от друга. Но если критерий Фишера отделение классов понимает как нормированное расстояние между средними, то SVM в качестве расстояния между множествами использует расстояние между ближайшими (крайними) точками. Направление выбирается так, чтобы максимально отдалить объекты от разделяющей поверхности.

2.5.1 Постановка задачи классификации

Введём необходимые обозначения.

Пусть $X = R^n$ – пространство значений прогнозирующих переменных, $Y = \{-1, 1\}$ – значения прогнозируемой переменной, $D = X \times Y$.

Решающая функция (алгоритм классификации) есть отображение

$$f : X \rightarrow Y.$$

Пусть $S = ((x^i, y^i) \in D \mid i = \overline{1, N})$ – случайная независимая выборка, $S \in D^N$.

Метод построения решающих функций $Q : D^N \rightarrow \Lambda$ есть отображение множества выборок во множество решений (Λ – заданный класс решающих функций).

Чтобы задача имела смысл, нужно определить качество решения. Для этого вводится понятие целевой функции потерь $L(\cdot, \cdot)$, которая характеризует степень несоответствия прогноза и истинного класса.

В данном параграфе будем понимать потери как несовпадение прогнозируемого и истинного класса, т. е. $L(y, f(x)) = I(y \neq f(x))$, где $I(\cdot)$ – индикаторная функция (возвращает 1, если выражение истинно, иначе – 0).

Эмпирическим риском называются средние потери на обучающей выборке

$$\tilde{R}(S, f) = \frac{1}{N} \sum_{i=1}^N L(y^i, f(x^i)).$$

Решающую функцию будем искать в форме линейного порогового классификатора

$$f(x) = \text{sign}(wx - w_0).$$

Требуется найти вектор w и скаляр w_0 , минимизирующие эмпирический риск при дополнительных требованиях.

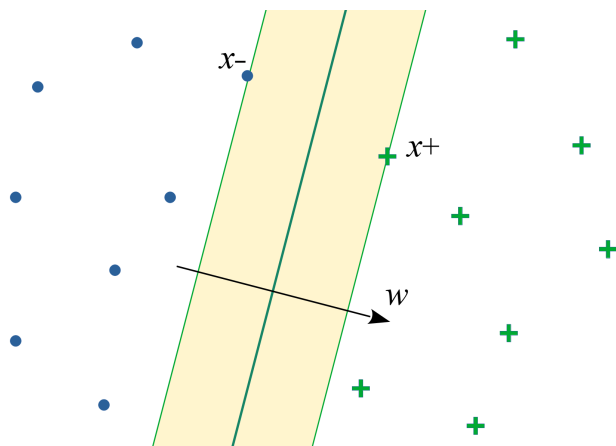


Рис. 2.5. Иллюстрация идеи метода SVM

2.5.2 Линейно разделимая выборка

Предположим, что выборка оказалась такова, что существует гиперплоскость, которая разделяет классы. Это эквивалентно тому, что существует линейный классификатор с нулевым значением эмпирического риска.

Если существует гиперплоскость, безошибочно разделяющая выборку, то почти всегда её можно немного «подвинуть», так что выборка всё ещё разделится безошибочно. Получается бесконечное (континуальное) множество решений с нулевым эмпирическим риском. Нужно выбрать из этого множества «наилучший» вариант.

Критерием для такого выбора будет ширина разделяющей полосы. Если мы спроецировали точки на нормаль к гиперплоскости, то ширина разделяющей полосы будет равна наименьшему расстоянию между проекциями точек разных классов. Пример построения разделяющей полосы для некоторого направления w показан на рис. 2.5.

Заметим, что для задания направления можно использовать вектор любой длины. Таким образом, длина вектора w — это

«лишний» свободный параметр, от которого нужно избавиться, чтобы решение было единственным.

Наиболее очевидное решение — это рассматривать векторы единичной длины, но такой вариант не позволит эффективно решать задачу оптимизации. Удобнее оказывается использовать следующее условие нормировки

$$\min_{(x^i, y^i) \in S} y^i(x^i w - w_0) = 1. \quad (2.14)$$

Легко убедиться, что для любого направления w' , вдоль которого выборка классифицируется безошибочно, можно выбрать такие числа (скаляры) a и w_0 , что при $w = aw'$ условие (2.14) будет выполнено.

Минимум в (2.14) достигается на каких-то выборочных точках, которые назовём граничными. При этом ширина разделяющей полосы оказывается максимальной, если в числе граничных точек оказываются представители обоих классов. Выберем два таких представителя: x_+ и x_- .

Разделяющая полоса в некотором смысле «опирается» на эти граничные точки, которые потому и называются *опорными векторами*.

При дополнительном требовании максимизации ширины разделяющей полосы условие (2.14) определяет однозначный выбор нормы направляющего вектора и константы w_0 .

Для граничных точек условие (2.14) превращается в

$$x_+ w - w_0 = 1, \quad -(x_- w - w_0) = 1.$$

Ширину разделяющей полосы легко выразить через координаты граничных точек

$$(x_+ - x_-) \cdot \frac{w}{|w|} = \frac{(w_0 + 1) - (w_0 - 1)}{|w|} = \frac{2}{|w|}.$$

Получается, что максимизация ширины разделяющей полосы при условии (2.14) эквивалентна минимизации нормы вектора w , или, что то же самое, минимизации его квадрата.

В результате поиск вектора w и скаляра w_0 сводится к решению следующей оптимизационной задачи

$$\begin{cases} w^2 \rightarrow \min, \\ w, w_0 \\ y^i(x^i w - w_0) \geq 1, \quad i = 1, \dots, N. \end{cases} \quad (2.15)$$

Граничные условия в этой задаче эквивалентны условию (2.14) при дополнительном требовании минимизации нормы w .

Данная задача относится к хорошо известному классу задач квадратичной оптимизации. Для решения таких задач существуют готовые алгоритмы. Оптимизационную задачу можно решать методом активных ограничений (incremental active set method, INCAS), частным случаем которого является симплекс-метод.

2.5.3 Линейно неразделимая выборка

Очевидно, что линейная делимость является очень обременительным ограничением, которое необходимо снять, чтобы метод имел приемлемую область применения.

Если выборка не является линейно делимой, то не существует вектора w , который бы удовлетворял всем ограничениям задачи (2.15), т. е. ограничения получаются несовместными и задача не имеет решения.

Получается, что ограничения нужно ослабить. Для этого введём величины ξ_i , которые будут задавать «допустимую степень нарушения» каждого ограничения. Получим следующую задачу квадратичного программирования

$$\begin{cases} \frac{w^2}{2} + C \sum_{i=1}^N \xi_i \rightarrow \min, \\ w, w_0, \xi \\ y^i(x^i w - w_0) \geq 1 - \xi_i, \quad i = 1, \dots, N, \\ \xi_i \geq 0, \quad i = 1, \dots, N, \end{cases} \quad (2.16)$$

где $C > 0$ – параметр. «Степень нарушения» не может быть отрицательной, поэтому накладываем условия $\xi_i \geq 0$.

Очевидно, что чем меньше ξ_i , тем лучше решение, поэтому сумму всех ξ_i мы добавили в целевую функцию, которая минимизируется. Чтобы управлять вкладом каждого слагаемого, введён параметр C .

Заметим, что для линейно разделимой выборки решение уже не обязательно даст нулевой эмпирический риск. При не очень больших C может оказаться выгоднее допустить нарушение разделяющей полосы или даже ошибочную классификацию некоторых объектов, но за счёт этого построить полосу большей ширины.

Задача (2.16) приближается к задаче для линейно разделимой выборки, когда $C \rightarrow \infty$.

Константу C обычно выбирают по критерию скользящего контроля.

2.5.4 Двойственная задача

Метод опорных векторов является одним из самых красивых в математическом плане методов машинного обучения. Наиболее изящные его возможности получаются с помощью перехода к двойственной задаче.

Чтобы разобраться в этой технике, попытаемся кратко и наглядно изложить суть некоторых идей математического программирования.

Рассмотрим простейшую постановку задачи на нахождение условного экстремума

$$\begin{cases} f(x) \rightarrow \min_x, \\ \psi(x) = 0. \end{cases}$$

Эту задачу можно проиллюстрировать наглядно. Пусть имеется некоторый рельеф местности и некоторая траектория движения (тропинка). Требуется найти на этой тропинке наивысшую или самую низкую точку (рис. 2.6). Очевидно, что в таких точках тропинка будет идти параллельно склону, т. е. параллельность

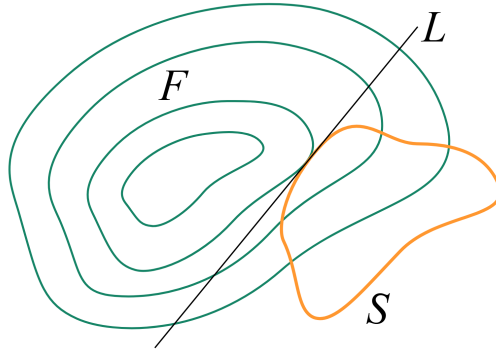


Рис. 2.6. Иллюстрация идеи множителей Лагранжа

склону (или касание тропинкой некоторой линии уровня) — это необходимое условие экстремума.

Математически параллельность склону означает параллельность векторов градиента высоты и нормали к траектории, т. е.

$$\frac{\partial f(x)}{\partial x} = \lambda \frac{\partial \psi(x)}{\partial x}.$$

Здесь λ имеет смысл коэффициента пропорциональности двух векторов.

После переноса в левую часть и выноса производной получаем в скобках функцию Лагранжа

$$\frac{\partial(f(x) - \lambda\psi(x))}{\partial x} = \frac{\partial \mathcal{L}(x)}{\partial x}.$$

Пусть теперь в задаче на нахождение экстремума ограничение задано не равенством, а неравенством

$$\begin{cases} f(x) \rightarrow \min_x, \\ \psi(x) \geq 0. \end{cases}$$

Тогда решение (искомый минимум) находится либо внутри области, либо на её границе. Вторым случаем мы только что рассмотрели, а внутри области необходимым условием экстремума является просто равенство градиента нулю.

Эти два случая можно свести в одну систему, которые называются условиями Каруша–Куна–Таккера. Необходимые условия экстремума

$$\begin{cases} \frac{\partial f(x)}{\partial x} = \lambda \frac{\partial \psi(x)}{\partial x}, \\ \lambda \psi(x) = 0, \\ \lambda \geq 0. \end{cases}$$

Второе условие называют условием дополняющей нежёсткости. Его смысл очень простой. Для его выполнения один из множителей должен быть нулевым. Если $\lambda = 0$, то первое уравнение (условие стационарности) превращается в равенство градиента нулю, и мы ищем безусловный экстремум. Если $\lambda \neq 0$, получаем условный экстремум.

Условие неотрицательности для λ означает, что если минимум находится на границе области, то градиент в этой точке должен быть направлен внутрь области (т.е. функция должна уменьшаться при движении «наружу» — иначе минимум окажется внутри).

Применим данный аппарат к задаче (2.16). Функция Лагранжа для SVM есть

$$\begin{aligned} \mathfrak{L}(w, w_0, \xi; \lambda, \eta) &= \\ &= \frac{w^2}{2} + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \lambda_i (y^i (x^i w - w_0) - 1 + \xi_i) - \sum_{i=1}^N \xi_i \eta_i. \end{aligned}$$

После преобразований

$$\begin{aligned} \mathfrak{L}(w, w_0, \xi; \lambda, \eta) &= \\ &= \frac{w^2}{2} - \sum_{i=1}^N \lambda_i (y^i (x^i w - w_0) - 1) - \sum_{i=1}^N \xi_i (\lambda_i + \eta_i - C). \end{aligned}$$

Дифференцируем (находим условия стационарности)

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0, & w = \sum_{i=1}^N \lambda_i y^i x^i, \\ \frac{\partial \mathcal{L}}{\partial w_0} = 0, & \sum_{i=1}^N \lambda_i y^i = 0, \\ \frac{\partial \mathcal{L}}{\partial \xi_i} = 0, & \eta_i + \lambda_i = C, \quad i = 1, \dots, N. \end{cases}$$

Выразим всё через λ_i

$$-\mathcal{L}(\lambda) = \sum_{i=1}^N \lambda_i + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y^i y^j x^i x^j.$$

В результате получаем двойственную задачу

$$\begin{cases} \sum_{i=1}^N \lambda_i + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y^i y^j x^i x^j \rightarrow \min_{\lambda}, \\ 0 \leq \lambda_i \leq C, \quad i = 1, \dots, N, \\ \sum_{i=1}^N \lambda_i y^i = 0. \end{cases} \quad (2.17)$$

Приведённые выкладки являются слишком краткими и поверхностными, чтобы по ним полноценно разобраться в выводе, поэтому рекомендуется углубленно изучить тему в специализированных источниках.

Для понимания дальнейших рассуждений достаточно просто принять как факт, что задача (2.17) равносильна (эквивалентна) задаче (2.16), поэтому для реализации метода SVM можно решать любую из них.

Точки, для которых $\lambda_i > 0$, называются *опорными векторами*.

2.5.5 Ядра и спрямляющие пространства

Задача (2.17) примечательна тем, что вектора x^i входят в выражения только через скалярные произведения друг с другом. Поэтому если мы знаем скалярные произведения, то непосредственно компоненты векторов нам уже не нужны.

Поскольку $w = \sum_{i=1}^N \lambda_i y^i x^i$, итоговый классификатор имеет вид

$$f(x) = \text{sign} \left(\sum_{i=1}^N \lambda_i y^i x^i - w_0 \right). \quad (2.18)$$

Здесь тоже используются только скалярные произведения, но не компоненты векторов.

Получается, что если мы знаем скалярные произведения между векторами, описывающими объекты, то сама таблица данных не нужна. Это позволяет решать так называемую задачу беспризнаковой классификации. Её мы обсудим чуть позже, поскольку сначала нужно ввести понятие функции ядра.

Идея ядра возникла из желания снять ограничение линейности решения.

Есть много задач, где решающая функция должна быть нелинейной. При этом функция может быть простой. Например, объекты первого класса находятся все в некотором круге, а другой класс — снаружи круга. Линейной функцией в исходном пространстве такие данные не разделить.

Однако есть такой типовой приём как *feature engineering*, который заключается в конструировании новых переменных как функций от исходных. Если мы добавим к исходным переменным их квадраты, то линейная функция в новом пространстве будет квадратичной функцией исходных переменных (в частности, может отделить круг).

Добавление новых переменных есть переход от исходного пространства X к новому пространству H с помощью некоторого преобразования $\psi : X \rightarrow H$. Пространство H называют *спрямляющим*.

Теперь отметим простой, но важный факт: скалярное произведение в спрямляющем пространстве есть некоторая функция от исходных переменных

$$K(x, x') = \psi(x) \cdot \psi(x').$$

Такие функции будем называть *ядрами*.

Функция ядра (или просто ядро) — это такая функция, заданная на парах векторов, для которой существует спрямляющее пространство, в котором эта функция соответствует скалярному произведению.

Можно вообще не вводить новое пространство (достаточно знать, что оно существует), а сразу задать ядро K . Такой приём называется *kernel trick*.

Пример работы SVM с ядром приведён на рис. 2.8 на стр. 76.

Приём с ядрами, как минимум, позволяет экономить память (не нужно хранить новые переменные). Кроме того, некоторые ядра соответствуют спрямляющим пространствам бесконечной размерности (и тут явно ввести переменные не получится).

Заметим, что далеко не любая симметричная функция, заданная на парах векторов, является ядром. Существует ряд требований (условия Мерсера), которым она должна удовлетворять.

2.5.6 Беспознаковая классификация

Существует довольно большой класс задач, где объекты не имеют признакового описания. В качестве примеров можно привести музыкальные произведения, картины, тексты.

Конечно, для таких данных существуют способы построить признаковые пространства (например, с помощью нейросетей), но это требует большого объёма данных.

Однако в ряде задач мы можем каким-то образом (например, от экспертов) получить численные оценки схожести (или непохожести) объектов друг на друга. А непохожесть — это по смыслу «расстояние» (только аксиомы метрики могут не выполняться).

Вспомним, что самая известная метрика — Евклида — определяется на основе скалярного произведения.

Эти соображения наталкивают на идею построить по экспертным оценкам схожести метрику, а по метрике подобрать скалярное произведение, которого достаточно для применения SVM. Однако не любая непохожесть — метрика, и не в любом метрическом

пространстве можно ввести скалярное произведение.

С другой стороны, именно скалярное произведение нам не требуется, поскольку достаточно функции ядра.

В результате применение метода SVM для задачи беспризнаковой классификации выглядит следующим образом.

Берётся матрица экспертных оценок «непохожести» объектов (если эксперт оценивал «похожесть», то превращаем её в различие).

Эта матрица корректируется так, чтобы выполнялись требования к функции ядра.

На скорректированной матрице запускается SVM.

2.5.7 Задача безусловной оптимизации

Нужно отметить очень важный факт, а именно: задача (2.16) эквивалентна безусловной минимизации (по w, w_0) функции

$$\frac{w^2}{2} + C \sum_{i=1}^N \text{relu}(1 - y^i(x^i w - w_0)), \quad (2.19)$$

где

$$\text{relu}(z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases}.$$

Функция relu широко используется в нейросетях.

Может показаться, что решать задачу безусловной оптимизации должно быть проще, чем задачу квадратичного программирования. Зачастую это так, но не в данном случае. Функция relu имеет излом, что приведёт к затруднениям при использовании градиентных методов. Кроме того, такая формулировка не позволит использовать приём с ядром.

Тем не менее, постановка (2.19) важна.

Во-первых, она показывает, что метод опорных векторов может быть сформулирован в форме минимизации потерь.

Во-вторых, такая форма позволяет увидеть сходство SVM и логистической регрессии. Для этого достаточно сравнить функции потерь.

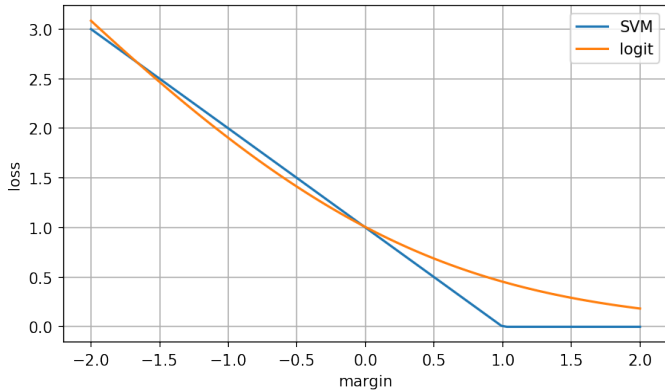


Рис. 2.7. Функции потерь для SVM и логистической регрессии

Оба метода эквивалентны безусловной минимизации соответствующей функции потерь с регуляризацией.

На рис. 2.7 приведены функции потерь для SVM и логистической регрессии (из формулы (2.9)). В качестве аргумента функций потерь взят так называемый отступ (margin), а именно выражение $y^i(x^i w - w_0)$. Видим, что функции разные по форме, но близки по значениям. Из этого следует, что и результаты этих методов должны быть (при прочих равных) похожи.

Такое сходство методов, в частности, даёт основания перенести из SVM вид регуляризатора, а именно положить $\kappa(w^2) = \frac{w^2}{2C}$ в (2.9).

2.5.8 Свойства метода

Подытожим основные свойства метода опорных векторов.

- ✧ Метод сводится к эффективно решаемой задаче квадратичного программирования.
- ✧ Разреженность (решение определяется только опорными векторами).
- ✧ Возможно обобщение введением функции ядра.

- ✧ Возможна беспризнаковая классификация.
- ✧ Низкая эффективность при сильно перекрывающихся распределениях классов.
- ✧ Неустойчив к выбросам. Целесообразна фильтрация как для логистической регрессии.

Существуют также вопросы, на которые пока нет точного ответа:

- ✧ как соотносится SVM с метрическими методами;
- ✧ является ли разреженность достоинством;
- ✧ как выбирать ядро;
- ✧ как применить kernel trick в логистической регрессии.

Пик популярности метода пришёлся на 90-е годы. Однако и в настоящее время метод остаётся вполне актуальным.

На большинстве задач метод показывает достаточно высокую точность. Чаще всего, эта точность несколько ниже, чем у бустинга, но несущественно. При этом в каких-то задачах SVM может оказаться даже лучшим по точности.

Однако, даже если точность SVM оказалась несколько ниже, чем у бустинга, может быть целесообразным использовать именно его, поскольку линейное решение имеет очень важное достоинство: простоту и интерпретируемость (к сожалению, это не относится к ядрам). В некоторых задачах это даже важнее точности.

2.6 Характеристика линейных методов

Между линейными методами мы установили достаточно неочевидную общность, т. е. сходство.

- ✧ Дискриминант Фишера не требует вероятностных предположений, но по результату почти совпадает с линейным дискриминантом, основанным на нормальности.
- ✧ Наивный байесовский классификатор является частным случаем логистической регрессии (дополненной заменой переменных).

- ✧ Метод SVM оказывается близок логистической регрессии ввиду сходства функций потерь.

В целом, линейные модели являются очень широким и важным классом методов машинного обучения и используются как самостоятельно, так и в составе других методов (например, в нейросетях в основе лежит как раз линейная функция, а многие функции активации также заимствованы из линейных моделей).

2.7 Примеры кода

2.7.1 Визуализация решений

Первый пример демонстрирует визуализацию решающих функций, построенных разными методами на первых двух переменных таблицы Iris.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, svm, tree
from sklearn.discriminant_analysis import (
    LinearDiscriminantAnalysis,
    QuadraticDiscriminantAnalysis)
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import (
    GradientBoostingClassifier)

def make_meshgrid(x, y, padding = 1.0, N = 200):
    x_min = x.min() - padding
    x_max = x.max() + padding
    y_min = y.min() - padding
    y_max = y.max() + padding
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, N),
                          np.linspace(y_min, y_max, N))

    return xx, yy

models = {
    'SVC with linear kernel'
        : svm.SVC(kernel='linear', C=1.0),
    'LinearSVC (linear kernel)'
```

```

        : svm.LinearSVC(C=1.0, max_iter=10000),
'LogisticRegression'
        : LogisticRegression(),
'LinearDiscriminantAnalysis'
        : LinearDiscriminantAnalysis(),
'naive Gauss'
        : GaussianNB(),
'SVC with RBF kernel'
        : svm.SVC(kernel='rbf', gamma=0.7, C=1.0),
'SVC with polynomial (degree 2) kernel'
        : svm.SVC(kernel='poly', degree=2),
'QuadraticDiscriminantAnalysis'
        : QuadraticDiscriminantAnalysis(),
'DecisionTreeClassifier'
        : tree.DecisionTreeClassifier(),
'GradientBoosting'
        : GradientBoostingClassifier(
            n_estimators=200, learning_rate=0.1,
            max_depth=1, random_state=0)
}

iris = datasets.load_iris()

X = iris.data[iris.target!=0, :2]
y = iris.target[iris.target!=0]

models = {title : clf.fit(X, y) for
          title, clf in models.items()}

plot_size = 15
cmap = plt.cm.winter
n_rows = (len(models)+1)//2
fig, sub = plt.subplots(n_rows, 2,
                        figsize=(plot_size, plot_size*n_rows//2))
plt.subplots_adjust(wspace=0.15, hspace=0.2)

X0, X1 = X[:, 0], X[:, 1]
xx, yy = make_meshgrid(X0, X1)

for (title, clf), ax in zip(models.items(),
                             sub.flatten()):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    zz = Z.reshape(xx.shape)

```

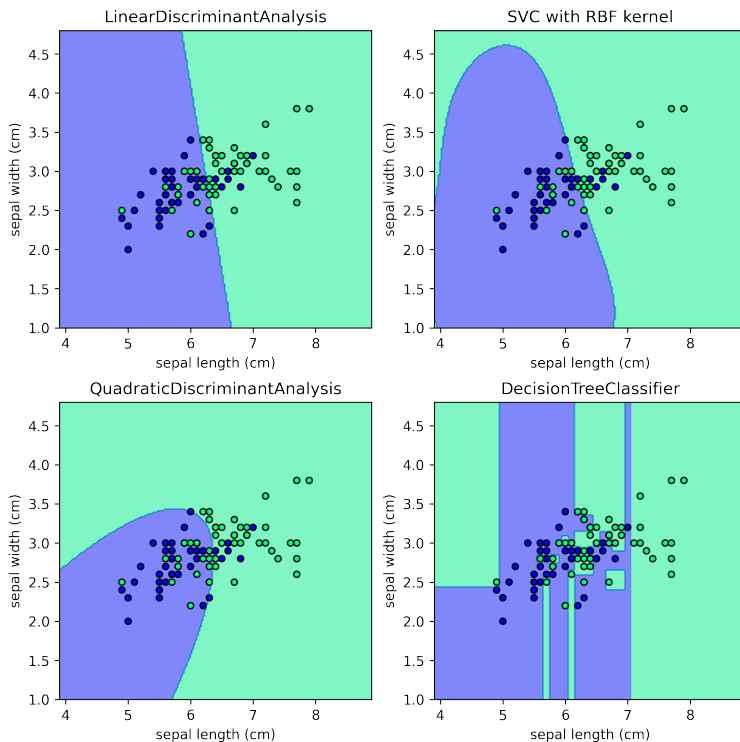


Рис. 2.8. Визуализация прогнозов разными методами

```
ax.contourf(xx, yy, zz, cmap=cmap, alpha=0.5)

ax.scatter(X0, X1, c=y, cmap=cmap,
           s=20, edgecolors='k')
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xlabel(iris['feature_names'][0])
ax.set_ylabel(iris['feature_names'][1])
ax.set_title(title)

plt.show()
```

Результат выполнения кода (частично) приведён на рис. 2.8. Код может использоваться в качестве основы для дальнейших

модификаций. В него можно легко добавить вызов других методов, для чего достаточно модифицировать словарь `models`.

2.7.2 Кодирование категориальных переменных и наивный байесовский классификатор

Следующий код демонстрирует один из наиболее эффективных способов кодирования категориальных переменных, а именно `target encoding` с логистической трансформацией.

На основе полученных значений наивный байесовский классификатор реализуется простым суммированием.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

```
# Загружаем данные из репозитория:
```

```
df = pd.read_csv(
    'https://archive.ics.uci.edu/ml/' +
    'machine-learning-databases/' +
    'mushroom/agaricus-lepiota.data', header=None)
```

```
# Первый столбец - целевая переменная
```

```
Y = (df[0]== 'p').astype(int)
```

```
# Вычисляем частоту первого класса:
```

```
p0 = Y.sum()/Y.shape[0]
z0 = np.log(p0) - np.log(1.0 - p0)
print('p0, z0:', p0, z0)
```

```
# Выполняем target encoding:
```

```
X = pd.DataFrame(index=df.index)
for i in df.columns[1:]:
    f = df.groupby(i)[0].value_counts(
        ).unstack().fillna(0)
    f['prob'] = (f['p']+1)/(f['p']+f['e']+2)
    f['logit'] = np.log(f['prob']) - np.log(1.0
        - f['prob']) - z0
    X[i] = df[i].map(lambda x: f.at[x, 'logit'])
```

```
# Прогноз в предположении независимости:
```

```

P = X.sum(axis=1) + z0
P = 1/(1+np.exp(-P))
print('Naive Bayes accuracy:',
      ((P>0.5)==(Y==1)).sum() / Y.shape[0])

```

#Логистическая регрессия:

```

clf = LogisticRegression().fit(X,Y)
P1 = clf.predict(X)
print('LogisticRegression accuracy:',
      ((P1>0.5)==(Y==1)).sum() / Y.shape[0])

```

В результате выполнения кода будет выведено.

```

p0, z0: 0.482028557 -0.07191675
Naive Bayes accuracy: 0.956548498
LogisticRegression accuracy: 1.0

```

Настоятельно рекомендуется запустить все предложенные примеры и понять работу кода, в том числе внося в него изменения и наблюдая результат.

Контрольные вопросы

1. Какая польза от предположений о форме распределений в задачах машинного обучения? Имеет ли смысл проверять гипотезы о форме распределений?
2. Что такое байесовский уровень ошибки и байесовский классификатор? Можно ли определить эти понятия, не используя формулу Байеса?
3. В чём идея дискриминанта Фишера?
4. В чём идея логистической регрессии? Какова функция потерь в методе логистической регрессии? Зачем нужна регуляризация?
5. Что такое «выбросы» и как с ними бороться?
6. Какое предположение лежит в основе наивного байесовского классификатора? Как метод использует формулу Байеса?

7. Чем важен наивный байесовский классификатор?
8. В чём состоит основная идея метода опорных векторов? К каким задачам квадратичного программирования сводится метод?
9. В чём заключается идея kernel trick?
10. За счёт чего SVM способен решать задачи беспризнаковой классификации?
11. К какой задаче безусловной оптимизации сводится метод опорных векторов?
12. Как реализуется target encoding?

Глава 3

Логические методы классификации

Логические методы — широко используемый класс методов.

Основные варианты:

- ✧ поиск закономерностей, решающие списки;
- ✧ решающие деревья.

В настоящее время из логических методов наиболее известны деревья решений, которые используются, в том числе, как базовый классификатор для бустинга и случайного леса. Но для полноты картины рассмотрение стоит начать с понятия логической закономерности.

3.1 Логические закономерности

3.1.1 Понятие закономерности

Слово «закономерность» обычно используется как антоним слова «случайность».

Однако мы будем термин «случайность» использовать исключительно в определённом математически строгом смысле, а именно: будем называть случайным то, чему мы приписали вероят-

ность. Поэтому «закономерность» будем полагать скорее антонимом слову «хаотичность».

Итак, под закономерностью в широком смысле будем понимать какую-то особенность в данных, которая обладает статистической устойчивостью.

При этом мы предполагаем, что закономерности нам понадобятся в конечном счёте для прогнозирования класса объекта, поэтому нас интересуют такие особенности в данных, которые можно использовать для прогнозирования.

В результате *закономерностью* будем называть заданной формы область в пространстве переменных, в которой объектов одного класса «много», а других классов «мало» (смысл этих понятий уточним ниже).

Первым делом нужно выбрать форму областей.

Очевидно, что форма должна быть простой, поскольку областью достаточно сложной формы можно отделить любое подмножество точек выборки. Самая простая форма — многомерный интервал (гиперпараллелепипед). Исторически рассматривались и другие формы (в частности, сферы), но мы ограничимся интервалами.

Наверное, уже напрашивается вопрос: где здесь «логика», т. е. почему мы называем закономерности логическими, а не, к примеру, «геометрическими», раз уж речь идёт о гиперпараллелепипедах.

Дело в том, попадание точки в область удобно описывать в терминах предикатов.

Собственно, *предикат* — это такая функция, которая в качестве аргумента принимает точку, а в качестве результата возвращает значение «истина», если точка попала в заданную область, и «ложь» — иначе.

Предикаты удобны тем, что для них можно использовать логические операции (например, конъюнкцию), а также предикаты удобно использовать в узлах дерева.

Для примера приведём логические закономерности для задачи Iris. На рис. 3.1 приведено по одной закономерности для каждого

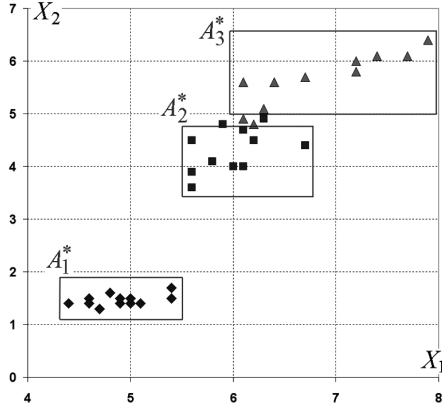


Рис. 3.1. Логические закономерности для задачи Iris

класса. Если с первым классом вопросов не возникает, то интервал для второго класса не захватывает два объекта этого класса. Мы можем интервал расширить, включив эти объекты, но тогда в него попадут также точки третьего класса. Вопрос — какой вариант лучше. Чтобы на него ответить, нужно формализовать критерий качества.

Пусть X — пространство значений прогнозирующих переменных, $Y = \{1, 2, \dots\}$ — прогнозируемая переменная, $\varphi : X \rightarrow \{0, 1\}$ — предикат.

Классы могут нумероваться (обозначаться) произвольным образом, но будем считать, что среди них есть класс, обозначенный как первый.

Поскольку закономерности будем искать для каждого класса независимо, нам достаточно рассмотреть эту процедуру для первого класса.

Пусть $S = ((x^i, y^i) \mid i = \overline{1, N})$ — выборка объектов, из которых M принадлежит 1-му классу, n — число¹ точек (выборки), на ко-

¹В других местах мы через n обозначали и будем обозначать количество переменных (размерность пространства), использование n в ином значении ограничено текущим параграфом.

торых предикат φ истинный, m – сколько из них 1-го класса.

Закономерность тем «лучше» («сильнее»), чем больше она захватывает объектов своего класса и чем меньше чужих. Это можно записать как

$$a = \frac{m}{M} \rightarrow \max, \quad b = \frac{n - m}{n} \rightarrow \min.$$

Обе величины a и b характеризуют качество закономерности. Необходимо свести их в одно выражение, т. е. задать единый критерий качества. Это можно сделать множеством способов, например

$$\frac{a}{b} \rightarrow \max, \quad a - b \rightarrow \max, \quad \sqrt{a} - \sqrt{b} \rightarrow \max.$$

Нужно понять, какое из выражений более «правильное» и можно ли получить подобное выражение не эвристически, а из рациональных рассуждений.

3.1.2 Критерии качества закономерности

Естественно считать закономерность тем более «сильной», чем менее она «случайна». Этот тезис интуитивно вполне понятен, но мы только что договорились не использовать термин «случайность» в том смысле, который тут вкладывается. Изменим формулировку, заодно сделав её строгой.

Будем считать закономерность тем более *сильной*, чем в большей степени (на меньшем уровне значимости) она отвергает «нулевую гипотезу».

В задаче классификации естественным вариантом *нулевой гипотезы* является предположение, что все классы имеют одинаковое распределение. Эквивалентная формулировка: вероятность попадания объекта в область не зависит от класса, к которому он принадлежит.

Идея статистического критерия (согласия) для проверки гипотез заключается в следующем. Выбирается некоторое событие, вероятность которого при выполнении гипотезы очень низка (порог, ниже которого вероятность считается низкой, называется уровнем

значимости). Если это событие всё же происходит, то гипотеза отвергается.

Пусть в некоторую область пространства попало n объектов. Какова вероятность, что среди них окажется m объектов первого класса?

Можно привести практическую иллюстрацию. Выпущена лотерея на N билетов, из которых M выигрышных. Купили n билетов. Какова вероятность, что из них будет m выигрышных?

Это простая типовая задача по теории вероятностей, ответ в которой выражается формулой гипергеометрического распределения. Вероятность при отборе n объектов получить m из них 1-го класса равна

$$P(m) = \frac{C_M^m C_{N-M}^{n-m}}{C_N^n} = \frac{C_n^m C_{N-n}^{M-m}}{C_N^M}.$$

Однако для выражения качества закономерности следует использовать не $P(m)$, а сумму

$$U(m) = \sum_{m'=m}^M P(m'),$$

которая есть вероятность того, что в области окажется не менее m объектов 1-го класса.

Причина, по которой нужно суммировать $P(m)$, объясняется в курсах математической статистики в теме о критериях согласия. В данном учебнике аналогичное объяснение встретится в подразделе 6.2.2, где приводится оценка доверительного интервала по контрольной выборке, поэтому если данный момент сейчас непонятен, к нему можно вернуться позже.

Содержательно $U(m)$ есть достигаемый уровень значимости для проверки нулевой гипотезы. Чем он ниже, тем «сильнее» отвергается гипотеза.

Заметим, что $U(m)$ не зависит от сложности закономерности и размерности пространства, поскольку посчитана для априорно выбранной закономерности.

Если мы ищем закономерность, целенаправленно минимизируя $U(m)$, то полученный минимум уже не годится в качестве достигаемого уровня значимости для проверки нулевой гипотезы. Требуется поправка, которая уже зависит от сложности предиката. Для оценки этого эффекта потребуются рассуждения в духе принципа равномерной сходимости.

Вероятность $P(m)$ можно вычислять приближённо.

Введём обозначение

$$H(p) = -p \ln p - (1 - p) \ln(1 - p).$$

Данная величина есть энтропия (часто используют логарифм по основанию 2, но для удобства возьмём натуральный).

Вспомним формулу Стирлинга:

$$\ln(k!) \approx k \ln k - k + \frac{1}{2} \ln(2k\pi) + \frac{1}{12k} - \frac{1}{360k^2}.$$

Воспользовавшись первыми двумя членами данного разложения, оценим логарифм биномиального коэффициента

$$\begin{aligned} \ln C_N^n &= \ln \left(\frac{N!}{n!(N-n)!} \right) \approx \\ &\approx N \ln N - N - (n \ln n - n) - ((N-n) \ln(N-n) - (N-n)) = \\ &= -n(\ln n - \ln N) - (N-n)(\ln(N-n) - \ln N) = NH \left(\frac{n}{N} \right). \end{aligned}$$

Оказывается, биномиальный коэффициент просто выражается через энтропию. Заметим, что вторые члены приближения попросту сократились.

Применим это выражение для оценки вероятности закономерности. Получим

$$\ln P(m) \approx -NG,$$

где

$$G = H \left(\frac{M}{N} \right) - \frac{n}{N} \cdot H \left(\frac{m}{n} \right) - \frac{N-n}{N} \cdot H \left(\frac{M-m}{N-n} \right).$$

Здесь G – так называемый информационный критерий, или количество информации в закономерности. Эта величина есть разность между изначальной энтропией системы и её энтропией после выделения заданной области.

3.1.3 Решающие списки

Когда выбран критерий качества, можно создавать алгоритм поиска закономерностей. Основная техническая проблема заключается в большом пространстве поиска, практически исключающем полный перебор. Напрощается использование направленного перебора.

Схема направленного поиска закономерностей («жадный» алгоритм):

- ✧ дискретизация признаков (границы между проекциями точек выборки);
- ✧ интервальные предикаты;
- ✧ конъюнкции элементарных предикатов.

Наиболее известные алгоритмы такого типа: КОРА, ТЭМП.

В результате работы этих алгоритмов мы получаем большой список закономерностей со значениями критерия качества. Список целесообразно упорядочить по качеству.

На основе списка закономерностей можно решать задачу классификации. Решение принимается по первой закономерности, которой удовлетворяет объект, т. е. приписывается тот номер класса, к которому относится закономерность. Можно применять голосование по нескольким самым информативным закономерностям.

Помимо возможности решения задачи классификации логические закономерности имеют самостоятельную ценность, поскольку отражают особенности данных, что полезно для их анализа.

3.2 Решающие деревья

При независимом поиске закономерностей они оказываются сильно пересекающимися. Если основная цель — решение задачи классификации, то имеет смысл искать закономерности так, чтобы их набор сразу давал разбиение пространства переменных.

Напомним, что *разбиением* называется представление множества в виде объединения непересекающихся подмножеств.

Дерево решений строит иерархическое разбиение пространства переменных.

3.2.1 Понятие дерева решений

Решающие деревья являются одним из наиболее часто используемых инструментов для анализа данных. К достоинствам решающих деревьев относятся их наглядность, применимость к широкому кругу задач, возможность варьировать сложность решений, подстраиваясь под различный объём выборки, способность автоматически выбирать информативные переменные.

Методы, основанные на деревьях решений, применяются почти во всех задачах анализа данных. В первую очередь, рассмотрим задачу классификации.

На рис. 3.2 приведена модельная (т. е. созданная искусственно, а не взятая из реальной задачи) выборка, представленная 24 точками двух классов в двумерном пространстве. Обе переменные количественные, и область значений каждой представляет собой интервал $[0, 1]$. Треугольные маркеры соответствуют первому, круглые — второму классу.

3.2.2 Построение дерева решений

Дерево решений представляет собой последовательное разбиение пространства значений на области. Идея метода, основанного на решающих деревьях, заключается в последовательном разбиении пространства значений переменных на области E_1, \dots, E_L и

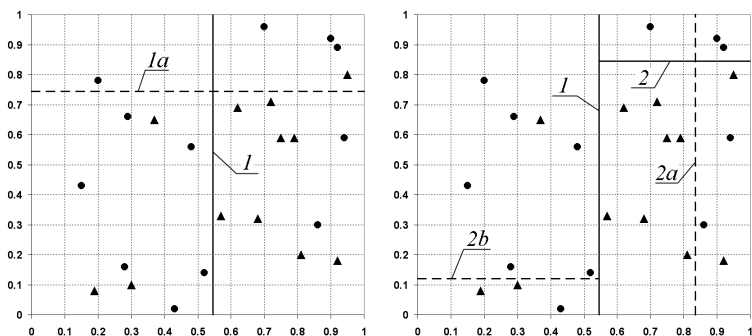


Рис. 3.2. Построение разбиения деревом решений

приписывании каждой полученной области решения в виде номера класса.

Под последовательным разбиением понимается то, что сначала всё пространство $X = X_1 \times X_2$ разбивается на две области $E_1^{(1)}$ и $E_2^{(1)}$, из которых затем выбирается область, которая, в свою очередь, разбивается на две, в результате чего получается три области $E_1^{(2)}, E_2^{(2)}, E_3^{(2)}$, и так далее, пока через $L - 1$ таких шагов не получится L областей.

Заметим, что способов подобного разбиения пространства X на L областей очень много. Естественно выбрать разбиение, минимизирующее число ошибочно классифицируемых объектов выборки. Однако число различных на выборке объёма N разбиений в случае n количественных переменных составляет порядка $(nN)^L$, поэтому полный перебор всех разбиений в реальных задачах обычно невозможен.

При этом задача поиска разбиения, точно классифицирующего обучающую выборку и имеющего минимальное число областей, является NP-полной. Это значит, что не следует рассчитывать на появление имеющих низкую трудоёмкость точных алгоритмов нахождения наилучшего разбиения. В этой ситуации используют эвристические алгоритмы, простейшим вариантом которых является так называемый *жадный* алгоритм.

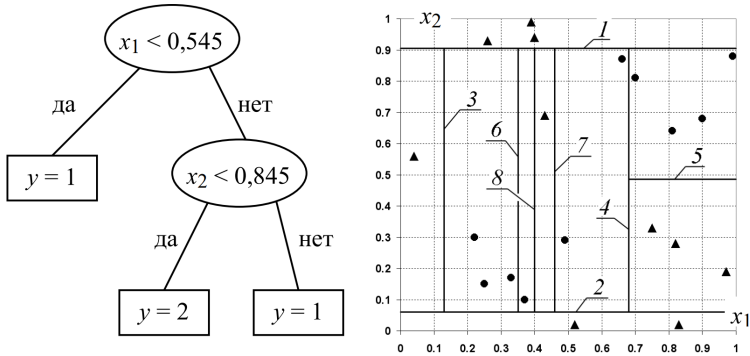


Рис. 3.3. Дерево решений (слева) и неудачное разбиение для задачи «исключающего или» (справа)

Рассмотрим работу жадного алгоритма на примере построения разбиения выборки из рис. 3.2.

На первом шаге алгоритм перебирает все варианты разбиения пространства X на две области. Разбиение можно проводить как по переменной X_1 , так и по X_2 .

Переберём сначала варианты разбиения по переменной X_1 .

Хотя число вариантов разбиения области значений количественной переменной на два интервала бесконечно (континуально), нас интересуют только варианты, различимые на обучающей выборке, поэтому будем рассматривать только границы, расположенные посередине между проекциями выборочных точек.

Каждая граница s разбивает пространство X на области $E_1^{(1)} = [0, s) \times [0, 1]$ и $E_2^{(1)} = [s, 1] \times [0, 1]$. Обозначим за $N_l^1(s)$ число точек первого, а $N_l^2(s)$ – число точек второго класса в области $E_l^{(1)}$.

Естественно, что в каждой получившейся области решающая функция припишет тот класс, выборочных объектов которого в этой области больше. Тогда оставшиеся объекты будут классифицированы ошибочно. Таким образом, число ошибочно классифи-

цированных объектов составит

$$\tilde{N}(s) = \min(N_1^1(s), N_1^2(s)) + \min(N_2^1(s), N_2^2(s)).$$

Перебрав все варианты разбиения пространства по переменной X_1 , видим, что наименьшее число ошибок (восемь) получается при $s = 0,545$. Соответствующая граница показана на рис. 3.2 цифрой 1.

Аналогичным образом следует перебрать все варианты разбиения по переменной X_2 . Наименьшее число ошибок (девять) будет достигнуто для границы, обозначенной на рис. 3.2 как 1a.

Поскольку разбиение по переменной X_1 даёт меньшее число ошибок, окончательно выбираем разбиение с границей 1, т. е. $E_1^{(1)} = [0; 0,545) \times [0, 1]$, $E_2^{(1)} = [0,545; 1] \times [0, 1]$.

На втором шаге алгоритма аналогичным образом перебираем варианты разбиений областей $E_1^{(1)}$ и $E_2^{(1)}$.

В результате перебора находим (см. рис. 3.2), что граница 2b уменьшает общее число ошибок на 1, граница 2a уменьшает общее число ошибок на 2, а граница 2 уменьшает общее число ошибок на 3. В результате выбираем вариант 2 и получаем разбиение на области

$$\begin{aligned} E_1^{(2)} &= [0; 0,545) \times [0; 1], \\ E_2^{(2)} &= [0,545; 1] \times [0; 0,845), \\ E_3^{(2)} &= [0,545; 1] \times [0,845; 1]. \end{aligned}$$

Решающая функция $f(x)$ принимает значение 1 при $x \in E_1^{(2)} \cup E_3^{(2)}$ и значение 2 при $x \in E_2^{(2)}$.

На рис. 3.3 слева эта же решающая функция представлена в форме дерева решений.

На рис. 3.3 справа приведён пример другой выборки. Легко заметить, что данная выборка безошибочно классифицируется при разбиении

$$\begin{aligned} E_1 &= [0; 0,5) \times [0; 0,5), & E_2 &= [0,5; 1] \times [0; 0,5), \\ E_3 &= [0; 0,5) \times [0,5; 1], & E_4 &= [0,5; 1] \times [0,5; 1]. \end{aligned}$$

При этом жадный алгоритм достигает безошибочной классификации только при разбиении на 9 областей (показаны на рис. 3.3, цифрами отражён порядок проведения границ). Заметим, что некоторые проведённые границы вообще не уменьшают число ошибок и проводились так, чтобы разделять область с наибольшим числом объектов на подобласти, разделяя объекты примерно поровну.

Последний пример демонстрирует, что жадный алгоритм в общем случае находит далеко не лучшее решение. Существуют многочисленные усовершенствованные варианты алгоритма направленного поиска дерева решений, однако до настоящего времени проблема построения наиболее эффективного алгоритма остаётся открытой.

Одно из важнейших достоинств решающих деревьев заключается в том, что основанные на них методы построения решающих функций очень легко переносятся на случай разнотипного пространства переменных. Если для вещественных переменных в качестве областей разбиений брались интервалы, то для номинальных (категориальных) переменных в разбиениях участвуют любые подмножества значений.

Варианты алгоритма построения дерева:

- ✧ жадный (без учёта будущих ветвлений);
- ✧ рекурсивный [25] (предикат в узле выбирается с учётом ветвления на нижнем уровне, решает задачу XOR);
- ✧ неограниченный (строит дерево, затем его оптимизирует всё доступное время).

Сейчас на практике почти всегда используется жадный алгоритм.

3.2.3 Критерии ветвления

При построении дерева требуется оценивать качество получаемых вариантов. Если мы наращиваем дерево последовательно, то на каждом шаге требуется сравнивать варианты, которые отли-

чаются только одним нетерминальным узлом и выходящими из него конечными вершинами. Поэтому технически критерий качества можно пересчитывать только для изменяющихся вершин. Иногда критерий выписывают в форме разности качества разных вариантов, однако для понимания проще рассматривать критерий качества всего дерева.

Критерий качества дерева λ задаётся как функция от количества объектов каждого класса в конечных вершинах.

$$K(\lambda) = \sum_{t=1}^T N^t L(\nu_1^t, \dots, \nu_k^t),$$

где N^t – число объектов в конечной вершине t , $(\nu_1^t, \dots, \nu_k^t)$ – относительные частоты классов в вершине t , T – число конечных вершин дерева.

Критерии $L(\dots)$ оценки дерева (для классификации):

- ✧ число ошибок классификации (для детерминированного прогноза);
- ✧ критерий Джини (Gini impurity) — это по сути число ошибок для вероятностного прогноза;
- ✧ информационный (log loss, на основе функции правдоподобия для вероятностного прогноза).

Выражение для критерия числа ошибок

$$L_E(\nu_1^t, \dots, \nu_k^t) = 1 - \max_{\omega}(\nu_{\omega}^t).$$

Эта величина есть доля ошибочно классифицированных объектов, если всей вершине приписан наиболее частый (в ней) класс.

Выражение для критерия Джини

$$L_G(\nu_1^t, \dots, \nu_k^t) = 1 - \sum_{\omega=1}^k (\nu_{\omega}^t)^2.$$

Может показаться странным, что частота возводится в квадрат. На самом деле критерий имеет простой смысл: это вероятность

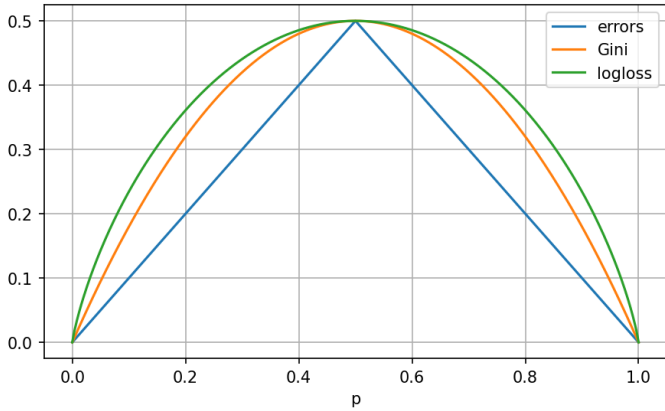


Рис. 3.4. Сравнение критериев ветвления

ошибочной классификации объекта, если класс будем приписывать случайно с вероятностями, равными относительным частотам.

Выражение для критерия log loss

$$L_L(\nu_1^t, \dots, \nu_k^t) = - \sum_{\omega=1}^k \nu_{\omega}^t \ln(\nu_{\omega}^t).$$

Это критерий максимума правдоподобия для вероятностного прогноза в соответствии с частотами.

Первый критерий (доля ошибок) работает плохо, остальные сопоставимо.

На рис. 3.4 можно увидеть, что Gini impurity и log loss отличаются, но не слишком сильно.

3.3 Выводы

Недостатки логических методов:

- ✧ невозможность «гладких» решений;
- ✧ много эвристик (настраиваемых параметров);

- ✧ вычислительная трудоёмкость нахождения точных решений;
- ✧ неизвестен лучший критерий ветвления.

Достоинства логических методов:

- ✧ интерпретируемость решений;
- ✧ работа в разнотипном пространстве (однако не все реализации это поддерживают);
- ✧ работа с пропусками;
- ✧ применимость в ансамблевых методах.

Бустинг на деревьях — один из лучших (по точности) методов для табличных данных.

При использовании деревьев пропуски заполнять нецелесообразно (кроме некоторых случаев, когда с пропусками связан target leak).

3.4 Примеры кода

Приведём пример кода для построения дерева и его визуализации.

В примере также показана возможность сохранения рисунка в файл.

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn import tree

clf = tree.DecisionTreeClassifier(max_depth=3,
                                 random_state=0)

iris = load_iris()
clf = clf.fit(iris.data, iris.target)

plt.figure(figsize=(8,8))
tree.plot_tree(clf)
plt.savefig('tree_iris.png', dpi=300,
           bbox_inches='tight')
plt.show()
```

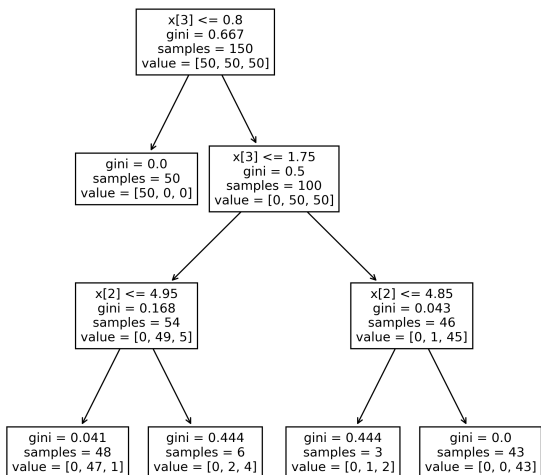


Рис. 3.5. Дерево решений для задачи Iris

Результат выполнения кода приведён на рис. 3.5. Размер (глубина) дерева принудительно ограничен с помощью соответствующего параметра.

Контрольные вопросы

1. Что такое закономерность с точки зрения машинного обучения? В чём идея статистического критерия качества закономерности?
2. Что такое решающие списки?
3. Что такое дерево решений?
4. Какова формула и в чём содержательный смысл критерия ветвления Gini impurity?

5. Как устроены жадный и рекурсивный алгоритмы построения дерева? Каковы трудоёмкости этих алгоритмов?
6. Какова вычислительная сложность задачи построения оптимального дерева?
7. Как дерево работает с категориальными переменными?
8. Как дерево работает с пропусками в данных?
9. В чём заключаются основные достоинства и недостатки методов, основанных на деревьях?

Глава 4

Методы, основанные на КОМПОЗИЦИЯХ

4.1 Композиции решающих функций

Пусть, как и ранее, X – пространство значений прогнозирующих переменных, $Y = \{-1, 1\}$ – прогнозируемая переменная, $D = X \times Y$.

Решающая функция (алгоритм классификации) есть отображение

$$f : X \rightarrow Y.$$

Дана случайная независимая выборка

$$S = ((x^i, y^i) \in D \mid i = \overline{1, N}), \quad S \in D^N.$$

Метод построения решающих функций есть

$$Q : D^N \rightarrow \Lambda,$$

где Λ – заданный класс решающих функций.

Введём обобщение решающей функции

$$\lambda : X \rightarrow (-\infty, +\infty),$$

где отображение происходит в так называемое *пространство оценок* [20; 39]. Здесь решающая функция не приписывает номер класса объекту, а даёт некоторую оценку, которая монотонно связана с предполагаемой вероятностью принадлежности объекта выбранному классу, но задана в произвольной шкале.

Оценкой может быть, например, так называемый «сырой» прогноз. Такой прогноз выдаёт метод `decision_function` классов библиотеки `scikit-learn`.

Пусть имеются T решающих функций $\lambda_1(x), \dots, \lambda_T(x)$.

Композиция классификаторов есть решение в виде

$$\lambda(x) = C(\lambda_1(x), \dots, \lambda_T(x)),$$

где $C(\cdot, \dots, \cdot)$ – монотонна (не убывает) по всем аргументам.

Функции $\lambda_t(x)$ принимают значения из пространства оценок, значения функции $\lambda(x)$ – из множества Y , но могут быть и оценками вероятности.

Функцию $C(\cdot, \dots, \cdot)$ будем называть агрегирующей функцией или *метамоделью* (метаклассификатором в случае задачи классификации).

Ограничение монотонности по аргументам оставляет слишком широкий произвол для выбора $C(\cdot, \dots, \cdot)$, поэтому часто ограничиваются линейными функциями.

Линейная композиция есть

$$\lambda(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t \lambda_t(x) \right).$$

Если $\alpha_t \geq 0$, то речь идёт о выпуклых композициях (чаще всего рассматривают именно их).

Мы рассмотрим следующие методы построения композиций:

- ✧ бустинг (AdaBoost, градиентный бустинг);
- ✧ бэггинг и метод случайных подпространств (случайный лес);
- ✧ нейронные сети;
- ✧ голосование (блендинг);

✧ стэкинг (решения в качестве новых переменных).

Перечисленные композиции в основном линейные, кроме нейронных сетей (которые отличаются ещё и многоуровневостью) и стэкинга.

4.2 Независимые подвыборки

Предположим, что у нас есть некоторый метод классификации (или, в общем случае, построения решающих функций), который назовём базовым методом.

Мы хотим на его основе построить ансамбль из решений. Для этого нужно каким-то образом добиться, чтобы один и тот же метод выдавал разные решения.

Относительно многие реализации методов при поиске решения используют случайность. Для таких реализаций можно добиться разных решений, задавая разные значения параметра базы генератора псевдослучайных чисел.

Но такой подход, во-первых, не для всех алгоритмов возможен, а во-вторых, получаемые таким образом решения для нас «недостаточно различны».

Кроме того, формально под методом построения решающих функций мы понимаем отображение из множества выборок во множество решений. В этом случае получить одним методом разные решения можно, только подав на вход разные выборки. При этом обучающая выборка у нас изначально одна.

Простейший и совершенно напрашивающийся способ получить из одной выборки несколько — это брать случайные подвыборки. При этом (если представлять выборку в виде таблицы данных) можно выбирать как строки (объекты), так и столбцы (признаки).

Оба этих варианта используются на практике (как правило, совместно).

4.2.1 Бэггинг

Выбирая случайные подмножества строк и столбцов таблицы данных, можно получать различные решающие функции. Но что если мы захотим получить «ещё более различные» базовые решения?

Для этих целей придумана идея бутстреп-выборок, а весь подход назван *бэггингом* (bagging — сокращение от bootstrap aggregation).

Чтобы добиться большего различия решений, мы будем не просто брать подвыборку, но ещё и искажать её. При этом искажения будут очень умеренными: мы всего лишь повторим некоторые строки два или более раз. Фактически мы этим просто приписываем объектам некоторый случайный (целочисленный) вес.

Но как именно реализовать подобное искажение подвыборки? Для этого предложена изящная идея *бутстреп-выборок*.

Идея проста: формировать новую выборку на основе исходной как выборку с повторениями, причём размер новой выборки задаётся равным исходной.

Когда говорят о случайной подвыборке, то подразумевают, что подвыборка формируется без повторений: случайный объект исходной выборки мы как бы переносим в новую выборку, и в следующий раз он нам попасться не может.

Формируя выборку с повторениями, мы как бы копируем объект в новую выборку — и он может быть выбран много раз.

Если размер формируемой выборки равен исходной, то в среднем $(1 - \frac{1}{N})^N \approx e^{-1} \approx 37\%$ объектов в неё не попадут (на их месте будут копии других объектов).

Таким образом, идея бэггинга очень простая. Осталось лишь добавить, что она... как ни странно, работает.

4.2.2 Случайный лес

Метод *случайного леса* (random forest) строит ансамбль деревьев путём простого усреднения решений, построенных по бутстреп-выборкам на случайном подмножестве переменных.

Увеличение числа деревьев в методе случайного леса не увеличивает (в отличие от бустинга) эффективную сложность итогового решения и не может привести к падению качества (на контрольной выборке).

Таким образом, увеличивая число деревьев, можно только улучшить решение, однако использовать слишком большое число деревьев (больше сотен) обычно не имеет смысла, поскольку качество достаточно быстро стабилизируется на одном уровне.

Случайный лес использует привычные реализации алгоритма построения деревьев.

Соответственно он требует подбора (оптимизации) ровно тех же параметров, которые подбираются при построении дерева решений.

В первую очередь, требуется подбирать размер (глубину) дерева. Но если при построении одного дерева случаи, когда наилучшим выбором оказывались деревья неограниченной глубины, можно считать экзотикой, то для случайного леса такой вариант довольно реалистичен (однако не настолько, чтобы пренебрегать оптимизацией глубины).

4.2.3 Свойства

Случайный лес обычно даёт достаточно хорошее решение, однако автору не известен ни один случай, когда (оптимально настроенный) случайный лес превзошёл бы по качеству (оптимально настроенный) бустинг (это не значит, что таких случаев нет).

Как правило, случайный лес даёт качество выше, чем одно дерево. Это кажется естественным, но по факту вовсе не очевидно, что так и должно быть.

Очевидно, что ансамбль независимых деревьев должен работать лучше, чем одно такое дерево.

Но дело в том, что в случайном лесе деревья «не такие», как при обычном построении дерева. Обычно мы дерево строим по всем переменным и по всей (не искажённой бутстрепом) выборке. Случайный же лес строит искусственно «ослабленные» деревья.

Можно доказать (через разложение ошибки на смещение и разброс), что случайный лес как ансамбль ослабленных деревьев должен работать лучше, чем одно ослабленное дерево. Но ниоткуда (кроме интуиции) не следует, что ансамбль ослабленных деревьев должен работать лучше, чем одно «полноценное» дерево.

Также заметим, что, строго говоря, деревья случайного леса не являются независимыми, поскольку строятся на подвыборках одной и той же выборки. Отсюда, в частности, следует, что дисперсию прогноза невозможно уменьшать до нуля (в пределе), увеличивая число деревьев.

4.3 Бустинг

Как и в предыдущем разделе, мы хотим получать одним базовым методом различные решения путём построения новых выборок на основе исходной.

Но теперь новые выборки будут получаться за счёт изменения весов объектов (а для градиентного бустинга будут также корректироваться значения целевой переменной).

Идея бустинга заключается в том, чтобы каждый последующий базовый классификатор исправлял ошибки предыдущих.

Вторым принципом бустинга является то, что мы не пытаемся улучшить ранее построенные классификаторы, т. е. если базовая решающая функция построена, то она включается в композицию и в дальнейшем не изменяется.

Рассмотрим модельный пример. Пусть есть некоторая выборка, и базовый метод построил по ней линейный классификатор как на рис. 4.1.

Ошибочно классифицированные объекты выделены крупными маркерами.

Представим теперь, что мы хотим, чтобы следующий классификатор исправил эти ошибки. Простейший способ этого добиться — оставить только эти ошибочно классифицированные объекты, а остальные просто убрать из выборки. Легко представить, к чему это приведёт: второй классификатор разобьёт пространство

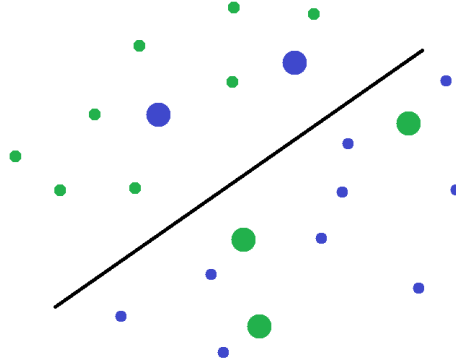


Рис. 4.1. Идея бустинга

на примерно те же полуплоскости, но метки классов припишет им противоположным образом. Если теперь усреднить прогнозы от двух полученных решений, то... ничего хорошего не получится: прогноз будет везде нулевым (или совпадёт с одним из базовых решений, если веса при усреднении возьмём неравными).

Данные рассуждения призваны показать, что несмотря на очевидность идеи бустинга, реализовывать её нужно аккуратно.

4.3.1 Алгоритм AdaBoost

Увеличиваем веса объектов, на которых допущена ошибка.

Данный алгоритм является исторически первым и самым простым вариантом бустинга.

Изложим AdaBoost как собственно алгоритм классификации.

В методе AdaBoost решение строится в виде композиции

$$\lambda(x) = \text{sign}(\beta_T(x)), \quad \beta_T(x) = \sum_{t=1}^T \alpha_t \lambda_t(x),$$

где базовые классификаторы $\lambda_t(x)$ и их веса α_t находятся следующим образом.

Первый базовый классификатор строится базовым методом на основе исходной выборки, объектам которой приписаны начальные веса $w^1 = (w_1^1, \dots, w_N^1)$.

Заметим, что мы будем задавать начальные веса объектам в соответствии с выбранным распределением, но в стандартном варианте метода начальные веса выбираются одинаковыми, т. е. $w_i^1 = \frac{1}{N}$.

Пересчёт весов.

Вес построенного базового классификатора в композиции определяется по формуле

$$\alpha_t = \frac{1}{2} \ln \frac{W^+(S, w^t, \lambda_t)}{W^-(S, w^t, \lambda_t)},$$

где

$$W^+(S, w, \lambda) = \sum_{i=1}^N w_i \cdot I(y^i = \lambda(x^i)) + \varkappa,$$

$$W^-(S, w, \lambda) = \sum_{i=1}^N w_i \cdot I(y^i = -\lambda(x^i)) + \varkappa.$$

Здесь $I(\cdot)$ – индикаторная функция, а \varkappa – небольшое положительное число (параметр), выполняющее функцию регуляризатора (иначе под логарифмом может оказаться 0). Итерационный процесс.

Следующие базовые классификаторы строятся тем же базовым методом по выборке, веса объектов в которой вычисляются по формулам

$$w_i^{t+1} = \frac{\bar{w}_i^{t+1}}{\sum_{i=1}^N \bar{w}_i^{t+1}}, \quad \bar{w}_i^{t+1} = w_i^t \cdot e^{-\alpha_t y^i \lambda_t(x^i)}.$$

Веса правильно классифицированных объектов умножаются на $e^{-\alpha t}$, а веса неправильно классифицированных объектов умножаются на $e^{\alpha t}$.

Как и в классическом варианте алгоритма, мы здесь использовали перенормировку весов на каждом шаге. На самом деле, её

можно было не делать, так как на решение она не влияет. Единственный смысл этой нормировки — избежать очень больших и очень маленьких значений веса, которые теоретически могут выйти за границы диапазона машинного представления вещественных чисел.

4.3.2 Свойства

Если проанализировать результаты соревнований (конкурсов) по анализу данных, то можно сделать вывод, что на задачах с табличными данными из всех методов именно бустинг чаще всего даёт самые точные решения.

То, что бустинг решает задачу точнее, чем одно дерево, кажется естественным, но по факту вовсе не очевидно, что так и должно быть.

Естественно, что ансамбль деревьев работает лучше, чем одно дерево из ансамбля.

Но в бустинге деревья «не такие», как при обычном построении дерева. Обычное дерево строится по исходной выборке.

Бустинг строит деревья по изменённой выборке, и эти деревья вообще не предназначены для использования в качестве самостоятельного решения, поскольку обучались корректировать ошибки.

Можно доказать (через разложение ошибки на смещение и разброс), что бустинг как ансамбль специфических деревьев должен работать лучше, чем одно дерево из ансамбля. Но ниоткуда (кроме интуиции) не следует, что ансамбль специфических деревьев должен работать лучше, чем одно «полноценное» дерево.

4.4 Нейронные сети

Под *нейронной сетью* (прямого распространения) понимается решающая функция в виде суперпозиции произвольного количества линейных функций многих переменных и нелинейных функций одной переменной.

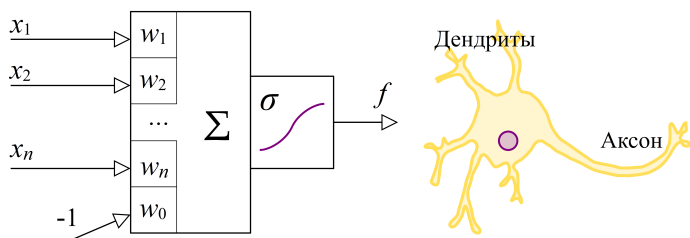


Рис. 4.2. Схемы искусственного и реального нейронов

Без ограничения общности можно считать, что линейные и нелинейные функции в суперпозиции строго чередуются (по уровням), поскольку суперпозиция любого числа линейных функций является линейной функцией (аналогичным образом, идущие подряд нелинейные функции можно считать одной функцией).

Последовательно применённые линейную и нелинейную функции будем называть (искусственным) *нейроном*.

4.4.1 Схема нейрона

На рис. 4.2 приведены схемы искусственного и реального нейронов.

Математически нейрон представляет собой функцию вида

$$f(x) = \sigma(wx - w_0),$$

где x – вектор входных переменных, w – вектор весов, а $\sigma(\cdot)$ – так называемая функция активации.

Несмотря на простоту, математический нейрон, пожалуй, всё же является некоторой (разумеется, грубой) моделью нейрона реального. Действительно, веса входов могут соответствовать разной «проводимости» дендритов, а функция активации некоторым образом соответствует процессу возбуждения нейрона, в результате которого сигнал передаётся по аксону.

Ранее обозначение σ мы использовали для логистической функции. В дальнейшем будем этого придерживаться, но в контексте нейронных сетей σ будет обозначать функцию активации.

Теоретически функция активации может быть любой, но на практике используются только монотонные. Из монотонных без ограничения общности можно оставить только монотонно неубывающие, поскольку произвольный знак обеспечивается линейными коэффициентами.

4.4.2 Аппроксимирующая способность сети

Для начала выясним, как с помощью нейросети реализовать булевы функции.

Начнём с простейших. Пусть x_1, x_2, x – булевы переменные, тогда

$$x_1 \wedge x_2 = I(x_1 + x_2 - \frac{3}{2} > 0),$$

$$x_1 \vee x_2 = I(x_1 + x_2 - \frac{1}{2} > 0),$$

$$\neg x = I(-x + \frac{1}{2} > 0).$$

Теперь мы можем прийти к очевидному заключению, что двухслойная сеть может представить произвольную булеву функцию. Это следует из того, что любая функция может быть представлена дизъюнктивной или конъюнктивной нормальной формой, а дизъюнкция и конъюнкция любого числа переменных могут быть реализованы одним нейроном.

Для функций произвольных переменных соответствующие результаты уже не настолько просты, поэтому мы ограничимся констатацией факта, что достаточно сложная сеть с заданной точностью аппроксимирует любую непрерывную функцию.

4.4.3 Обучение сети

На рис. 4.3 для примера приведена схема двухслойной полносвязной сети.

Если стоит задача классификации на несколько классов, то для каждого класса на выходном слое помещается нейрон, выход

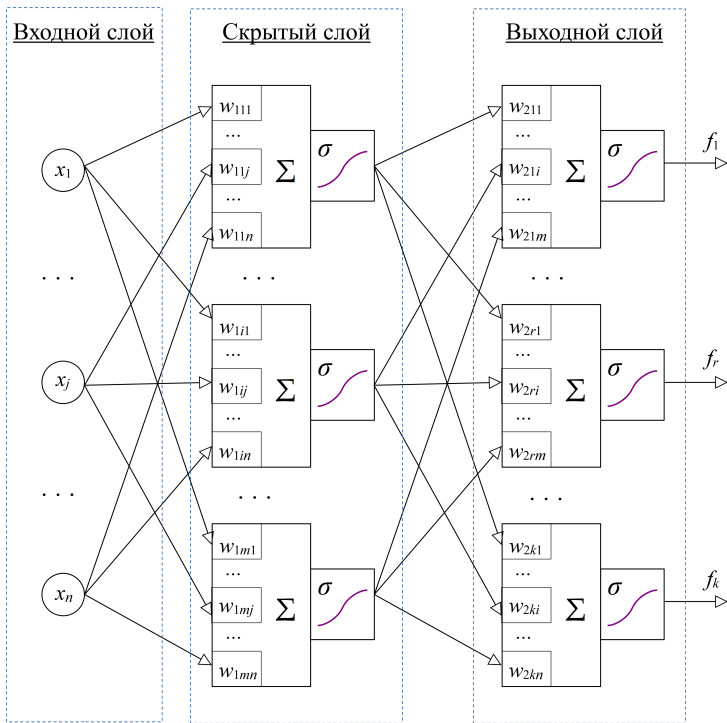


Рис. 4.3. Схема двухслойной полносвязной сети

которого преобразуется в вероятность принадлежности объекта этому классу.

Для получения вероятности на выходном слое в качестве функции активации нужно использовать softmax. Заметим, что получившаяся архитектура будет немного отличаться от изображённой на схеме, а именно — функция активации на выходном слое будет применяться не независимо к выходу каждого нейрона, а ко всем выходам вместе. Функцию softmax можно представлять и иначе: к выходу каждого нейрона применяется функция активации в виде экспоненты, а затем сумма выходов нормируется на единицу.

Если классов два, то эта схема будет эквивалентна сети с одним выходным нейроном с логистической функцией активации.

Поставим задачу обучения сети.

Обучение подразумевает минимизацию функции потерь на обучающей выборке. Для начала будем минимизировать потери на одном объекте выборки, который обозначим (x, y) . Мы для него не используем индексов, чтобы не загромождать обозначения.

Итак, нужно минимизировать функцию

$$L(f(x), y),$$

где

$$f(x) = (f_1(x), \dots, f_k(x)), \quad f_r(x) = \sigma(z_{2r}),$$

$$z_{2r} = \sum_{i=1}^m w_{2ri} \sigma(z_{1i}), \quad z_{1i} = \sum_{j=1}^n w_{1ij} x_j.$$

Минимизацию будем вести градиентным методом в пространстве параметров w_{ij} , для чего нам понадобится вычислять производные по этим параметрам.

Вычислим производную по параметрам последнего слоя

$$\begin{aligned} \frac{\partial L(f(x), y)}{\partial w_{2ri}} &= L'_r(f(x), y) \cdot \frac{\partial f_r(x)}{\partial w_{2ri}} = \\ &= L'_r(f(x), y) \sigma'(z_{2r}) \sigma(z_{1i}). \end{aligned}$$

Всё, что нам нужно знать для вычисления данной производной, это z_{lij} – сигналы внутри сети, для получения которых нужно всего лишь подать на вход сети рассматриваемый объект.

Вычислим производную по параметрам скрытого слоя

$$\begin{aligned} \frac{\partial L(f(x), y)}{\partial w_{1ij}} &= \sum_{r=1}^k L'_r(f(x), y) \cdot \frac{\partial f_r(x)}{\partial w_{1ij}} = \\ &= \sum_{r=1}^k L'_r(f(x), y) \sigma'(z_{2r}) w_{2ri} \sigma'(z_{1i}) x_j. \end{aligned}$$

Видим (сравнив полученную формулу с предыдущей), что производные выражаются через уже вычисленные производные последнего слоя.

Такой процесс называется процедурой *обратного распространения ошибок* — когда производные по параметрам некоторого слоя выражаются через производные последующих слоёв. Это позволяет эффективно вычислить градиент по всем параметрам, пройдя сеть в обратном направлении.

4.4.4 Архитектуры нейронных сетей

На практике используются сети прямого распространения:

- ✧ полносвязные;
- ✧ рекуррентные;
- ✧ свёрточные;
- ✧ с механизмом внимания (трансформеры).

Мы дали определение нейросети как суперпозиции функций. И все реально используемые сети под такое определение подходят.

Но возможно нарисовать такую сеть нейронов, что она не будет являться суперпозицией функций.

Для этого достаточно соединить нейроны так, чтобы получилась петля, т. е. такой путь по сети, когда сигнал повторно попадает в нейрон, где он уже был.

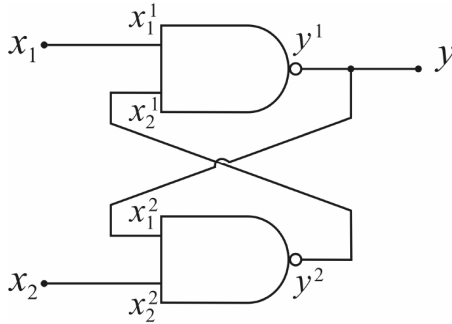


Рис. 4.4. Схема триггера из логических элементов

Простейшим примером такой сети является всем известный триггер (рис. 4.4). Представим триггер как нейросеть с памятью.

Для представления триггера нейросетью достаточно вспомнить, что мы уже научились представлять нейросетями базовые логические функции, а триггер как раз и состоит из логических элементов.

Как известно, триггер — это элементарная ячейка памяти, т. е. то, что позволяет хранить 1 бит информации. И это хранение обеспечивается именно петлями в связях.

Заметим, что есть сети с архитектурой LSTM, которые как раз называются сетями с памятью.

Значит ли это, что эффект памяти можно реализовать без петель? Нет. То, что реализовано в LSTM — это не память, а всего лишь медленно распространяющиеся сигналы (входом сети является вся последовательность, и отклик сети на эту последовательность не зависит от предыдущих последовательностей). Но поскольку такие сигналы уже общепринято называть памятью, нам придётся этому следовать, но держать в голове, что это «не настоящая» память.

Можно пофантазировать, что «петли» при распространении сигналов в живых нейронных сетях могут как-то участвовать в формировании кратковременной памяти.

4.4.5 Глубокое обучение

Глубокое обучение — методы машинного обучения, формирующие многоуровневое представление данных. Это очень расплывчатая формулировка, поэтому желательно конкретизировать. Данные мы представляем с помощью признаков (переменных). И первый уровень представления у нас уже есть — это тот вид, в котором данные представлены изначально.

Методом глубокого обучения можем назвать любой метод, который в ходе работы формирует внутреннее признаковое представление. Таким образом, в результате получается иерархическое представление данных.

Очевидный пример — нейросеть, использующая эмбединги. Другой пример — обычные свёрточные сети.

Многослойный перцептрон — спорный случай. Уровней много, но они дают просто аппроксимацию (имеем не новые признаки, а вспомогательные переменные).

Бустинг следует (хотя это спорное утверждение) отнести к глубокому обучению, поскольку он и выделяет, и формирует признаки. В частности, бустинг может реализовать наивный байесовский классификатор с формированием соответствующих признаков (по сути, автоматически выполняя *target encoding*).

Метод опорных векторов можно было бы отнести к глубокому обучению, если бы ядра строились автоматически.

Часто под глубоким обучением подразумевают просто использование глубоких нейронных сетей.

Тема глубокого обучения требует отдельного изложения, для ознакомления можно рекомендовать [38].

4.4.6 Бустинг и нейронные сети

В задачах с табличными данными нейронные сети, как правило, уступают (и существенно) бустингу в точности¹. В справедливости

¹Недавно появившаяся архитектура TabNet оспаривает данное утверждение, но для объективных выводов пока недостаточно информации. Требуется массовая апробация метода на kaggle.

ности этого наблюдения читатель может убедиться, проанализировав результаты соревнований на kaggle.

Ещё раз подчеркнём, что этот вывод касается только задач с табличными данными. Если данные представляют собой изображения, сложные сигналы, текст, то нейронные сети вне конкуренции.

Автору известны лишь отдельные примеры, когда нейросеть по точности превосходила бустинг.

Один из таких примеров представлял задачу с большим количеством целевых переменных. Не путать с многоклассовой классификацией, когда переменная одна, а значений классов — много. Здесь именно случай нескольких целевых переменных (например, прогнозируем одновременно и образование, и профессию человека).

Проблема с такими задачами в том, что существующие реализации бустинга не способны строить один ансамбль деревьев сразу для всех целевых переменных. Они строят свои деревья для каждой целевой переменной. На самом деле, ситуация ещё хуже, и большинство реализаций даже обычную многоклассовую задачу решает методом «один против всех», и лишь некоторые реализации умеют строить один ансамбль деревьев на все классы.

Заметим, что это не проблема бустинга как подхода, а именно ограничения реализации метода.

Особенности бустинга, отличающие его от нейросетей:

- ✧ делает отбор переменных;
- ✧ может эксплуатировать независимость;
- ✧ не корректирует уже построенные деревья.

Первые два пункта безусловно являются достоинствами.

С последним — не всё так очевидно. На первый взгляд кажется, что это искусственное ограничение, принятое для простоты реализации. И если его снять (т. е. при добавлении дерева корректировать и предыдущие), то качество должно возрасти.

Однако более вероятно, что эта особенность — также работает в пользу эффективности бустинга.

Заметим, что произвольное дерево решений может быть (точно) представлено нейросетью соответствующей архитектуры. Достаточно вспомнить, что дерево представляется с помощью логических функций и предикатов, а это всё легко представимо нейросетью.

Получается, что дерево решений, а также ансамбль деревьев могут быть заданы в форме нейросети, т. е. являются частными случаями нейросети.

Это так, но из этого не следует, что нейросеть может заменить бустинг. Всё дело в том, что в машинном обучении важен не только (и не столько) класс решающих функций, но и метод обучения. Например, и дискриминант Фишера, и метод опорных векторов — оба работают в классе линейных функций, но решения дают разные.

Нейросеть подходящей архитектуры вполне может совпасть как решающая функция с ансамблем деревьев — но в процессе обучения на реальных данных такого совпадения (почти навверное) не произойдёт.

Это примерно так же, как обезьяна, посаженная за печатную машинку, может случайно напечатать «Войну и мир». Но не печатает.

Свойства нейронных сетей:

- ✧ являются универсальным аппроксиматором;
- ✧ решения почти всех методов классификации можно выразить через нейросеть, но нейросеть не заменяет этих методов;
- ✧ имеют меньшую гибкость в способах обучения и в выборе регуляризатора;
- ✧ на «табличных» задачах обычно уступают бустингу;
- ✧ во многих других задачах (изображения, текст) не имеют конкурентных альтернатив;
- ✧ в задачах с нулевым байесовским уровнем ошибки переобучение менее критично.

4.5 Ансамбли на основе различных методов

Рассмотрим теперь ансамбли, в которых базовые решения строятся разными методами.

В этом случае не возникает проблемы построения разных решений на одной выборке.

Достаточно взять различные доступные алгоритмы. Также можно повторно использовать один и тот же алгоритм, задавая различные параметры. Мы будем формально считать такие варианты разными методами.

Базовые алгоритмы также называют «слабыми» моделями (weak learners). Для бустинга или случайного леса это вполне оправдано, но в других случаях такой термин становится условным, поскольку при простом усреднении точность ансамбля зачастую не очень значительно превосходит точность базовых классификаторов, а, например, стэкинг может давать даже меньшую точность, чем модели, на которых он построен.

Особенность ансамблей данного раздела как раз в том, что базовые модели в них являются относительно «сильными».

4.5.1 Простое голосование

Самой простой способ построения ансамбля — это просто усреднить предсказания базовых методов. Такой подход мы будем называть *блендингом* (blending). Данный термин кажется уместным, поскольку blend означает «смешивать», а усреднение — это и есть смешивание.

Заметим, что в литературе термин блендинг чаще используется как полный или частичный синоним стэкинга. Но объективно идея стэкинга совершенно непохожа на «смешивание», кроме частного случая, когда метаклассификатор выбран линейным. Поэтому будем использовать менее распространённую, но более логичную и тоже встречающуюся в литературе интерпретацию блендинга как именно усреднение.

Однако при использовании даже такого простого подхода требуется аккуратность.

Предположим, что мы решаем задачу классификации с двумя классами и у нас есть три базовых метода. Пусть для какого-то объекта первые два метода оценили вероятность принадлежности объекта классу 1 как 0,2, а третий метод оценил эту вероятность как 0,99.

Если просто взять среднее арифметическое трёх вероятностей, то получим 0,46, т. е. будет приписан класс –1.

Но усреднять непосредственно вероятности в данном случае — безусловно плохая идея. При этом не учитывается, что чем ближе вероятность к 0 или 1, тем более значимы различия. Иными словами, вероятности 0,99 и 0,999 отличаются сильнее, чем 0,5 и 0,6.

Вспомним, что при рассмотрении наивного байесовского классификатора мы уже сталкивались с подобной задачей. Там мы агрегировали прогнозы по независимым переменным. И тогда мы получили формулу, по которой индивидуальные прогнозы надо было подвергнуть обратному логистическому преобразованию, затем сложить, а в конце применить прямую логистическую функцию. Если проделать эти действия для вероятностей 0,2, 0,2 и 0,99, то получим 0,86.

Однако эта формула справедлива для независимых переменных, но решения, полученные разными методами нельзя считать независимыми, поскольку они построены по одной и той же выборке (хотя вопрос не так прост).

Разумным решением в такой ситуации кажется использовать логистическое преобразование, но не складывать преобразованные величины, а усреднять их (что мы, собственно, изначально и собирались делать). По такому способу для рассмотренного примера мы получим 0,65. Будет приписан класс 1, но уверенность невысокая (что оправдано, поскольку два метода проголосовали «против», и только высокая уверенность третьего метода склонила решение к 1-му классу).

Разумеется, веса методов при усреднении не обязательно брать равными. Их можно задавать эвристически (исходя из опыта применения каждого метода).

Также разумный вариант — задавать веса в соответствии с качеством, которое метод достигает на кроссвалидации (методу кроссвалидации будет посвящён параграф 6.4).

Также напрашивается идея подбирать веса, напрямую минимизируя потери на выборке. Для этого подойдёт уже знакомый метод логистической регрессии. Но это уже тема следующего подраздела.

Заметим, что на практике самое простое голосование (с равными весами) может оказываться лучше всех «продвинутых» способов ансамблирования. Это возможно при условии, что базовые алгоритмы хорошо откалиброваны, т. е. выдают адекватный уровень уверенности в прогнозе.

4.5.2 Стэкинг

Стэкинг (stacking) — довольно популярный метод ансамблирования. А если учесть, что target encoding — это фактически частный случай блендинга — то весьма популярный.

Идея заключается в том, что мы решаем задачу разными методами, как в блендинге, но полученные прогнозы не усредняем, а используем в качестве новых переменных.

Новые переменные мы используем «обычным образом», т. е. на них строим решение, совершенно так же, как на любых других переменных.

Эти новые переменные могут как заменять исходные, так и добавляться к ним.

Важный момент: прогнозы базовыми алгоритмами нужно (желательно) строить в режиме кроссвалидации, а именно, по схеме out of fold. Подробно эта схема описана в соответствующем разделе. Если пренебречь этим советом, то проявится эффект domain shift (обсуждается в главе 10). Суть этого эффекта применительно к стэкингу заключается в следующем. Практически любые методы на обучающей выборке дают гораздо более «уверенные» прогнозы, чем на тестовых данных. И если мы используем «сырой прогноз» (или вероятностный прогноз с обрантым логистическим

преобразованием), то на обучающих данных такие прогнозы будут относительно большими по абсолютной величине, а на тестовых данных — маленькими. Если теперь мы будем обучать метамодель, используя базовые прогнозы на обучении, то она обучится («привыкнет») работать с большими по модулю входными значениями. А на новых объектах базовые прогнозы будут меньше по модулю — с такими величинами метамодель окажется необученной работать. Это и есть сдвиг домена.

Строго говоря, для target encoding также нужно использовать скорректированные оценки частот, например на основе leave one out (описан в последующих разделах) либо с использованием регуляризации.

Заметим, что для получения новых переменных в стэкинге нам не обязательно (базовыми алгоритмами) прогнозировать целевую переменную. Можно прогнозировать любые переменные. Это довольно экзотический вариант, но иногда подобные приёмы срабатывают.

4.6 Выводы

4.6.1 Преимущества ансамблей

Практически на всех задачах анализа данных лучшие результаты достигаются именно ансамблевыми методами.

Этот факт кажется вполне естественным.

Например, известно, что если провести несколько измерений какой-либо величины, а затем взять среднее, то ошибка уменьшится.

Это легко доказывается в теории вероятностей: дисперсия среднего независимых одинаково распределённых случайных величин в N раз меньше дисперсии каждой величины.

Но за счёт усреднения уменьшается только случайная ошибка, систематическая ошибка остаётся прежней. Похожие рассуждения используются в разложении на bias и variance (смещение и разброс). В дальнейшем мы подробно рассмотрим это разложение

и обсудим, насколько оно объясняет эффективность ансамблей.

В примере с уменьшением ошибки при усреднении результатов измерений есть ключевой момент: ошибки измерений предполагаются независимыми.

В противоположность этому, решения, используемые в ансамбле, являются зависимыми, поскольку строятся на основе одной и той же выборки. Поэтому пример с независимыми измерениями совершенно не годится для объяснения эффективности ансамблей решающих функций.

Также часто как пример эффективности коллектива приводится случай, как на рынке корову старик продавал эксперимент, который провёл Фрэнсис Гальтон. В этом эксперименте всем желающим предлагалось на глаз оценить вес животного. Оказалось, что отдельные оценки имели значительный разброс, однако среднее значение получилось очень близко к правильному.

Что этот эксперимент показывает? Только то, что мы и так знаем: при усреднении независимых измерений ошибка уменьшается.

Можно ли считать оценки участников эксперимента независимыми? Скорее, да, поскольку друг с другом они не советовались.

Можно ли считать эти оценки измерениями? Давайте проинтерпретируем их как результат работы нейросети. Ведь человеческий мозг — нейросеть, которая обучается в процессе жизни. Получается, что в эксперименте мы имеем ансамбль нейросетей. Но можно ли назвать полученное решение блендингом? Нет. Поскольку каждая нейросеть обучалась на своих уникальных данных (у каждого человека свой уникальный опыт).

Данное рассуждение призвано показать, что бытовая интуиция, выражаемая поговоркой «одна голова хорошо, а две — лучше», на самом деле не имеет отношения к ситуации с ансамблевыми методами.

Если повторением измерений мы можем случайную ошибку сколь угодно приблизить к нулю, то добавлением методов в ансамбль погрешность можно уменьшить только до определённого предела. Всё дело в том, что выборка одна и добавление методов

не добавляет новой информации. Всё, на что мы можем рассчитывать, — это более полно использовать имеющуюся информацию.

Если говорить о практическом применении блендинга, то он, как правило, действительно повышает точность, но зачастую не очень существенно. Для победы в конкурсе даже маленькая прибавка важна, поэтому в конкурсах блендинг и стэкинг применяются очень часто. Однако при решении реальных задач их применение часто оказывается неоправданным, поскольку небольшая прибавка точности не компенсирует радикального повышения сложности и громоздкости решения.

4.6.2 Бустинг и случайный лес

Методы существенно различаются в настройке:

- ✧ для бустинга параметры сложности — глубина дерева и количество деревьев;
- ✧ для `random forest` сложность не увеличивается с ростом числа деревьев;
- ✧ как правило, бустинг использует деревья меньшей глубины.

Как правило, бустинг достигает лучшего качества.

Свойства ансамблей деревьев:

- ✧ работа в разнотипном пространстве;
- ✧ работа с пропусками;
- ✧ интерпретируемость решений теряется.

Бустинг на деревьях — один из лучших методов.

4.7 Примеры кода

4.7.1 Подбор сложности ансамбля

Следующий пример кода демонстрирует построение ансамблей деревьев.

```
import pandas as pd
import numpy as np
```

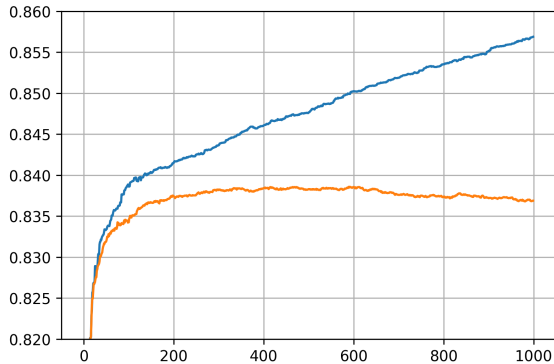


Рис. 4.5. Точность на обучающей выборке и кроссвалидации

```

import matplotlib.pyplot as plt
from sklearn.ensemble import (GradientBoostingClassifier,
                               RandomForestClassifier)
from sklearn.model_selection import KFold

data = pd.read_csv('https://archive.ics.uci.edu/ml/' +
                  'machine-learning-databases/adult/adult.data',
                  header=None)
X = data[[2, 4, 10, 11, 12]]
Y = (data[14] == '>50K').astype(int)
print(X.shape)

def accuracy(P, Y):
    return (P == Y).sum() / Y.shape[0]

print(accuracy(0, Y))

def crossval(X, Y, model, n_splits):
    kf = KFold(n_splits, shuffle=True, random_state=0)
    scor = []
    for train, test in kf.split(X):
        clf = model.fit(X.iloc[train], Y.iloc[train])
        q = np.array([accuracy(P, Y.iloc[test])
                      for P in clf.staged_predict(X.iloc[test])])
    return np.array(scor).mean(axis=0)

```

```

clf = GradientBoostingClassifier(n_estimators=1000,
                                learning_rate=0.1, max_depth=3, random_state=0)
clf = clf.fit(X,Y)
score_train = np.array([accuracy(P,Y)
                        for P in clf.staged_predict(X)])
score_cv = crossval(X,Y,clf,5)

plt.figure(figsize=(6,4))
plt.plot(score_train)
plt.plot(score_cv)
plt.ylim((0.82,0.86))
plt.grid()
plt.show()

```

Поясним основные действия.

Используются данные известной задачи Adult из репозитория UCI. Таблица загружается непосредственно с сайта. Достоинством этой задачи является достаточно большой объём выборки (32561 строка).

Чтобы не загромождать пример кодированием категориальных признаков, выбираем из таблицы только числовые переменные. Последняя переменная — целевая. Она принимает два значения, которые закодируем как 0 и 1.

Заметим, что классы имеют умеренный дисбаланс, и если всегда приписывать класс 0, то получим точность прогноза 0,76. Это значение выводится на печать и является базовым уровнем точности, с которым будем сравнивать дальнейшие результаты.

В результате запуска фрагмента кода получим диаграмму, приведённую на рис. 4.5, где показаны зависимости точности решения на обучающей выборке и кроссвалидации от числа деревьев.

Как и следовало ожидать, точность на обучающей выборке (почти) монотонно растёт, а на кроссвалидации имеет один выраженный максимум.

Можно задать вопрос, как получается, что точность на обучающей выборке ведёт себя не вполне монотонно, хотя метод бустинга в силу своей конструкции потери на обучении может только уменьшать. Дело в том, что функция потерь в данном при-

мере не число ошибок, а `log loss`, и он действительно не возрастает (на обучении) с добавлением деревьев.

Заметим, что для получения данной зависимости не требуется запускать обучение заново для каждого числа деревьев. Достаточно обучить ансамбль один раз при максимальном числе деревьев, а затем воспользоваться функцией `staged_predict`, которая позволяет получать решения на подансамблях.

Однако сложность решения методом бустинга зависит не только от числа деревьев, но и от их размера. Поэтому при решении задач необходимо подбирать как число деревьев, так и глубину (или другой связанный с ней параметр).

Для подбора глубины дерева запустим следующий фрагмент кода (он является продолжением предыдущего).

```
scores = {}
for depth in [1, 2, 3, 4, 5]:
    print(depth)
    clf = GradientBoostingClassifier(n_estimators=1000,
                                     learning_rate=0.1, max_depth=depth,
                                     random_state=0)
    scores[depth] = crossval(X, Y, clf, 5)

plt.figure(figsize=(8,5))

for depth, sc in scores.items():
    plt.plot(sc, label='max_depth = %i, best = %.3g'
             %(depth, sc.max()))

plt.ylim((0.82, 0.84))
plt.legend()
plt.grid()
```

Глядя на рис. 4.6, можно сделать вывод, что оптимальным выбором будет глубина 3 при числе деревьев 500.

Однако для глубины 1 мы не достигли максимума точности, и следует увеличить число деревьев (также целесообразно увеличить скорость обучения). Если это проделать, то можно обнаружить, что при скорости обучения 0,2 и 6000 деревьев достигается точность 0,838.

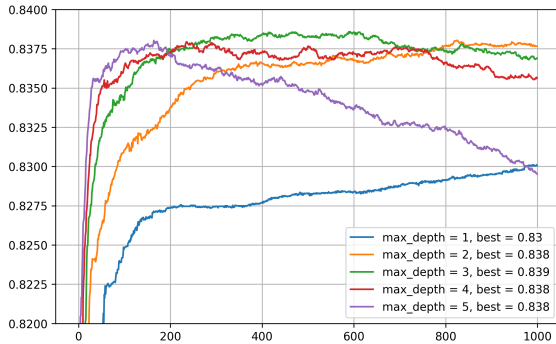


Рис. 4.6. Точность на кроссвалидации при разной глубине деревьев

Таким образом, при подборе параметров следует контролировать, что максимум точности «пройден».

Продолжение примера посвящено настройке случайного леса.

```
def crossval_pred(X,Y,model,n_splits):
    kf = KFold(n_splits, shuffle=True, random_state=0)
    q_cv = np.zeros(X.shape[0])
    P = np.zeros(Y.shape)
    for train, test in kf.split(X):
        clf = model.fit(X.iloc[train], Y.iloc[train])
        P[test] = clf.predict(X.iloc[test])
    return P
```

```
scores_rf = {'train': {}, 'cv': {}, 'oob': {}}
```

```
for depth in [1,2,3,5,7,10,15,20,30,40,50,60]:
    print(depth)
    model = RandomForestClassifier(n_estimators=200,
                                  max_depth=depth, random_state=0,
                                  oob_score=True).fit(X,Y)
    P = model.predict(X)
    scores_rf['train'][depth] = accuracy(P,Y)
    scores_rf['oob'][depth] = model.oob_score_
    P = model.oob_decision_function_
    P = (P[:,1]>P[:,0]).astype(int)
    scores_rf['oob'][depth] = accuracy(P,Y)
```

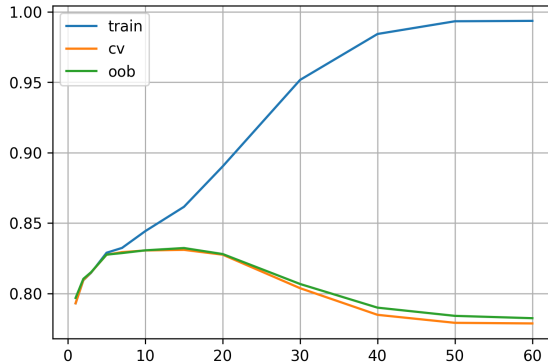


Рис. 4.7. Точность для RandomForest

```
P = crossval_pred(X,Y,model,5)
scores_rf['cv'][depth] = accuracy(P,Y)
```

```
pd.DataFrame(scores_rf).plot(figsize=(8,5))
plt.grid()
```

Здесь следует обратить внимание на функцию `crossval_pred`, которая делает прогнозы для всех объектов обучающей выборки методом кроссвалидации. Именно про подобный способ прогноза шла речь при описании метода стэкинга.

Для случайного леса увеличение числа деревьев может только улучшать результат, поэтому мы не будем подбирать этот параметр, а возьмём значение 200, которое обычно бывает «достаточным».

Подбирать будем только глубину деревьев.

На рис. 4.7 показаны зависимости точности от глубины.

Видим, что после глубины 50 точность перестаёт меняться. Это объясняется тем, что фактически более глубокие деревья не строятся, поскольку уже этой глубины достаточно, чтобы классифицировать выборку без ошибок.

Но почему тогда точность на обучении не достигает единицы? Всё дело в том, что каждое дерево строится не по всей выборке и не попавшие в подвыборку объекты могут классифицироваться с

ошибкой.

Видим, что оптимальная глубина равна 15. При этом достигаемая точность существенно выше, чем если глубину не ограничивать. Это наблюдение, очевидно, опровергает иногда встречающееся мнение, что метод случайного леса не требует подбора параметров.

Без чего метод RandomForest действительно позволяет обойтись — это без кроссвалидации, поскольку в него встроена возможность получать оценку out of bag, вычисляемую по объектам, которые не были выбраны процедурой bootstrap.

В приведённом коде значение «oob» присваивается дважды. Это не ошибка, поскольку значения в обоих вариантах одинаковы. Первый способ проще, а второй приведён для примера как более универсальный (на его основе можно вычислять другие критерии).

Заметим, что оценка «oob» отличается от «cv». Это можно объяснить тем, что оценка методом кроссвалидации является смещённой (незначительно). Подробнее это обсуждается в параграфе 6.4.

4.7.2 Визуализация нейронной сети

Попытаемся визуализировать структуру простейшей нейросети.

Для примера используем реализацию нейронных сетей из библиотеки sklearn. Выбор обусловлен исключительно тем, что эту библиотеку мы уже использовали, поэтому для запуска кода не потребуется установка новых библиотек. Для решения реальных практических задач нейросетевыми методами целесообразно использовать другие библиотеки.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.inspection import DecisionBoundaryDisplay

def plot_line(ab, c, k=2, ax=None):
```

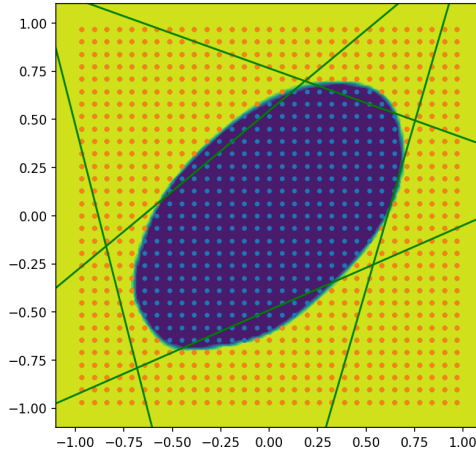


Рис. 4.8. Решение на основе нейросети

```

n = np.sqrt((ab*ab).sum())
a, b = ab/n
c = c/n
plt if ax is None else ax).plot([a*c-b*k, a*c+b*k],
    [b*c+a*k, b*c-a*k], c='green')

M = 31
X = np.array([[i, j] for i in range(M)
               for j in range(M)])
X = ((X+0.5)/M - 0.5) * 2
Y = (X[:,0]*X[:,0] - X[:,0]*X[:,1]
      + X[:,1]*X[:,1] > 0.35).astype(int)

K = 5
clf = MLPClassifier(hidden_layer_sizes=(K,),
                    activation='logistic', solver='lbfgs',
                    max_iter=500).fit(X, Y)

_, ax = plt.subplots(figsize=(6,6))
DecisionBoundaryDisplay.from_estimator(clf, X,
    response_method='predict_proba',
    grid_resolution=200, ax=ax, cmap='viridis')
ax.scatter(X[:,0], X[:,1], s=8)

```

```
ax.scatter(X[Y>0,0],X[Y>0,1],s=8)
```

```
for i in range(K):  
    plot_line(clf.coefs_[0][:,i],clf.intercepts_[0][i],  
             ax=ax)
```

```
ax.set_xlim((-1.1,1.1))
```

```
ax.set_ylim((-1.1,1.1))
```

Код начинается с функции, которая отрисовывает прямую по её каноническому уравнению. В качестве параметров уравнений прямых берутся коэффициенты скрытого слоя.

На рис. 4.8 мы можем видеть, как нейроны скрытого слоя аппроксимируют прямыми линиями границу между классами. Результирующая граница оказывается сглаженной благодаря свойствам выбранной функции активации.

Для наглядности вместо случайной выборки взята регулярная расстановка точек. Принадлежность объекта классу определяется в соответствии с принадлежностью точки заданному эллипсу.

Заметим, что получаемое решение сильно варьируется от запуска к запуску, и в части испытаний сеть обучается неудовлетворительно.

Контрольные вопросы

1. Что такое bagging? Какие две идеи лежат в основе метода случайного леса?
2. За счёт чего случайный лес превосходит по точности отдельное дерево, построенное по всей выборке?
3. Что определяет сложность модели в методе random forest?
4. В чём идея метода AdaBoost и по каким формулам пересчитываются веса?
5. Что определяет сложность модели в методе бустинга?

6. Какова архитектура многослойного персептрона и каков механизм его обучения?
7. Какие можно выделить основные идеи архитектур нейросетей?
8. Почему простое голосование может работать лучше, чем стэкинг?
9. В чём опасность domain shift при реализации стэкинга?
10. Почему за счёт ансамблирования нельзя неограниченно повышать точность решения?
11. Можно ли идеи бустинга и случайного леса использовать совместно?
12. За счёт чего бустинг превосходит по точности нейронные сети на табличных данных?

Глава 5

Градиентный бустинг

Идея бустинга заключается в том, чтобы каждый последующий базовый классификатор исправлял ошибки предыдущих.

Вторым принципом бустинга является то, что мы не пытаемся улучшить ранее построенные классификаторы, т. е. если базовая решающая функция построена, то она включается в композицию и в дальнейшем не изменяется.

5.1 Алгоритм AdaBoost

Увеличиваем веса объектов, на которых допущена ошибка. В методе AdaBoost решение строится в виде композиции

$$\lambda(x) = \text{sign}(\beta_T(x)), \quad \beta_T(x) = \sum_{t=1}^T \alpha_t \lambda_t(x),$$

где T – параметр, задающий количество базовых классификаторов.

Требуется построить базовые классификаторы $\lambda_t(x)$ и задать их веса α_t так, чтобы минимизировать число ошибок на обучающей выборке, т. е. минимизировать величину $\sum_{i=1}^N I(\lambda(x^i) \neq y^i)$.

Индикаторную функцию потерь можно выразить как

$$I(\lambda(x^i) \neq y^i) = I(\lambda(x^i)y^i \leq 0) = I(z_i \leq 0),$$

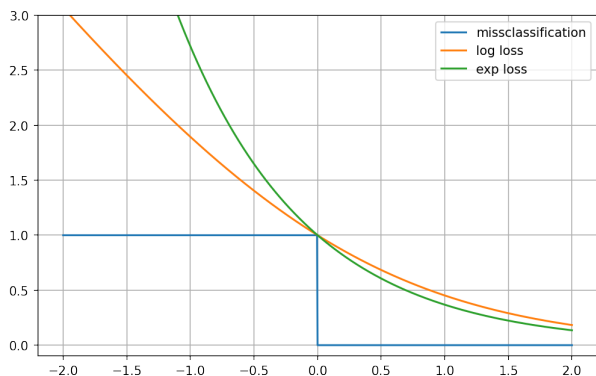


Рис. 5.1. Функции потерь для бустинга

где введено обозначение $z_i = \beta_T(x^i)y^i$.

Величина z_i — так называемый отступ (для i -го объекта выборки). Через неё удобно выражать функции потерь.

5.1.1 Непосредственная оптимизация экспоненциальной функции потерь

Попробуем теперь заменить индикаторную функцию потерь на такую функцию, для которой задача минимизации решалась бы аналитически и при этом новая функция аппроксимировала бы исходную.

В качестве такой функции подойдёт e^{-z} .

Заметим, что $I(z \leq 0) \leq e^{-z}$. Обе функции приведены на рис 5.1.

Видим, что минимизируя экспоненциальные потери, мы также минимизируем и индикаторную функцию потерь. Это, конечно, не означает, что минимумы для средних (на выборке) потерь также будут достигаться одновременно. Но это уже привычная ситуация, что вместо исходной функции потерь мы используем другую, которая качественно с ней согласуется (так мы использовали критерий Джини для ветвления дерева).

Итак, теперь мы будем минимизировать величину

$$\begin{aligned}
 R_T &= \sum_{i=1}^N e^{-z_i} = \sum_{i=1}^N e^{-y^i \beta_T(x^i)} = \\
 &= \sum_{i=1}^N e^{-y^i(\beta_{T-1}(x^i) + \alpha_T \lambda_T(x^i))} = \sum_{i=1}^N w_i^{T-1} e^{-y^i \alpha_T \lambda_T(x^i)},
 \end{aligned}$$

где $w_i^{T-1} = e^{-y^i \beta_{T-1}(x^i)}$.

Напомним, что оптимизацию проводим итеративно, т. е. на каждом шаге оптимизируем только очередной базовый классификатор, зафиксировав все предыдущие.

Рассмотрим итерационный шаг. Предположим, что $T-1$ классификаторов уже построено и на очередном шаге нужно оптимизировать только α_T и λ_T .

Отложим пока вопрос, как построить очередной классификатор λ_T и предположим, что какой-то λ_T у нас есть. Требуется подобрать α_T .

Вспомним, что λ_T принимает только значения $+1$ и -1 . Перегруппировав слагаемые, получим

$$R_T = W_+^T e^{-\alpha_T} + W_-^T e^{\alpha_T},$$

где

$$W_+^T = \sum_{i:y^i \lambda_T(x^i)=1} w_i^{T-1}, \quad W_-^T = \sum_{i:y^i \lambda_T(x^i)=-1} w_i^{T-1}.$$

Дифференцируем

$$\frac{\partial R_T}{\partial \alpha_T} = -W_+^T e^{-\alpha_T} + W_-^T e^{\alpha_T}.$$

Из условия $\frac{\partial R_T}{\partial \alpha_T} = 0$ находим

$$W_+^T e^{-\alpha_T} = W_-^T e^{\alpha_T}, \quad \alpha_T = \ln \sqrt{\frac{W_+^T}{W_-^T}}.$$

Подставив оптимальное значение α_T , находим минимальные потери

$$R_T^* = W_+^T \sqrt{\frac{W_-^T}{W_+^T}} + W_-^T \sqrt{\frac{W_+^T}{W_-^T}} = 2\sqrt{W_+^T W_-^T}.$$

Заметим, что потери на предыдущем шаге выражаются как

$$R_{T-1}^* = \sum_{i=1}^N w_i^{T-1} = W_+^T + W_-^T.$$

Найдём, насколько потери уменьшились

$$\begin{aligned} R_{T-1}^* - R_T^* &= W_+^T + W_-^T - 2\sqrt{W_+^T W_-^T} = \\ &= \left(\sqrt{W_+^T} - \sqrt{W_-^T} \right)^2. \end{aligned}$$

Видим, что чем больше различие W_+^T и W_-^T , тем сильнее уменьшаются потери. При этом не важно, какая из величин больше, но если больше W_-^T , то α_T получается отрицательным (фактически это означает, что мы инвертируем прогноз текущего классификатора).

Разберёмся, наконец, как же строить базовые классификаторы λ_T .

Если величины интерпретировать w_i^{T-1} как веса объектов выборки на шаге T , то W_+^T будет иметь смысл суммарного веса правильно классифицированных классификатором λ_T объектов, а W_-^T – суммарный вес неправильно классифицированных объектов.

И чтобы эти величины максимально различались, достаточно строить λ_T как решение обычной задачи классификации с весами.

Таким образом, AdaBoost разрешает в качестве базового метода классификации использовать абсолютно любую готовую реализацию любого метода, которая позволяет задавать веса объектов.

Если на каком-то шаге базовый алгоритм классифицирует выборку без ошибок, то α_T становится бесконечным, и AdaBoost не

может продолжить работу. Но бустинг и не имеет смысла, если базовый алгоритм не делает ошибок, поскольку задача и так решена. Тем не менее, мы можем избежать подобной ситуации, добавив к W_+^T и W_-^T небольшую одинаковую константу в качестве регуляризатора.

5.1.2 AdaBoost как градиентный бустинг

Как и ранее, мы будем минимизировать величину

$$R_T = \sum_{i=1}^N e^{-y^i \beta_T(x^i)},$$

но теперь используем представление

$$\beta_T(x^i) = \beta_{T-1}(x^i) + \xi_i,$$

где ξ_i будем считать свободной переменной, по которой и будем вести минимизацию.

На самом деле, ξ_i – это прогноз базового классификатора на объекте x^i , и он не может быть произвольным. Но пока будем считать, что может.

Заметим, что в этом случае минимум даже не существует. Однако, это не мешает нам двигаться градиентным методом в сторону уменьшения потерь.

Движение против градиента означает, что

$$\xi_i = -\alpha_T \frac{\partial R_T}{\partial \xi_i},$$

где α_T – величина шага. Если раньше мы оптимизировали его в первую очередь, то теперь сделаем это позже.

Теперь нам нужно добиться, чтобы базовый метод давал решение λ_T , приближающее антиградиент. Это можно сделать, если на базовом уровне решать задачу регрессии.

Но чтобы получить именно AdaBoost, на базовом уровне мы должны поставить задачу классификации, когда λ_T принимает значения $+1, -1$.

Решение λ_T можно интерпретировать как вектор, задающий направление, вдоль которого будет смещаться композиция. А чтобы уменьшалась функция потерь, это смещение должно осуществляться примерно вдоль антиградиента.

Скорость уменьшения потерь определяется величиной скалярного произведения направления сдвига и антиградиента, а именно

$$-\sum_{i=1}^N \frac{\partial R_T}{\partial \xi_i} \lambda_T(x^i) \rightarrow \max.$$

Вычислим градиент для экспоненциальных потерь

$$\frac{\partial R_T}{\partial \xi_i} = -y^i e^{-y^i \beta_{T-1}(x^i)} = -y^i w_i^{T-1}$$

и подставим в скалярное произведение. Получим задачу максимизации

$$\sum_{i=1}^N y^i w_i^{T-1} \lambda_T(x^i) \rightarrow \max.$$

Видим, что это ровно та же задача классификации, которая решается базовым методом в алгоритме AdaBoost.

Этим мы показали, что AdaBoost является частным случаем градиентного бустинга, если использовать экспоненциальную функцию потерь и если сводить поиск антиградиента к задаче классификации.

Последнее является очень важным уточнением. Так как, например, если взять реализацию градиентного бустинга из scikit-learn и выбрать в ней экспоненциальные потери, то в результате AdaBoost мы не получим, поскольку градиентный бустинг из scikit-learn на базовом уровне использует деревья регрессии, а не классификации.

Кастомизацией функции потерь в XGBoost также нельзя получить алгоритм, совпадающий с AdaBoost.

5.1.3 Сходимость процесса бустинга

Если бустинг не останавливать, то он будет стремиться оценить функцию условной вероятности. Если в точке пространства находится один объект выборки, то эмпирическая условная вероятность соответствующего класса в этой точке равна 1. Если в одной точке находятся объекты разных классов, то эмпирическая условная вероятность есть доля объектов класса в точке.

Бустинг приближает вероятность через логистическую функцию. Логистическая функция не может принять значения 0 или 1, а только стремится к ним в пределах. Поэтому, если эмпирическая условная вероятность принимает эти значения, то бустинг будет неограниченно добавлять деревья в ансамбль. Если условные вероятности нигде не равны 0 или 1, то бустинг в некотором смысле останавливается сам (веса деревьев стремятся к 0).

В реальных выборках довольно редко встречаются объекты с полностью совпадающими значениями всех переменных, т.е. попадающие в одну точку пространства. Для непрерывных переменных вероятность такого события формально вообще нулевая (на практике, конечно, ненулевая, поскольку реальные значения представлены конечным числом разрядов). Поэтому для исследования свойств бустинга имеет смысл сконструировать «искусственную выборку», поместив в одну и ту же точку пространства объекты всех классов с некоторыми весами.

Условную вероятность $g(x) = P(y = 1 | x)$ представим как находящиеся в точке x два объекта: класса 1 с весом $w_0 g(x)$ и класса -1 с весом $w_0(1 - g(x))$.

В результате выполнения бустинга вес первого объекта станет равным

$$w^{+1}(x) = w_0 g(x) \cdot A e^{-\beta(x)},$$

где константа A есть произведение всех нормировочных множителей.

Конечный вес второго объекта есть

$$w^{-1}(x) = w_0(1 - g(x)) \cdot A e^{\beta(x)}.$$

Если приравнять веса объектов, то получим

$$g(x) = \frac{1}{1 + e^{-2\beta(x)}} = \sigma(2\beta(x)).$$

Мы вывели формулу для оценки функции условной вероятности на основе ансамбля.

Если веса объектов сравниваются в каждой точке признакового пространства, то точность любого нового базового классификатора будет равна 0,5, а вес этого классификатора в ансамбле будет нулевым. Такую ситуацию будем называть самостоятельной остановкой бустинга.

Заметим, что бустинг может «остановиться» (перейти в режим околонулевых весов) не только в случае, если веса объектов в каждой точке сравниваются. Если сложность базовых классификаторов ограничена (например, мы ограничили глубину деревьев), то возможны и иные ситуации, когда новые классификаторы не могут дать точность (заметно) выше 0,5. Один такой пример мы рассмотрим далее.

5.2 Градиентный бустинг

Фактически мы уже познакомились с градиентным бустингом, когда использовали его идею для вывода алгоритма AdaBoost. Рассмотрим теперь его в более общем виде.

Итак, у нас есть некоторая функция потерь $L(\beta(x), y)$, и мы хотим минимизировать средние потери на выборке

$$R(S, \beta_T) = \sum_{i=1}^N L(\beta_T(x^i), y^i).$$

Пусть мы уже построили $T - 1$ базовых решающих функций, и требуется построить очередную функцию $\lambda_T(x)$.

Обозначим $\xi_i = \lambda_T(x^i)$ и будем считать ξ_i свободными переменными.

Попробуем провести минимизацию потерь методом градиентного спуска (в пространстве переменных ξ_i). Вычислим производную

$$\frac{\partial R(S, \beta_T)}{\partial \xi_i} = \frac{\partial L(\beta_T(x^i), y^i)}{\partial \xi_i} = \frac{\partial L(\beta_{T-1}(x^i) + \xi_i, y^i)}{\partial \xi_i}.$$

Обозначим через ω^T значение антиградиента в точке $\xi = 0$. Имеем для i -й компоненты вектора ω^T

$$\omega_i^T = - \left. \frac{\partial R(S, \beta_T)}{\partial \xi_i} \right|_{\xi_i=0} = - \left. \frac{\partial L(u, y^i)}{\partial u} \right|_{u=\beta_{T-1}(x^i)}.$$

Компоненты антиградиента известны только для точек выборки. Для обобщения на всё пространство требуется применить базовый метод классификации (для краткости будем говорить о классификации, однако все рассуждения справедливы для общей задачи построения решающей функции).

Итак, очередное базовое решение $\lambda_T(x)$ в точках выборки должно быть близким к ω^T .

Значения $\lambda_T(x)$ на выборке удобно представлять в виде вектора. Обозначим через $\tilde{\lambda}_T$ вектор с компонентами $\lambda_T(x^i)$.

Получается, что критерием качества для базового метода должна быть близость векторов $\tilde{\lambda}_T$ и ω^T .

Очевидным кажется определить близость векторов через евклидову норму их разности. В этом случае базовый метод должен решать задачу регрессии с функцией потерь в виде квадрата отклонения.

Однако вспомним, что бы хотим вести градиентный спуск. В этом случае нам важно, насколько точно мы можем выдерживать направление градиента. А точность направления, очевидно, определяется углом между вектором градиента и вектором, в направлении которого мы фактически двигаемся.

Минимизация угла между векторами эквивалентна максимизации скалярного произведения векторов при фиксированной норме. Таким образом, базовый метод может решать задачу регрес-

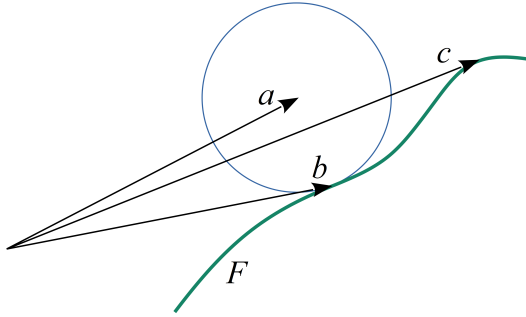


Рис. 5.2. Различные приближения к вектору градиента

сии с критерием

$$\begin{cases} \tilde{\lambda}_T \omega^T \rightarrow \max \\ (\tilde{\lambda}_T)^2 = 1 \end{cases}.$$

Может показаться, что задачи минимизации квадрата разности и максимизации скалярного произведения эквивалентны. При фиксированной норме искомого вектора это действительно так, поскольку

$$(\tilde{\lambda}_T - \omega^T)^2 = (\tilde{\lambda}_T)^2 + (\omega^T)^2 - 2\tilde{\lambda}_T \omega^T.$$

Если же норму не фиксировать, то результаты могут существенно различаться.

На рис. 5.2 приведён условный пример, где a выступает в роли вектора градиента, к которому мы ходим приблизиться, а F – множество решений, которые способен построить базовый алгоритм. Видим, что с точки зрения минимума квадрата отклонения ближайшим к a из допустимых решений будет вектор b , а минимальный угол с вектором градиента обеспечивает вектор c .

Заметим, что базовый метод может решать задачу классификации, даже если сам бустинг решает задачу регрессии.

Действительно, задача

$$\begin{cases} \tilde{\lambda}_T |\omega^T| \operatorname{sign}(\omega^T) \rightarrow \max \\ (\tilde{\lambda}_T)_i \in \{-1, 1\} \end{cases}$$

является задачей классификации с целевой функцией $\operatorname{sign}(\omega^T)$ и весами $|\omega^T|$.

И поставить задачу в такой форме мы можем для любой исходной задачи.

Следует отметить важный момент, на который редко акцентируют внимание.

В методах бустинга мы имеем в общем случае четыре различные функции потерь:

- ✧ исходная функция потерь, которую мы должны минимизировать (например, число ошибок классификации);
- ✧ функция потерь, которую мы решили минимизировать вместо исходной (например, экспоненциальная функция потерь для AdaBoost);
- ✧ функция потерь, которую должен минимизировать базовый алгоритм (например, для AdaBoost это ошибка классификации с учётом весов объектов);
- ✧ функция потерь, которую мы будем минимизировать базовым алгоритмом вместо той, что получилась в предыдущем пункте (например, для AdaBoost на деревьях решений разумным выбором будет Gini impurity или log loss).

5.3 Метод XGBoost

Данный метод использует всего одну функцию потерь.

Как и ранее, задана функция потерь $L(\beta(x), y)$, и мы хотим минимизировать средние потери на выборке

$$R(S, \beta_T) = \sum_{i=1}^N L(\beta_T(x^i), y^i).$$

Решение на текущем шаге выражается через решение на предыдущем шаге как

$$\beta_T(x^i) = \beta_{T-1}(x^i) + \xi_i.$$

Найдём первую и вторую производные функции потерь в точке $\xi = 0$

$$g_i = \left. \frac{\partial R(S, \beta_T)}{\partial \xi_i} \right|_{\xi_i=0} = \left. \frac{\partial L(u, y^i)}{\partial u} \right|_{u=\beta_{T-1}(x^i)},$$

$$h_i = \left. \frac{\partial^2 R(S, \beta_T)}{(\partial \xi_i)^2} \right|_{\xi_i=0} = \left. \frac{\partial^2 L(u, y^i)}{(\partial u)^2} \right|_{u=\beta_{T-1}(x^i)}.$$

Разложим функцию потерь в ряд Тейлора до второго порядка

$$L(\beta_T(x^i, y^i)) = L(\beta_{T-1}(x^i, y^i)) + g_i \xi_i + \frac{1}{2} h_i \cdot (\xi_i)^2.$$

Метод XGBoost в качестве базового метода использует деревья решений.

Для построения дерева нужно иметь критерий ветвления, для чего требуется уметь оценивать качество любых вариантов дерева. А поскольку критерий для дерева — это сумма критериев по вершинам, достаточно понять, как оценить решение в произвольной вершине дерева.

Дерево решений в каждой вершине приписывает одинаковые значения для всех объектов, попавших в эту вершину. Обозначим этот прогноз через ζ .

Запишем потери в вершине через разложение в ряд

$$R(J, \zeta) = \sum_{i \in J} L(\beta_{T-1}(x^i, y^i)) + g_i \zeta + \frac{1}{2} h_i \zeta^2,$$

где J — множество номеров объектов, попавших в вершину.

Преобразуем к виду

$$R(J, \zeta) = G(J) \cdot \zeta + \frac{1}{2} H(J) \cdot \zeta^2 + C(J),$$

где

$$G(J) = \sum_{i \in J} g_i, \quad H(J) = \sum_{i \in J} h_i,$$
$$C(J) = \sum_{i \in J} L(\beta_{T-1}(x^i, y^i)).$$

Из условия $\frac{\partial R(J, \zeta)}{\partial \zeta} = 0$ находим оптимальный прогноз в вершине

$$\zeta^* = -\frac{G(J)}{H(J)}.$$

Потери в вершине при оптимальном прогнозе составят

$$R^*(J) = R(J, \zeta^*) =$$
$$= -G(J) \cdot \frac{G(J)}{H(J)} + \frac{1}{2}H(J) \cdot \left(-\frac{G(J)}{H(J)}\right)^2 + C(J) =$$
$$= -\frac{1}{2} \cdot \frac{G^2(J)}{H(J)} + C(J).$$

Заметим, что сумма $C(J)$ по всем вершинам не зависит от структуры дерева (поскольку вершины в сумме покрывают всю выборку), поэтому критерий ветвления от $C(J)$ не зависит.

Мы получили, что критерий ветвления при построении дерева должен максимизировать сумму по всем вершинам величин

$$\frac{G^2(J)}{H(J)}.$$

В критерий ветвления в методе XGBoost также входит слагаемое, отвечающее за регуляризацию. Чтобы не перегружать выкладки, мы его не использовали.

Найдём теперь конкретные выражения полученного критерия ветвления для некоторых функций потерь (их графики приведены на рис. 5.1, стр. 131).

Начнём с \log loss, которая представляет собой функцию правдоподобия и требует решения в виде оценки вероятности класса.

Используя композицию деревьев, мы можем оценить условную вероятность в форме логистической функции от суммы прогнозов, т. е.

$$g(x) = \frac{1}{1 + e^{-\beta(x)}}.$$

Выразим теперь логарифмическую функцию потерь

$$\begin{aligned} L(y, g(x)) &= \begin{cases} -\ln g(x), & y = 1 \\ -\ln(1 - g(x)), & y = -1 \end{cases} = \\ &= \ln(1 + e^{-y\beta(x)}) = -\ln \sigma(y\beta(x)). \end{aligned}$$

Вычислим производные от функции потерь

$$\begin{aligned} \frac{\partial -\ln \sigma(y\beta(x))}{\partial \xi} &= \frac{\partial \ln(1 + e^{-y\beta(x)})}{\partial \xi} = \\ &= -\frac{e^{-y\beta(x)}}{1 + e^{-y\beta(x)}} y = -y\sigma(-y\beta(x)), \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 -\ln \sigma(y\beta(x))}{(\partial \xi)^2} &= \frac{\partial -y\sigma(-y\beta(x))}{\partial \xi} = \\ &= y^2 \frac{e^{y\beta(x)}}{(1 + e^{y\beta(x)})^2} = y^2 \sigma(y\beta(x))\sigma(-y\beta(x)) = \sigma(\beta(x))\sigma(-\beta(x)). \end{aligned}$$

Мы учли, что $y^2 = 1$.

Получаем

$$g_i = \left. \frac{\partial L(u, y^i)}{\partial u} \right|_{u=\beta_{T-1}(x^i)} = -y^i \sigma(-y^i \beta_{T-1}(x^i)).$$

$$h_i = \left. \frac{\partial^2 L(u, y^i)}{(\partial u)^2} \right|_{u=\beta_{T-1}(x^i)} = \sigma(\beta_{T-1}(x^i))\sigma(-\beta_{T-1}(x^i)).$$

Для экспоненциальных потерь

$$g_i = \left. \frac{\partial e^{-y^i u}}{\partial u} \right|_{u=\beta_{T-1}(x^i)} = -y^i e^{-y^i \beta_{T-1}(x^i)} = -y^i w_i.$$

$$h_i = \frac{\partial^2 e^{-y^i u}}{(\partial u)^2} \Big|_{u=\beta_{T-1}(x^i)} = e^{-y^i \beta_{T-1}(x^i)} = w_i.$$

Величину w_i будем считать весом i -го объекта выборки (на текущем шаге). Для отдельной вершины дерева (как и ранее, J – множество индексов объектов в этой вершине) просуммируем веса объектов отдельно по классам

$$W_{-1}(J) = \sum_{i \in J} w_i I(y^i = -1) \quad W_1(J) = \sum_{i \in J} w_i I(y^i = 1).$$

Для критерия ветвления

$$\begin{aligned} \frac{G^2(J)}{H(J)} &= \frac{(W_{-1}(J) - W_1(J))^2}{W_{-1}(J) + W_1(J)} = \frac{(W(J) - 2W_1(J))^2}{W(J)} = \\ &= W(J) - 4W_1(J) + 4\frac{(W_1(J))^2}{W(J)}, \end{aligned}$$

где $W(J) = W_{-1}(J) + W_1(J)$.

Если просуммировать $W(J)$ или $W_1(J)$ по всем вершинам дерева, то результат не будет зависеть от собственно дерева, поэтому эти слагаемые не влияют на выбор ветвления.

Таким образом, критерий ветвления будет основан на сумме величин

$$\frac{(W_1(J))^2}{W(J)}.$$

Полученный критерий ветвления оказывается полностью эквивалентным критерию Gini impurity. Действительно, для Gini impurity качество дерева определяется как сумма величин

$$\begin{aligned} W_{-1}(J) \cdot \frac{W_{-1}(J)}{W(J)} + W_1(J) \cdot \frac{W_1(J)}{W(J)} &= \\ = \frac{(W(J) - W_1(J))^2}{W(J)} + \frac{(W_1(J))^2}{W(J)} &= \\ = W(J) - 2W_1(J) + 2\frac{(W_1(J))^2}{W(J)}. \end{aligned}$$

Получается, что метод XGBoost при экспоненциальной функции потерь будет на каждом шаге строить дерево по критерию Gini impurity.

Но ведь данный критерий используется для построения деревьев классификации, а в методе XGBoost всегда строится дерево регрессии. Но противоречия здесь нет.

Дело в том, что критерий ветвления — это функция от объектов выборки, попавших в вершину, и его можно уже не рассматривать как функцию от оптимального прогноза, поскольку последний — функция выборки.

Так что мы действительно имеем ситуацию, что дерево регрессии строится по критерию Gini impurity.

Но мы можем представить дерево регрессии из XGBoost, как дерево классификации, в каждой вершине которого бинарный прогноз умножается на некоторый коэффициент $\alpha(J)$.

Получается, что XGBoost при экспоненциальной функции потерь очень похож на AdaBoost, но имеет принципиальное отличие в том, что каждое дерево входит в композицию не просто с весом α_t , а с весом, который для каждой вершины дерева свой. При этом все веса дополнительно умножаются на некоторый коэффициент, который называется *скоростью обучения*.

Заметим, что и в AdaBoost ничто не запрещает ввести параметр скорости обучения.

Также в AdaBoost мы можем использовать не только Gini impurity.

Таким образом, XGBoost при экспоненциальной функции потерь вовсе не эквивалентен методу AdaBoost на деревьях, но имеет много общего.

Заметим, что библиотека XGBoost является очень гибкой, в частности позволяет реализовать также метод случайного леса, а также гибридные варианты ансамбля (например, бустинг со случайным выбором подмножеств переменных).

5.4 Свойства бустинга

5.4.1 Представление бустинга

Градиентный бустинг строит композицию $\beta(x)$, минимизируя функцию потерь на выборке. Ранее мы для наивного байесовского классификатора получили выражение в виде логистической регрессии

$$g(x) = \sigma \left(u_0 + \sum_{j=1}^n u_j \sigma^{-1}(g_j(x_j)) \right)$$

при $u_0 = (n-1)(\ln p - \ln(1-p))$, $u_j = 1$.

Обобщим это выражение, считая веса свободными параметрами и допуская произвольные оценочные функции. Получим

$$g(x) = \sigma \left(u_0 + \sum_{j=1}^n u_j s_j(x_j) \right).$$

Бустинг на пороговых классификаторах («пнях») является разновидностью обобщённого наивного байесовского классификатора.

Действительно, каждая $\lambda_t(x)$ в композиции

$$\beta(x) = \sum_{t=1}^T \alpha_t \lambda_t(x)$$

зависит только от одной переменной X_{j_t} , поэтому после группировки слагаемых выражение можно привести к виду

$$2\beta(x) = \sum_{j=1}^n u_j s_j(x_j),$$

где $s_j(x_j) = \sum_{j_t=j} \alpha_t \lambda_t(x)$, а $u_j = 2$. Мы попросту сгруппировали деревья по переменным, которые в них входят. Подставив полученное представление в выражение для $g(x)$, получим искомый вид.

Модель можно естественным образом обобщить по аналогии с рядом Бахадура, включив возможность учитывать зависимости между переменными, последовательно добавляя парные зависимости, зависимости в тройках и т. д.

$$g(x) = \sigma \left(u_0 + \sum_{j=1}^n u_j s_j(x_j) + \sum_{j,k} u_{jk} s_{jk}(x_j, x_k) + \right. \\ \left. + \sum_{j,k,l} u_{jkl} s_{jkl}(x_j, x_k, x_l) + \dots \right).$$

Здесь мы перебираем все сочетания переменных и для каждой комбинации выбираем деревья, которые используют именно эти переменные.

5.4.2 Запуск бустинга на распределениях

Методы машинного обучения строят решение по выборке. Но все они могут построить решающую функцию и непосредственно по распределениям. Даже метод kNN можно применить на распределениях, если вместо количества «соседей» задавать вероятность попадания в окрестность. Методы, минимизирующие средние потери на выборке, могут так же минимизировать ожидаемые потери на распределениях.

Вместе с тем существующие программные реализации методов машинного обучения ожидают на вход выборку и не позволяют подать на вход распределения. Однако мы всегда можем из распределения сгенерировать выборку такого большого размера, что результат на выборке будет мало отличаться от результата на распределении.

На самом деле, мы можем не генерировать выборку случайно, а аппроксимировать распределение выборкой. Только формально это будет уже не выборкой, а множеством точек, отражающем свойства распределения. Но мы эти точки можем подать на вход методу.

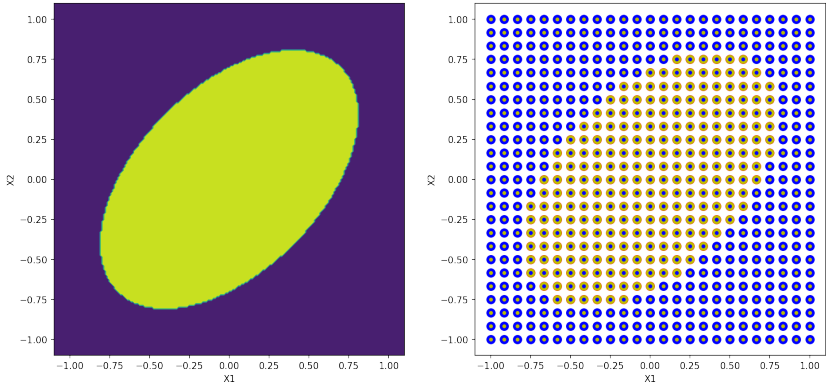


Рис. 5.3. Задуманная вероятностная модель (слева) и её аппроксимация «выборкой» (справа)

Рассмотрим пример. Зададим на переменных X_1 и X_2 равномерное распределение в квадрате $[-1, 1]^2$, а условная вероятность $P(y|x)$ пусть зависит от того, попала ли точка x в задуманный эллипс (рис. 5.3, слева). Внутри эллипса больше вероятность класса +1, снаружи — класса -1. Условная вероятность «чужого» класса в каждой точке равна 0,1.

Теперь выберем в квадрате равномерную сетку и в каждый узел сетки поставим две точки, по одной каждого класса. Точкам зададим веса (методы классификации позволяют передавать веса объектов). При этом вес каждого объекта зададим равным вероятности его класса (в соответствующей точке). На рис. 5.3 справа жёлтые точки (соответствуют классу +1) внутри эллипса имеют вес 0,9, у синих вес 0,1. Снаружи эллипса — наоборот.

На рис. 5.4 приведены решающие функции (вероятностный прогноз), полученные для сгенерированной выборки градиентным бустингом на деревьях глубины 2. Видим, что решающая функция приближает задуманную функцию условной вероятности. Точность приближения растёт с ростом числа деревьев.

На рис. 5.5 приведены решающие функции, полученные градиентным бустингом на деревьях глубины 1.

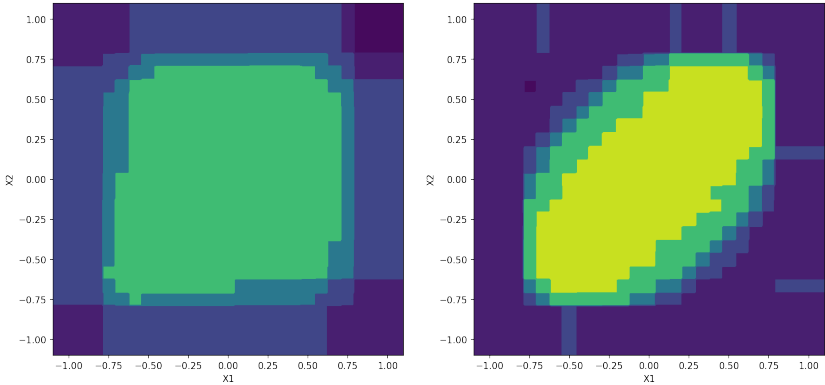


Рис. 5.4. Глубина дерева 2, решения на основе 15 (слева) и 1000 (справа) деревьев

Бустинг на деревьях минимальной глубины не может отразить зависимости между переменными и строит приближение как обобщённый наивный байесовский классификатор.

Вместо эллипса получаем круг. Это объясняется, что уравнение эллипса

$$a \cdot (x_1)^2 + b \cdot x_1 x_2 + c \cdot (x_2)^2 = 1$$

содержит слагаемое, в котором участвуют обе переменные. Ансамбль деревьев, каждое из которых использует только одну переменную, не способен аппроксимировать это слагаемое. А без него получаем уравнение эллипса, оси которого параллельны осям координат. Но в нашем примере модель симметрична по переменным, поэтому получаем круг.

5.4.3 Модельный пример

Рассмотрим пример одномерной задачи.

Зададим на переменной X_1 равномерное распределение на интервале $[-1, 1]$, а условная вероятность $P(y = 1 | x)$ пусть задаётся функцией, изображённой на рис. 5.6.

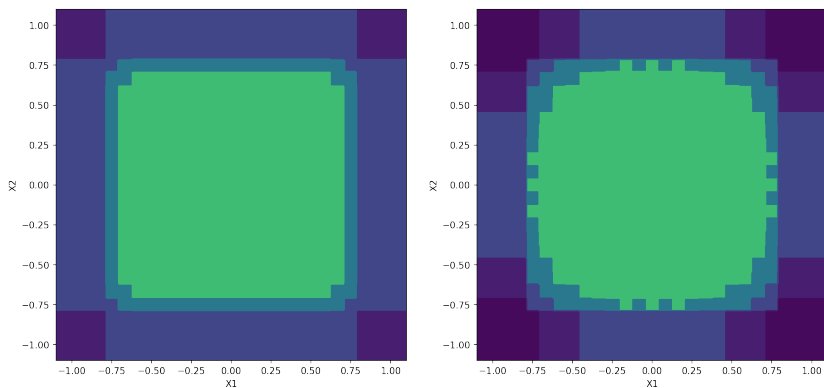


Рис. 5.5. Глубина дерева 1, решения на основе 15 (слева) и 1000 (справа) деревьев

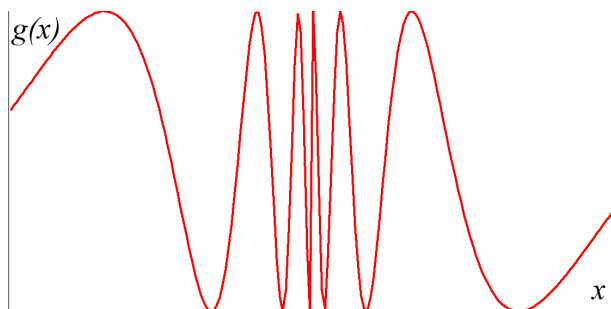


Рис. 5.6. Функция условной вероятности для одномерной задачи

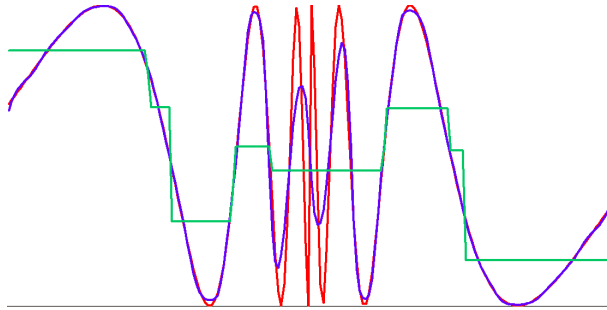


Рис. 5.7. AdaBoost 10 итераций

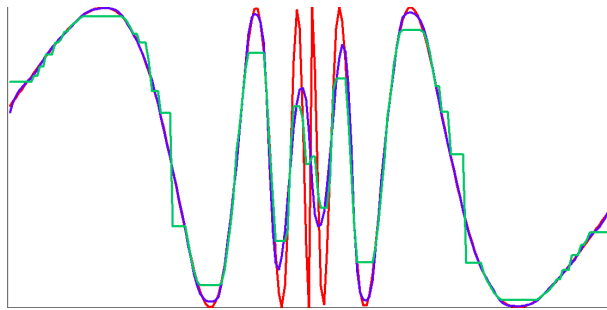


Рис. 5.8. AdaBoost 100 итераций

Сгенерируем выборку из задуманного распределения и запустим бустинг на деревьях глубины 1.

На рис. 5.7 приведено решение для 10 деревьев (зелёная кривая).

Для сравнения приведена аппроксимация сплайном (синяя кривая). Использован кубический сплайн на 20 интервалов.

Видим, что бустинг сначала аппроксимирует зависимость на областях, где она меняется медленно (находит «надёжные» закономерности), и только при большом числе деревьев начинает учитывать «быстрые» изменения.

5.4.4 Понятие отступа

Иногда для обоснования бустинга вводится понятие отступа. Отступ есть

$$\theta(x, y) = \frac{y\beta(x)}{\varkappa}, \quad \varkappa = \sum_{t=1}^T \alpha_t.$$

Были доказаны утверждения [45], согласно которым чем больше $\theta(x, y)$, тем выше обобщающая способность модели.

Однако отсюда не следует, что с ростом числа деревьев обобщающая способность должна всегда расти, поскольку с ростом числа деревьев в выражении для $\theta(x, y)$ растёт не только числитель, но и знаменатель.

Из-за нормировки в виде \varkappa сложность композиции влияет на оценку риска.

5.5 Выводы

Известно такое неформальное понятие, как «проклятие» размерности, смысл которого заключается в том, что чем выше размерность задачи, тем труднее её решать. В случае машинного обучения это выражается также в том, что чем больше зависимых переменных — тем больше требуемый объём выборки.

Для случая независимых переменных «проклятие» размерности превращается в преимущество: чем больше независимых переменных — тем меньше требуемая выборка. С увеличением числа независимых переменных качество решения (для адекватного метода) только растёт.

Бустинг и случайный лес существенно различаются в настройке. Для бустинга параметры сложности — глубина дерева и количество деревьев. Для random forest сложность не увеличивается с ростом числа деревьев. Как правило, бустинг использует деревья меньшей глубины.

Как правило, бустинг достигает лучшего качества в сравнении с random forest.

Выводы.

- ✧ Важнейшей причиной эффективности бустинга является использование эффекта независимости (переменных, подпространств, моделей).
- ✧ Бустинг на пороговых классификаторах является разновидностью непараметрической логистической регрессии, также его можно считать разновидностью (существенно обобщённого) наивного байесовского классификатора.
- ✧ Бустинг реализует «удачный» вариант непараметрической аппроксимации условной вероятности.
- ✧ Ансамбли на деревьях позволяют работать в разнотипном пространстве, а также с пропусками.
- ✧ Бустинг на деревьях — один из лучших методов машинного обучения.
- ✧ Идеи бустинга могут быть реализованы и в нейросетях.

Для задач с табличными данными бустинг может считаться методом для выбора по умолчанию.

5.6 Примеры кода

Приведенный ниже код иллюстрирует приближение функции условной вероятности методом бустинга и нейронной сетью. Для наглядности рассмотрена одномерная задача. Вместо генерирования случайной выборки делается равномерная расстановка точек, $N = 201$, для каждой точки создаются объекты обоих классов, веса объектов соответствуют задуманной условной вероятности (кривая «probability»).

В примере также показана возможность сохранения рисунка в файл.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
```

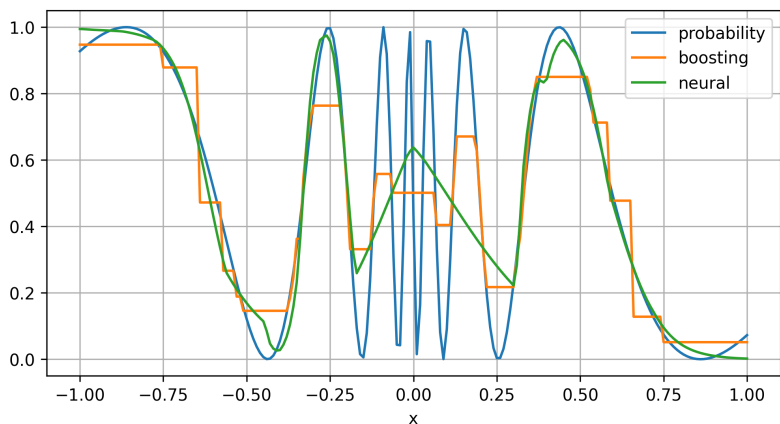


Рис. 5.9. Оценивание условной вероятности бустингом и многослойным персептроном

```

def gen_data(N):
    x = np.zeros(2*N)
    x[::2] = np.linspace(-1.0,1.0,N)
    x[1::2] = x[::2]
    y = np.ones(2*N, dtype=int)
    y[::2] *= -1
    p = np.sin(5/(x+0.2*(x>=0)-0.2*(x<0)))
    w = p * y
    y[::2] = 0
    return pd.DataFrame({'x': x, 'y': y,
                        'w': (w + 1)/2, 'probability': (p + 1)/2})

def predict(clf, df):
    p = clf.fit(df[['x']], df['y'], sample_weight=df['w'])
    .predict_proba(df[['x']])
    r = np.zeros(df.shape[0])
    pp = (p[::2,0] + p[1::2,0])/2
    r[::2] = pp
    r[1::2] = 1.0 - pp
    return r

df = gen_data(201)

```

```

clf = GradientBoostingClassifier(n_estimators=50,
                                learning_rate=1.0, max_depth=1, random_state=0)
df['boosting'] = predict(clf, df)

clf = MLPClassifier(hidden_layer_sizes=(100,15), alpha=0.0,
                    max_iter=100, random_state=0, solver='lbfgs')
df['neural'] = predict(clf, df)

df.iloc[1::2].set_index('x')[['probability', 'boosting',
                              'neural']].plot(figsize=(8,4), grid='on')
plt.show()

```

Результат выполнения кода приведён на рис. 5.9. Гиперпараметры методов подобраны так, чтобы средняя точность решения была близкой. Кривые наглядно показывают различие методов.

Контрольные вопросы

1. В чём заключается идея градиентного бустинга?
2. Можно ли AdaBoost реализовать как градиентный бустинг?
3. Почему для задачи классификации градиентный бустинг может использовать деревья регрессии?
4. Когда процесс бустинга сходится?
5. Как бустинг оценивает вероятности?
6. Что такое отступ в методе опорных векторов и в бустинге?
7. Что значит «запустить» метод на распределениях? Любой ли метод можно применить к распределениям?

Глава 6

Оценивание качества решений

В данной главе нас будут интересовать способы оценки качества построенного решения на основе конечного набора данных (выборки).

6.1 Постановка задачи

6.1.1 Точечные оценки риска

Точно вычислить качество построенной решающей функции возможно только, если известны распределения, т.е. полная вероятностная модель данных. Имея только выборку, можно построить лишь оценки качества (для определённости под качеством можно понимать вероятность ошибочной классификации).

Самое простое — это подставить в решающую функцию обучающую выборку. Полученная при этом доля ошибок — это эмпирический риск.

Легко понять, что в общем случае (без дополнительных ограничений) по эмпирическому риску нельзя оценить вероятность ошибочной классификации. Действительно, существуют (и активно используются) методы, использующие универсальные классы

решающих функций (например, деревья решений). Если в таких методах не задать критерий раннего останова, то они всегда обеспечат безошибочную классификацию обучающей выборки (если в ней нет объектов с одинаковыми значениями всех переменных). А если событие происходит всегда (или почти всегда, т. е. с вероятностью 1), то оно не несёт полезной информации.

Представим, что два класса имеют одинаковое непрерывное распределение в пространстве переменных и безусловные вероятности классов равны. В этом случае вероятность ошибочной классификации для любой решающей функции равна 0,5. Но метод неограниченной сложности (например 1NN) с вероятностью 1 даст нулевой эмпирический риск.

Однако если использовать характеристику сложности метода, то оценивать качество решения по эмпирическому риску всё же можно. Этот подход будет подробно рассмотрен в следующей главе.

Данная глава посвящена подходам, так или иначе эксплуатирующим идею отложенной выборки.

Точечные оценки риска:

- ✧ эмпирический риск;
- ✧ контрольная выборка;
- ✧ скользящий экзамен;
- ✧ bootstrap;
- ✧ out of bag;
- ✧ другие статистики.

6.1.2 Формальная постановка

Пусть X – пространство значений переменных, используемых для прогноза, а Y – пространство значений прогнозируемых переменных.

Предположим, что все переменные являются случайными величинами с некоторой неизвестной функцией совместного распределения $F_c(x, y)$.

В самом общем виде это можно формализовать, введя \mathcal{C} – множество всех вероятностных мер на заданной σ -алгебре подмножеств множества $D = X \times Y$. При каждом $c \in \mathcal{C}$ имеем вероятностное пространство: $\langle D, \mathcal{B}, P_c \rangle$, где \mathcal{B} – σ -алгебра, P_c – вероятностная мера.

Индекс c здесь — это формально введённый параметр. Дело в том, что все функции распределения привычно обозначать одной и той же буквой F , а вероятности — буквой P , и чтобы различать распределения требуется индекс.

Параметр c будем называть *стратегией природы*.

Решающей функцией¹ (алгоритмом классификации) называется соответствие $\lambda: X \rightarrow Y$, причём λ должна быть измеримой функцией.

Качество принятого решения оценивается заданной функцией потерь $\mathcal{L}: Y^2 \rightarrow [0, \infty)$.

Под риском будем понимать средние потери:

$$R(c, \lambda) = E\mathcal{L}(y, \lambda(x)) = \int_D \mathcal{L}(y, \lambda(x)) P_c(dx, dy),$$

$x \in X, y \in Y$. Заметим, что значение риска зависит от стратегии природы – распределения, которое неизвестно.

Если использованная нотация в виде интегралов по мере не привычна, можно использовать форму записи

$$E\mathcal{L}(y, \lambda(x)) = \int_D \mathcal{L}(y, \lambda(x)) dF_c(x, y).$$

Обе формы записи являются универсальными, т.е. позволяют не различать случаи дискретных, (абсолютно) непрерывных и даже сингулярных случайных величин.

Если вторая запись — также непривычна (непонятна) читатель может использовать вместо неё любую форму определения математического ожидания, которая ему знакома.

¹Ранее мы решающую функцию обозначали буквой f , но в данной главе используется обозначение λ .

Цель данной главы — научиться оценивать качество решающих функций, что означает построение оценок риска.

В математической статистике все оценки — это (измеримые) функции выборки (такие функции называются статистиками).

В задачах машинного обучения по крайней мере одна выборка у нас всегда есть — это обучающая выборка.

Пусть

$$S = ((x^i, y^i) \in D \mid i = 1, \dots, N)$$

— случайная независимая выборка из распределения P_c , $S \in D^N$.

Тогда любую (измеримую) функцию $\bar{R}(S, \lambda)$ можно считать оценкой риска. И основной задачей главы будет понять, что есть «хорошая» оценка риска и как её построить.

В отличие от классической задачи оценивания параметров оценивание риска подразумевает оценивание случайной величины. Действительно, риск зависит от решающей функции, которая строится по случайной выборке. Значит, риск — случайная величина. Формализуем это.

Методом построения решающих функций назовём отображение

$$Q: D^N \rightarrow \Lambda,$$

где Λ — заданный класс решающих функций.

Тогда риск будет величиной $R(c, \lambda_{Q,S})$, где $\lambda_{Q,S}$ — функция, построенная по выборке S методом Q .

Поскольку многие читатели, вероятно, знакомы с учебником [41], сделаем некоторые комментарии.

Приведём цитату из упомянутой работы. «Estimation of Err_T will be our goal, although we will see that Err is more amenable to statistical analysis, and most methods effectively estimate the expected error. It does not seem possible to estimate conditional error effectively, given only the information in the same training set.»

Переведём с использованием принятых нами обозначений. «Наша цель — оценить риск $R(c, \lambda)$, хотя в статистическом анализе чаще упоминается средний риск $ER(c, \lambda)$, и большинство методов эффективно оценивают именно средний риск. Кажется, что

только на основе обучающей выборки невозможно эффективно оценивать риск».

В данной главе мы вообще не будем касаться среднего риска, поскольку он нас пока не интересует (средний риск будет обсуждаться только в следующей главе). Здесь же мы будем делать именно то, что по утверждению из цитаты невозможно делать эффективно. На самом деле, у нас просто нет выбора, поскольку в практических задачах требуется именно оценка риска, а не среднего риска. Что касается эффективности, то этот вопрос будет обсуждаться.

На самом деле, мы вправе любую статистику формально объявить оценкой чего угодно. Вопрос только в том, насколько «хорошей» будет такая оценка. Вспомним, какие критерии качества оценок существуют.

Как известно, желательные свойства точечных оценок это:

- ✧ несмещённость;
- ✧ состоятельность;
- ✧ эффективность.

Перенёсем эти понятия на случай оценивания риска, с учётом того, что риск — случайная величина.

Оценка $\bar{R}_Q(S) = \bar{R}(S, \lambda_{Q,S})$ называется несмещённой оценкой риска $R_Q(c, S) = R(c, \lambda_{Q,S})$, если

$$E_S \bar{R}_Q(S) = E_S R_Q(c, S).$$

Математическое ожидание берётся по всем выборкам.

Оценка $\bar{R}_Q(S)$ является состоятельной, если

$$\bar{R}_Q(S) \xrightarrow[N \rightarrow \infty]{P} R_Q(c, S).$$

Эффективной оценкой в статистике называется несмещённая оценка с минимальной дисперсией.

В нашем случае (когда оценивается случайная величина) такое определение не подойдёт.

Будем исходить из того, что эффективная оценка должна обеспечивать минимальную погрешность, в роли которой будет выступать средний квадрат отклонения. Справедливо следующее разложение

$$E(\bar{R} - R)^2 = D\bar{R} + DR + (E\bar{R} - ER)^2 - 2\text{cov}(\bar{R}, R).$$

Видим, что помимо дисперсии оценки и её смещения на погрешность влияет и её корреляция с оцениваемой величиной.

Эта корреляция часто (в экспериментах) оказывается практически нулевой, однако в ряде случаев (при использовании log loss) наблюдались даже отрицательные значения. Может показаться странным, что измерение не коррелирует с измеряемой величиной (а тем более, имеет отрицательную корреляцию).

Для пояснения рассмотрим наглядный пример. Представим, что мы проводим измерения длины рельса, причём измерения проводим многократно и в разные дни. Из-за температурного расширения и непостоянства температуры сама измеряемая величина (длина рельса) становится случайной.

Если измерение проводим рулеткой, то погрешность измерения имеет систематическую составляющую (например из-за неточности шкалы рулетки) и случайную компоненту, обусловленную неточностью прикладывания, погрешностью при считывании меток, а также температурным удлинением самой рулетки. Последнее слагаемое даст положительную корреляцию с измеряемой величиной. Однако эта корреляция будет незначительной. Также можно представить, что рулетка сделана из материала с отрицательным коэффициентом температурного расширения, и тогда корреляция будет отрицательной.

Пример весьма условный, но позволяет наглядно понять смысл всех компонент разложения ошибки.

Рассмотрим в качестве точечных оценок риска известные выборочные функционалы качества (их можно называть функционалами, поскольку они являются функцией от решающих функций).

6.2 Контрольная выборка

Только что мы акцентировали внимание на том, что риск — случайная величина, но в данном параграфе временно «забудем» про это. Здесь мы будем рассматривать оценку риска на основе контрольной выборки. В этом случае мы полностью оказываемся в рамках классической математической статистики, и риск будет неизвестной величиной, но не случайной.

6.2.1 Точечная оценка

Пусть мы имеем некоторую решающую функцию $\lambda(x)$. При этом не важно, как эта функция получена: использовались некоторый метод и обучающая выборка или же функция выбрана вообще «наугад». Считаем решающую функцию фиксированной. Тогда риск — фиксированная неизвестная величина.

Пусть $S^* = ((x^i, y^i) \in D \mid i = 1, \dots, N^*)$ — «новая» случайная независимая выборка из распределения P_C , $S^* \in D^{N^*}$.

Оценку риска определим как средние потери на контрольной выборке:

$$R^*(S^*, \lambda) = \frac{1}{N^*} \sum_{i=1}^{N^*} \mathcal{L}(y^i, \lambda(x^i)).$$

Здесь мы обозначили объём контрольной выборки за N^* , чтобы подчеркнуть, что он не обязан совпадать с N — объёмом обучающей выборки. Однако, чтобы не загромождать обозначения, в дальнейшем размер контрольной выборки также будем обозначать как N .

В случае индикаторной функции потерь риск — это вероятность ошибочной классификации, а оценка на контрольной выборке — это просто доля ошибок на ней. Таким образом, имеем классическую задачу оценивания вероятности события в схеме Бернулли на основе частоты этого события в эксперименте. «Событием» здесь является факт ошибочного прогноза.

Данная задача очень хорошо изучена. Известно, что частота события — это несмещённая, состоятельная и эффективная оценка вероятности. Например, факт состоятельности известен как «закон больших чисел».

6.2.2 Доверительный интервал

Рассмотрим вопрос, как практически использовать оценку по контрольной выборке. Например, на 10 контрольных объектах решающая функция не допустила ни одной ошибки. Что можно сказать о вероятности ошибки на новых объектах?

Наиболее естественным кажется в качестве оценки вероятности ошибки взять её частоту. Частота ошибок в данном примере равна 0. Но корректно ли будет утверждать, что вероятность ошибки для данной решающей функции равна 0? Очевидно, что нет. Если сделать такое заявление, то пользователи этой модели будут полностью ей доверять (т. е. считать, что она не ошибается). При этом понятно, что проверка модели на 10 объектах не может быть основанием для такой уверенности.

Ситуация в примере эквивалентна следующей задаче. Известно, что в урне белые и чёрные шары. Извлекли 10 шаров, все оказались белыми. Какой прогноз о цвете следующего шара?

Математически корректный ответ можно дать, построив доверительный интервал.

Пусть p — вероятность чёрного шара, N — число извлечённых шаров, M — количество чёрных среди них. Тогда

$$P_{p,N}(M) = C_N^M p^M \cdot (1-p)^{N-M}, \quad P_{p,N}(0) = (1-p)^N.$$

Если, например, $p = 0,5$, то $P_{p,N}(0) = (0,5)^{10} \approx 0,001$. Получается, что при таком значении p вероятность получить 10 белых шаров из 10 очень низкая, но это случилось. Значит, предположение, что $p = 0,5$, является неправдоподобным, и на самом деле p гораздо меньше.

При уменьшении p вероятность полученного в эксперименте результата будет увеличиваться и с какого-то момента станет достаточно большой, чтобы считать ситуацию правдоподобной.

Иными словами, идея построения доверительного интервала заключается в том, что мы отвергаем все варианты (гипотезы относительно значения оцениваемого параметра), при которых вероятность полученного в эксперименте результата «слишком низкая». В каком-то смысле мы считаем, если у события «слишком низкая» вероятность, то оно «не должно было» случиться.

Порог, начиная с которого мы считаем вероятность «слишком низкой», называется уровнем значимости α и выбирается эвристически, например 0,05. Также используется понятие доверительной вероятности, которая есть $\eta = 1 - \alpha$.

Положив $P_{p,N}(0) = \alpha$, имеем $p = 1 - \alpha^{\frac{1}{N}} = 1 - e^{\frac{\ln \alpha}{N}}$.

При $\alpha = 0,1$ и $N = 10$ получим $p \approx 0,2$.

Для наглядности мы рассмотрели случай $M = 0$. В общем случае доверительные интервалы в схеме Бернулли выглядят следующим образом.

Односторонний интервал $[0, \hat{p}]$ находится из уравнения

$$\sum_{i=0}^M C_N^i \hat{p}^i \cdot (1 - \hat{p})^{N-i} = \alpha.$$

Двусторонний интервал $[p_1, p_2]$ находится из уравнений

$$\sum_{i=0}^M C_N^i p_2^i \cdot (1 - p_2)^{N-i} = \sum_{i=M}^N C_N^i p_1^i \cdot (1 - p_1)^{N-i} = \frac{\alpha}{2}.$$

Здесь мы к вероятности наблюдаемого события прибавляем вероятности всех событий, которые ещё менее правдоподобны при условии гипотезы. Это нужно в том числе ввиду того, что сама по себе вероятность даже наивероятнейшего числа успехов может быть очень низка.

На рис. 6.1 изображены односторонние доверительные интервалы для разных α (слева) и двусторонний интервал при $\alpha = 0,1$ (справа). Кривые «Equal» и «Bayes» на левом графике отражают усреднённые оценки, которые будут вычислены ниже.

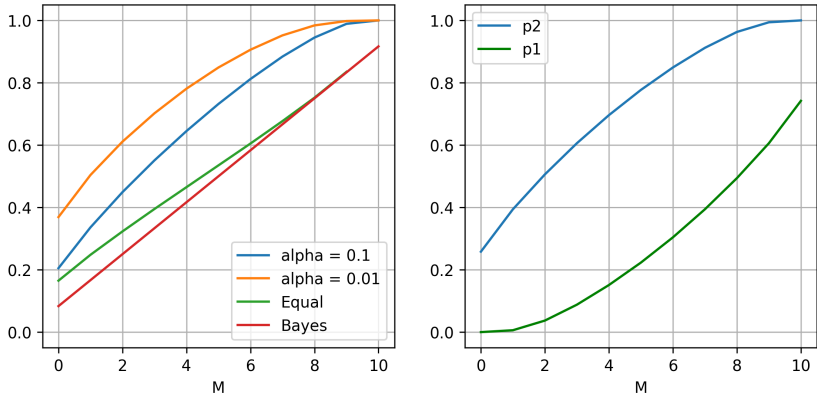


Рис. 6.1. Доверительные интервалы для риска при $N = 10$

Имеет место полезное тождество

$$\begin{aligned}
 & \sum_{i=0}^M C_N^i p^i \cdot (1-p)^{N-i} = \\
 & = 1 - N \cdot C_{N-1}^M \int_0^p x^M \cdot (1-x)^{N-M-1} dx = \\
 & = 1 - \int_0^p \beta_{M+1, N-M}(x) dx, \tag{6.1}
 \end{aligned}$$

где

$$\beta_{a,b}(x) = \frac{1}{B(a,b)} \cdot x^{a-1} \cdot (1-x)^{b-1}, \quad x \in [0, 1],$$

есть плотность вероятности для бета-распределения, а

$$B(a,b) = \int_0^1 x^{a-1} \cdot (1-x)^{b-1} dx$$

– бета-функция.

При целых значениях аргументов бета-функция имеет выражение

$$B(N-M+1, M+1) = \frac{1}{(N+1) \cdot C_N^M}.$$

Справедливость формулы (6.1) можно проверить, сравнив результаты вызовов `1 - scipy.stats.beta.cdf(p,m+1,n-m)` и `scipy.stats.binom.cdf(m,n,p)`. При всех допустимых значениях аргументов результаты совпадают (с точностью до последних знаков). Метод `cdf` вычисляет значение функции распределения.

Соотношение позволяет вычислять точные доверительные интервалы для параметра биномиального распределения через стандартные библиотечные функции.

Пример кода.

```
import scipy.stats as st

def conf_interval(cprob,M,N):
    q = 1 - cprob
    p1 = st.beta.ppf(q/2,M,N-M+1)
    p2 = st.beta.ppf(1-q/2,M+1,N-M)
    return p1, p2

def byes_interval(prob,M,N):
    q = 1 - prob
    p1 = st.beta.ppf(q/2,M+1,N-M+1)
    p2 = st.beta.ppf(1-q/2,M+1,N-M+1)
    return p1, p2
```

Функция `conf_interval` вычисляет двусторонний доверительный интервал для заданной доверительной вероятности `cprob` при числе испытаний N и числе успехов M .

Для вычисления одностороннего интервала (оценки сверху) достаточно вызвать `scipy.stats.beta.ppf(conf,M+1,N-M)`. Метод `ppf` вычисляет заданный квантиль распределения, т. е. реализует функцию, обратную `cdf`.

Функция `byes_interval` вычисляет байесовский интервал, смысл которого объясняется ниже.

6.2.3 Усреднённые оценки

С математической точки зрения доверительный интервал для риска — это, пожалуй, наиболее полный ответ, который можно дать (оценивая риск). Однако для конечного пользователя мо-

дели такая конструкция (доверительный интервал для вероятности ошибки при заданной доверительной вероятности) является слишком сложной. Пользователю желательно получить одно число, которое он мог бы интерпретировать как ожидаемую вероятность ошибки для модели.

Это можно сделать, например, в рамках байесовского подхода, который является одним из классических подходов математической статистики при оценивании параметров.

Будем считать параметр случайной величиной и предположим, что она имеет равномерное априорное распределение, т. е. плотность вероятности $\varphi(p) \equiv 1, 0 \leq p \leq 1$.

Тогда по формуле Байеса можем вычислить апостериорное распределение (при условии появления M успехов в серии N испытаний)

$$\varphi(p | M) = P_{p,N}(M) \frac{\varphi(p)}{P(M)}.$$

После преобразований, получаем

$$\varphi(p | M) = (N + 1) C_N^M p^M \cdot (1 - p)^{N-M} = \beta_{M+1, N-M+1}(p).$$

По условному распределению можем вычислить симметричный интервал значений, вероятность попадания в который равна заданной вероятности. Получим байесовский аналог доверительного интервала, назовём его байесовским интервалом.

Также можно вычислить условное математическое ожидание

$$E_M p = \int_0^1 p \varphi(p | M) dp = \frac{M + 1}{N + 2}.$$

Получаем, что если модель на 10 контрольных объектах ни разу не ошиблась, то вероятность ошибки для неё следует оценить как $\frac{1}{12}$.

На рис. 6.1 эта оценка имеет метку «Bayes».

Байесовский подход имеет очевидный недостаток, связанный с необходимостью задавать априорное распределение, что в данном случае является никак не обоснованным произволом.

Попробуем действовать иначе. Заметим, что функция $\eta(\hat{p}) = 1 - \alpha(\hat{p})$ обладает всеми свойствами функций распределения (хоть ей и не является).

Однако чисто технически мы можем взять усреднение

$$\hat{E}_{Mp} = \int_0^1 \hat{p} d\eta(\hat{p}) = \frac{M+1}{N+1}.$$

Данная операция выглядит сомнительно, но как известно: «если нельзя, но очень хочется, то — можно». Формально мы не называли его математическим ожиданием, а вычислять вправе всё, что угодно. В результате мы получаем выражение, очень похожее на полученное из байесовского подхода. Выражение можно использовать при $M < \frac{N}{2}$.

Ещё один способ заменить доверительный интервал одним числом — это приравнять \hat{p} и α . Действительно, с уменьшением уровня значимости ширина доверительного интервала растёт, и есть точка, где они совпадают. Это значение также можно использовать как оценку риска. На рис. 6.1 эта оценка имеет метку «Equal».

6.2.4 Случай произвольной функции потерь

В целях упрощения и наглядности, мы большую часть рассуждений вели для случая задачи классификации с индикаторной функцией потерь.

В общем случае выкладки, как правило, аналогичны.

Если критерий качества представляется математическим ожиданием функции потерь, то мы имеем классическую задачу статистики, а именно оценивание математического ожидания случайной величины при неизвестном (чаще всего) распределении.

Для упрощения выкладок введём обозначение $\zeta_i = \mathcal{L}(y^i, \lambda(x^i))$ для величины потерь на i -м объекте контрольной выборки. Поскольку элементы выборки независимы, случайные величины ζ_i также независимы. При этом они имеют одинаковое распределение.

Требуется оценить математическое ожидание $\mu = \mathbb{E}\zeta_i$, т. е. риск.

Используемая нами оценка R^* в текущих обозначениях принимает вид

$$\bar{\mu} = \frac{1}{N} \sum_{i=1}^N \zeta_i.$$

Чтобы оценить точность этой оценки, нужно оценить дисперсию $\sigma^2 = \mathbb{D}\zeta_i$.

Рассмотрим оценку

$$\bar{\sigma}^2 = \frac{1}{N} \sum_{i=1}^N (\zeta_i - \bar{\mu})^2 \equiv \frac{1}{N} \sum_{i=1}^N \zeta_i^2 - \bar{\mu}^2.$$

Вычислим её математическое ожидание

$$\mathbb{E}\bar{\sigma}^2 = \mathbb{E} \frac{1}{N} \sum_{i=1}^N \zeta_i^2 - \mathbb{E}\bar{\mu}^2 = \mathbb{E}\zeta_i^2 - \mathbb{E}\bar{\mu}^2.$$

$$\mathbb{E}\bar{\mu}^2 = \frac{1}{N^2} \mathbb{E} \left(\sum_{i=1}^N \zeta_i \right)^2 = \frac{1}{N^2} \left(\sum_{i=1}^N \mathbb{E}\zeta_i^2 + \sum_{j \neq i} \mathbb{E}\zeta_i \zeta_j \right).$$

Здесь мы раскрыли квадрат суммы и сгруппировали слагаемые. В силу независимости имеем $\mathbb{E}\zeta_i \zeta_j = \mathbb{E}\zeta_i \mathbb{E}\zeta_j = \mu^2$. Откуда

$$\mathbb{E}\bar{\mu}^2 = \frac{1}{N} \mathbb{E}\zeta_i^2 + \frac{N-1}{N} \mu^2.$$

Наконец

$$\mathbb{E}\bar{\sigma}^2 = \frac{N-1}{N} (\mathbb{E}\zeta_i^2 - \mu^2) = \frac{N-1}{N} \sigma^2.$$

Мы вывели известный факт, что несмещённая оценка дисперсии есть

$$\tilde{\sigma}^2 = \frac{N}{N-1} \bar{\sigma}^2.$$

Можно было вывести это же на основе того, что дисперсия суммы независимых величин равна сумме дисперсий. Из этого следует

$$D\bar{\mu} = \frac{1}{N} \sigma^2.$$

С другой стороны

$$D\bar{\mu} \equiv E\bar{\mu}^2 - (E\bar{\mu})^2 = E\bar{\mu}^2 - \mu^2.$$

Откуда

$$E\bar{\mu}^2 = \mu^2 + \frac{1}{N} \sigma^2.$$

Снова получаем

$$E\bar{\sigma}^2 = E\zeta_i^2 - \mu^2 - \frac{1}{N} \sigma^2 = \sigma^2 - \frac{1}{N} \sigma^2.$$

Подобный вывод приводится во всех курсах по математической статистике и достаточно было сослаться на них, но мы подробно воспроизвели выкладки ввиду их важности.

Итак, $\tilde{\sigma}^2$ является несмещённой оценкой для σ^2 , а $D\bar{\mu} = \frac{1}{N} \sigma^2$, поэтому $\frac{1}{N} \tilde{\sigma}^2$ будет несмещённой оценкой для $D\bar{\mu}$.

На основе полученных выражений можно построить доверительный интервал. Введём величину

$$t = (\bar{\mu} - \mu) \frac{\sqrt{N}}{\tilde{\sigma}}.$$

Если ζ_i имеют нормальное распределение, то случайная величина t имеет распределение Стьюдента с $N - 1$ степенями свободы, которое с ростом N сходится к нормальному распределению. При реально используемых размерах выборки распределение можно заменить нормальным.

В реальных задачах ζ_i крайне редко имеют нормальное распределение. Однако в любом случае в силу центральной предельной теоремы распределение величины $\bar{\mu}$ сходится к нормальному.

На практике это означает, что если мы построили несмещённую оценку для дисперсии средних потерь на контрольной выборке, то можем использовать её для оценки доверительного интервала, в том числе по правилу «три сигма».

Заметим, что скорость сходимости к нормальному закону существенно замедляется при наличии «выбросов», или «тяжёлых хвостов» у распределения величин ζ_i . В таких случаях доверительный интервал, построенный на нормальном приближении, может иметь значительную погрешность.

Для критериев, не представляемых через функцию потерь (например ROC-AUC), доверительный интервал построить сложнее, однако задача так или иначе решается (возможно, средствами статистического моделирования).

Предположим теперь, что мы разбили контрольную выборку на K частей, которые будем называть *фолдами*. Пусть J_k – множество индексов объектов, принадлежащих k -му фолду, а N_k – размер фолда.

Обозначим

$$\bar{\mu}_k = \frac{1}{N_k} \sum_{i \in J_k} \zeta_i.$$

Для удобства будем считать, что N делится на K . Положим все N_k одинаковыми, т. е. $N_k \equiv \frac{N}{K}$.

Тогда все μ_k будут независимыми одинаково распределёнными случайными величинами со средним μ и дисперсией

$$\sigma_K^2 = \frac{\sigma^2}{N_k} = \sigma^2 \frac{K}{N}.$$

Оценка

$$\tilde{\sigma}_K^2 = \frac{1}{K-1} \sum_{k=1}^K (\bar{\mu}_k - \bar{\mu})^2$$

будет несмещённой оценкой для σ_K^2 , а значит, $\hat{\sigma}_K^2 = \tilde{\sigma}_K^2 \cdot \frac{N}{K}$ будет несмещённой оценкой для σ^2 .

Означает ли это, что оценку $\hat{\sigma}_K^2$ можно использовать наряду с $\tilde{\sigma}^2$ в качестве оценки параметра σ^2 ?

Формально использовать можно, но это будет очень плохая оценка.

Чтобы убедиться в этом, вычислим дисперсию данной оценки. Поскольку в общем случае это сделать не получится, предположим, что ζ_i имеют нормальное распределение.

Тогда $(K - 1) \cdot \frac{N\hat{\sigma}_K^2}{K\sigma^2}$ имеет распределение χ^2 с $K - 1$ степенями свободы.

Дисперсия такого распределения известна и равна $2(K - 1)$. Откуда

$$D\hat{\sigma}_K^2 = 2(K - 1) \left(\frac{\sigma^2}{K - 1} \right)^2 = \frac{2\sigma^4}{K - 1}.$$

Минимальная дисперсия оценки достигается при $K = N$, т. е. при негруппированной выборке. Это вполне естественно: если мы хотим оценить дисперсию, т. е. «колебания» значений, то не следует их сглаживать усреднением. Использование такой оценки эквивалентно по точности уменьшению объема выборки до K .

Оценка

$$\check{\sigma}_K^2 = \frac{1}{N - K} \sum_{j=1}^N (\zeta_j - \bar{\mu}_{k(j)})^2$$

также является несмещённой оценкой для σ^2 . Здесь $k(j)$ – номер фолда, в который попадает j -й объект.

Если ζ_i имеют нормальное распределение, то $(N - K) \cdot \frac{\check{\sigma}_K^2}{\sigma^2}$ имеет распределение χ^2 с $N - K$ степенями свободы. Из этого можно вывести, что

$$D\check{\sigma}_K^2 = \frac{2\sigma^4}{N - K}.$$

Таким образом, точность оценки $\check{\sigma}_K^2$ снижается с ростом числа фолдов так, будто выборка уменьшается на K .

То, что группировка выборки уменьшает точность оценивания дисперсии, было вполне очевидно, и приведённые выкладки кажутся излишними. Действительно, в случае контрольной выборки данные манипуляции не имеют практического смысла, однако разбиение на фолды будет в дальнейшем использоваться при

кросс-валидации, и тогда выведенные сейчас формулы нам пригодятся.

6.2.5 Свойства оценки по контрольной выборке

Средний риск на контрольной выборке как оценка риска обладает следующими свойствами:

- ✧ простота вычисления;
- ✧ несмещённость (не совсем в том смысле, в котором хотелось бы);
- ✧ состоятельность;
- ✧ известен точный доверительный интервал;
- ✧ эффективность (не в том смысле, в котором хотелось бы);
- ✧ требует дополнительной выборки.

Некоторые свойства, очевидно, требуют пояснений. Например, почему несмещённость оценки не такая, которую бы хотелось. Вспомним, что получив ноль ошибок на контрольной выборке, мы оценили риск ненулевым значением. Получается, что вместо изначально несмещённой оценки мы использовали скорректированную оценку, которая формально является смещённой. Тем не менее, несмещённость — важное свойство оценки.

Что касается эффективности, то формально эффективность имеет место. А неформально эффективность интерпретируется как то, что оценка в некотором смысле самая лучшая.

При условии использования контрольной выборки и неиспользования обучающей, рассмотренную оценку действительно можно считать самой лучшей. Но на практике идеальным вариантом было бы обойтись вообще без контрольной выборки (использовать обучающую). Получается, что эффективность в данном случае не означает, что оценка лучшая из возможных.

В качестве общего вывода можно сказать, что контрольная выборка — единственный на настоящий момент способ получить оценку риска с математически строгим доверительным интервалом (т. е. с известной точностью).

Для двух случаев, а именно для биномиального и для нормального распределения риска, мы привели выражения для точных доверительных интервалов. В остальных случаях можно пользоваться нормальным приближением либо статистическим моделированием.

6.3 Отложенная выборка

6.3.1 Определение

Отложенная выборка (называется hold-out) представляет собой часть исходной выборки, которая не используется для обучения, а отложена для контроля (теста).

Кажется, что это не отличается от контрольной выборки. И технически это действительно такая же тестовая выборка, но отличие в том, что мы хотим оценить.

Ранее мы по обучающей выборке строили модель, а по контрольной выборке оценивали её качество. Но если мы отложили часть выборки для оценки качества решения, то для получения финальной решающей функции (которую отправим в качестве результата) имеет смысл провести обучение на всей выборке. Решение получится только точнее.

Получается, что оценку по отложенной выборке мы применяем не к той решающей функции, для которой она получена.

Заметим, что такое терминологическое различие контрольной и отложенной выборок не является общепринятым. Однако оно вполне логично и согласуется с общепринятым представлением, что отложенная выборка — частный случай кроссвалидации (будет рассмотрена далее).

Оценка hold-out определяется как

$$\check{R}(S, S^*, Q) = \frac{1}{N^*} \sum_{(x^i, y^i) \in S^*} \mathcal{L}(y^i, \lambda_{Q, S}(x^i)),$$

где S — обучающая выборка, S^* — отложенная (тестовая) выборка.

6.3.2 Несмещённость оценки hold-out

Оценка hold-out является несмещённой, т. е. имеет место

$$E_S E_{S^*} \check{R}(S, S^*, Q) = E_S R(S, Q). \quad (6.2)$$

Здесь мы при операторе математического ожидания указываем «переменную», по которой ведётся усреднение.

Величина

$$R(S, Q) = R(\lambda_{Q,S}) = E_{x,y} \mathcal{L}(y, \lambda_{Q,S}(x))$$

есть риск для решающей функции, построенной методом Q по выборке S .

Тождество (6.2) является элементарным следствием несмещённости оценки по контрольной выборке, т. е. тождества

$$E_{S^*} \check{R}(S, S^*, Q) = R(S, Q).$$

Для доказательства используются рассуждения, аналогичные рассуждениям в следующей задаче.

Задача. В урне находятся 7 чёрных и 3 белых шара. Наугад извлекли шар. Затем извлекли второй шар, который оказался белым. Какова вероятность, что и первый шар — белый?

Решение. Поскольку при извлечении первого шара мы не посмотрели на его цвет, ответ будет таким же, как если бы этот шар извлекался вторым, т. е. $\frac{2}{9}$.

Применительно к оценке hold-out можно аналогично сказать, что пока мы не использовали объекты из отложенной выборки, ожидаемая ошибка на них такая же, как на любых «новых» объектах.

6.3.3 Вероятностные характеристики модели

Для случайных величин вводят различные характеристики: математическое ожидание, дисперсия, медиана, мода и прочие.

Поскольку мы используем вероятностную постановку, переменные представляют собой систему случайных величин, и для этой системы мы сейчас введём некоторые характеристики.

Это будут характеристики совместного распределения, но они также будут зависеть от дополнительных параметров, а именно: метода построения решающих функций, а также параметров выборки.

Однако сначала выведем полезное для дальнейших выкладок тождество.

Пусть $f(u, v)$ – произвольная (в рамках некоторых ограничений, обеспечивающих существование рассматриваемых величин) функция от случайных величин u и v .

Имеет место тождество

$$Df(u, v) = D_v E_{u|v} f(u, v) + E_v D_{u|v} f(u, v), \quad (6.3)$$

где $E_{u|v} f(u, v)$ – условное математическое ожидание функции $f(u, v)$, когда усреднение ведётся по условному распределению на u при фиксированном значении v , а $D_{u|v} f(u, v)$ – условная дисперсия, определяемая аналогично.

Вывод основан на тождестве

$$Ef(u, v) = E_v E_{u|v} f(u, v),$$

которое является следствием формулы полной вероятности.

Имеем

$$\begin{aligned} & D_v E_{u|v} f(u, v) + E_v D_{u|v} f(u, v) = \\ &= E_v (E_{u|v} f(u, v))^2 - (E_v E_{u|v} f(u, v))^2 + \\ &+ E_v E_{u|v} (f(u, v))^2 - E_v (E_{u|v} f(u, v))^2 = \\ &= E_v E_{u|v} (f(u, v))^2 - (E_v E_{u|v} f(u, v))^2 = \\ &= E(f(u, v))^2 - (Ef(u, v))^2 = Df(u, v). \end{aligned}$$

Тождество (6.3) доказано.

Пусть S – выборка объёма N .

Введём характеристики:

$$\begin{aligned} \bar{\zeta}(N, Q) &= DR(S, Q), \\ \dot{\zeta}(N, Q) &= E_S D_{x,y|S} \mathcal{L}(y, \lambda_{Q,S}(x)), \\ \zeta(N, Q) &= D_{S,x,y} \mathcal{L}(y, \lambda_{Q,S}(x)). \end{aligned}$$

Из (6.3) следует

$$\varsigma(N, Q) = \zeta(N, Q) + \bar{\varsigma}(N, Q).$$

Для анализа процедуры скользящего экзамена нам потребуются характеристики, отражающие связь между потерями моделей, построенных на частично пересекающихся выборках.

Пусть $S_{\&}$ – выборка объёма $N_{\&} < N$, а S_a и S_b – независимые выборки одинакового объёма $N - N_{\&}$.

Сформируем выборки $S_A = S_a \cup S_{\&}$, $S_B = S_b \cup S_{\&}$.

Формально выборка является не множеством объектов, а последовательностью, но мы будем использовать для выборок привычные операции для множеств (объединение, пересечение, разность).

Определим характеристику

$$\begin{aligned} \varpi(N, N_{\&}, Q) &= \text{cov}_{S_a, S_{\&}, S_b} (R(S_A, Q), R(S_B, Q)) = \\ &= \text{cov}_{S_a, S_{\&}, S_b, x, y} (R(S_A, Q), \mathcal{L}(y, \lambda_{Q, S_B}(x))) = \\ &= \text{cov}_{S_a, S_{\&}, S_b, x', y', x'', y''} (\mathcal{L}(y', \lambda_{Q, S_A}(x')), \mathcal{L}(y'', \lambda_{Q, S_B}(x''))). \end{aligned}$$

Мы ввели обозначение для ковариации между риском для решений, построенных по частично пересекающимся выборкам. Для неё оказалось три эквивалентных выражения, поскольку риск — это средним потери на новом тестовом объекте. В последнем выражении мы оцениваем риск каждого решения на своём тестовом объекте.

Однако если оценивать риск этих решений на одном и том же тестовом объекте, то характеристика получается другая:

$$\overset{\circ}{\varsigma}(N, N_{\&}, Q) = \text{cov}_{S_a, S_{\&}, S_b, x, y} (\mathcal{L}(y, \lambda_{Q, S_A}(x)), \mathcal{L}(y, \lambda_{Q, S_B}(x))).$$

Имеет место неравенство

$$\overset{\circ}{\varsigma}(N, N_{\&}, Q) \leq \varsigma(N, Q) \equiv \overset{\circ}{\varsigma}(N, N, Q).$$

Величина $\overset{\circ}{\varsigma}(N, N_{\&}, Q)$ на практике очень близка к $\varsigma(N, Q)$. Наибольшее отличие между ними будет, если S_A и S_B не пересекаются (т. е. $N_{\&} = 0$).

Осталось рассмотреть случай, когда риск одной модели оценивается на объекте, который входит в обучающую выборку другой модели.

Имеют место два варианта. В первом мы оцениваем риск одной из моделей на новом объекте

$$\begin{aligned}\dot{\omega}(N, N_{\&}, Q) &= \text{cov}_{S_a, S_{\&}, S_b} (R(S_A, Q), \mathcal{L}(y', \lambda_{Q, S_B}(x'))) = \\ &= \text{cov}_{S_a, S_{\&}, S_b, x, y} (\mathcal{L}(y, \lambda_{Q, S_A}(x)), \mathcal{L}(y', \lambda_{Q, S_B}(x'))),\end{aligned}$$

где $(x', y') \in S_a$.

В последнем варианте риск каждой модели оценивается на объекте, который входит в обучающую выборку другой модели

$$\ddot{\omega}(N, N_{\&}, Q) = \text{cov}_{S_a, S_{\&}, S_b} (\mathcal{L}(y', \lambda_{Q, S_A}(x')), \mathcal{L}(y'', \lambda_{Q, S_B}(x''))),$$

где $(x', y') \in S_b, (x'', y'') \in S_a$.

6.3.4 Точность оценки hold-out

Применим (6.3) для вычисления дисперсии оценки hold-out. Имеем

$$\begin{aligned}D\check{R}(S, S^*, Q) &= D_S E_{S^* | S} \check{R}(S, S^*, Q) + E_S D_{S^* | S} \check{R}(S, S^*, Q) = \\ &= D_S R(S, Q) + \frac{1}{N^*} E_S D_{x, y} \mathcal{L}(y, \lambda_{Q, S}(x)).\end{aligned}\quad (6.4)$$

Второе слагаемое в полученном выражении есть дисперсия оценки по контрольной выборке, усреднённая по решающим функциям (зависящим от обучающей выборки). Первое слагаемое отражает разброс по решающим функциям. Это слагаемое как раз и отражает увеличение погрешности оценки, если оценка переносится с одной решающей функции на другую.

Для произвольных случайных величин u и v имеет место тождество

$$E(u - v)^2 = Du + (Eu - Ev)^2 + Dv - 2 \text{cov}(u, v). \quad (6.5)$$

В справедливости тождества легко убедиться, раскрыв дисперсии и ковариацию через их определения.

Предположим, что мы построили решающую функцию по новой выборке S^+ , которая имеет тот же объём N , что и S , и теперь хотим выяснить, насколько близок риск для этой функции к оценке hold-out, полученной для выборки S .

Воспользуемся формулой (6.5), где в роли u будет выступать $\check{R}(S, S^*, Q)$, а вместо v подставим $R(S^+, Q)$. Поскольку выборки S и S^+ не пересекаются, корреляция нулевая. Второе слагаемое в (6.5) будет равно нулю ввиду несмещённости оценки hold-out. Получаем

$$\begin{aligned} \mathbb{E} (\check{R}(S, S^*, Q) - R(S^+, Q))^2 &= D_{S^+} R(S^+, Q) + D_{S, S^*} \check{R}(S, S^*, Q) = \\ &= 2D_S R(S, Q) + \frac{1}{N^*} \mathbb{E}_S D_{x, y | S} \mathcal{L}(y, \lambda_{Q, S}(x)) = \\ &= 2\bar{\zeta}(N, Q) + \frac{1}{N^*} \zeta(N, Q). \end{aligned} \quad (6.6)$$

Для более компактной записи мы использовали ранее введённые обозначения

$$\bar{\zeta}(N, Q) = DR(S, Q), \quad \zeta(N, Q) = \mathbb{E}_S D_{x, y | S} \mathcal{L}(y, \lambda_{Q, S}(x)).$$

Полученная формула пока представляется мало полезной, поскольку сложно представить, в какой бы ситуации можно было на практике использовать такую схему (с двумя независимыми обучающими выборками).

Вместе с тем структура формулы (6.6) весьма примечательна. Второе слагаемое в ней — попросту средняя дисперсия оценки по контрольной выборке. А первое слагаемое — удвоенный разброс точности решений по разным обучающим выборкам.

В дальнейшем мы выведем аналогичную формулу для оценки скользящего экзамена, где выкладки будут сложнее, но часть слагаемых будет такой же, как в (6.6).

6.4 Скользящий экзамен

Использование контрольной выборки требует дополнительных данных, а эмпирический риск, как правило, сильно смещён. Однако есть способ избежать этих недостатков.

Идея *скользящего экзамена* заключается в том, чтобы использовать всю выборку и для обучения, и для контроля, но не одновременно, а так, чтобы объекты, на которых проводится тестирование, не участвовали в обучении.

6.4.1 Определение

Функционал скользящего экзамена определяется как

$$\check{R}(S, Q) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y^i, \lambda_{Q, S'_i}(x^i)),$$

где $S'_i = S \setminus (x^i, y^i)$ – выборка, получаемая из S удалением i -го наблюдения.

Суть процедуры в том, что каждый объект по очереди удаляется из обучающей выборки, по оставшимся объектам строится решающая функция, по которой делается прогноз для удалённого (отложенного) объекта. В результате мы получаем прогнозы для всех объектов. Далее для прогнозов вычисляем потери и усредняем.

В англоязычной терминологии такая процедура называется *leave-one-out*.

Данная процедура достаточно трудоёмкая вычислительно, поскольку требует провести обучение N раз.

Для сокращения затрат можно удалять объекты не по одному, а группами. Тогда получим метод кроссвалидации (*K-fold cross-validation*).

Исходная выборка разбивается на K равных частей, которые

называются фолдами. Оценка определяется как

$$\check{R}^K(S, Q) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y^i, \lambda_{Q, S'_{k(i)}}(x^i)),$$

где $S'_{k(i)}$ – выборка, получаемая из S удалением всей подвыборки (всего фолда), которой принадлежит i -е наблюдение. Здесь $k(i)$ – номер фолда, которому принадлежит i -й элемент выборки.

Для простоты выкладок полагаем, что N кратно K . В общем случае фолды будут не совсем равными, что практически несущественно.

Заметим, что в некоторых источниках такая оценка называется *out-of-fold*, а собственно кроссвалидацией называется эта же сумма, но со сгруппированными слагаемыми. Однако простая группировка слагаемых не кажется достаточным основанием для введения отдельных терминов, поэтому термин *out-of-fold* мы использовать не будем.

Оценка *leave-one-out* является частным случаем *K-fold cross-validation* при $K = N$.

Замечание по терминологии. Часто под скользящим экзаменом понимается именно *leave-one-out*. Мы же будем использовать термин «скользящий экзамен» двояко: в узком смысле как *leave-one-out*, и в широком смысле как любой метод, использующий всю выборку и для обучения, и для контроля, но так, чтобы объекты, на которых проводится тестирование, не участвовали в обучении.

Для дальнейших рассуждений нам нужно выразить кроссвалидацию через оценки *hold-out*. Обозначим

$$\check{R}_k(S, Q) = \frac{1}{N_k} \sum_{i \in J_k} \mathcal{L}(y^i, \lambda_{Q, S'_k}(x^i)),$$

где N_k – размер k -го фолда, а J_k – множество индексов, ему принадлежащих.

Тогда

$$\check{R}^K(S, Q) = \frac{1}{N} \sum_{k=1}^K N_k \cdot \check{R}_k(S, Q).$$

В процессе кроссвалидации каждый фолд по очереди выступает в роли отложенной выборки.

6.4.2 Несмещённость скользящего экзамена

Теорема. Для оценки кроссвалидации имеет место

$$\mathbb{E}\check{R}^K(S_N, Q) = \mathbb{E}R(S_{N-\frac{N}{K}}, Q). \quad (6.7)$$

Как частный случай

$$\mathbb{E}\check{R}(S_N, Q) = \mathbb{E}R(S_{N-1}, Q). \quad (6.8)$$

Данный факт означает, что математическое ожидание оценки кроссвалидации равно среднему риску для решений, построенных тем же методом по выборкам размера $N - \frac{N}{K}$.

Этот факт мы будем называть несмещённостью скользящего экзамена.

При этом, делая утверждение о несмещённости оценки скользящего экзамена, нужно обязательно делать оговорку, что несмещённость имеет место, если финальное решение строится по выборке размера $N - \frac{N}{K}$.

В реальной ситуации, когда финальное решение строится по выборке размера N , скользящий экзамен даёт вообще говоря смещённую оценку (несколько пессимистичную).

Доказательство теоремы элементарно, хотя и неочевидно.

Для простоты рассмотрим случай $K = N$, т. е. в каждом фолде один объект. Поскольку все фолды равноправны, возьмём первый фолд в качестве тестового.

Из несмещённости оценки hold-out следует, что

$$\mathbb{E}\check{R}_1(S, Q) = \mathbb{E}\mathcal{L}(y^1, \lambda_{Q, S_1'}(x^1)) = \mathbb{E}R(S_{N-1}, Q).$$

Для завершения доказательства теоремы осталось использовать тот факт, что математическое ожидание суммы есть сумма математических ожиданий, причём для любых, в том числе для зависимых, случайных величин.

6.4.3 Точность скользящего экзамена

Попытаемся оценить погрешность оценки скользящего экзамена как средний квадрат её отклонения от риска.

Необходимо, однако, выбрать, что использовать в качестве итоговой решающей функции.

Чаще всего после проведения кроссвалидации финальное решение строится путём обучения на всей выборке. Мы же для простоты в роли финального решения будем использовать решающую функцию, полученную для прогноза первого фолда.

Как и ранее, S'_k будет обозначать выборку без k -го фолда, а через S_k^* обозначим сам k -й фолд.

Воспользуемся формулой (6.5), где в роли u будет выступать $\check{R}^K(S, Q)$, а вместо v подставим $R(S'_1, Q)$ – риск решения, построенного при удалении первого фолда. Второе слагаемое в (6.5), как и ранее, будет равно нулю ввиду несмещённости оценки, но корреляция теперь не нулевая. Получаем

$$\begin{aligned} & \mathbb{E} \left(\check{R}^K(S, Q) - R(S'_1, Q) \right)^2 = \\ & = D_{S'_1} R(S'_1, Q) + D_S \check{R}^K(S, Q) - 2 \operatorname{cov} \left(R(S'_1, Q), \check{R}^K(S, Q) \right). \end{aligned} \quad (6.9)$$

Пусть N кратно K и $N_k \equiv \frac{N}{K}$.

Заметим, что

$$D_{S'_1} R(S'_1, Q) = DR(S_{N-\frac{N}{K}}, Q) = \bar{\varsigma} \left(N - \frac{N}{K}, Q \right).$$

Преобразуем

$$\begin{aligned} \operatorname{cov} \left(R(S'_1, Q), \check{R}^K(S, Q) \right) &= \frac{1}{K} \cdot \sum_{k=1}^K \operatorname{cov} \left(R(S'_1, Q), \check{R}_k(S, Q) \right) = \\ &= \frac{1}{K} \cdot \bar{\varsigma} \left(N - \frac{N}{K}, Q \right) + \frac{K-1}{K} \cdot \dot{\omega} \left(N - \frac{N}{K}, N - \frac{2N}{K}, Q \right), \end{aligned}$$

где $\dot{\omega} \left(N - \frac{N}{K}, N - \frac{2N}{K}, Q \right) = \operatorname{cov} \left(R(S'_{k'}, Q), \check{R}_{k''}(S, Q) \right)$, $k' \neq k''$.

Теперь (6.9) можем записать в виде

$$\begin{aligned} \mathbb{E} \left(\check{R}^K(S, Q) - R(S'_1, Q) \right)^2 &= \\ &= \frac{K-2}{K} \cdot \bar{\varsigma} \left(N - \frac{N}{K}, Q \right) + D_S \check{R}^K(S, Q) - \\ &\quad - 2 \cdot \frac{K-1}{K} \cdot \check{\omega} \left(N - \frac{N}{K}, N - \frac{2N}{K}, Q \right). \end{aligned} \quad (6.10)$$

В отличие от (6.6) здесь мы имеем ненулевую ковариацию.

6.4.4 Дисперсия скользящего экзамена

Критерий качества представляется математическим ожиданием функции потерь.

Обозначим $\xi_i = \mathcal{L}(y^i, \lambda_{k(i)}(x^i))$ – величину потерь на i -м объекте контрольной выборки.

Здесь λ_k – решающая функция, построенная по выборке с исключением k -го фолда, а $k(i)$ – номер фолда, которому принадлежит i -й элемент выборки.

Случайные величины ξ_i имеют одинаковое распределение. При этом они, как правило, зависимы. Под «как правило» здесь подразумевается, что зависимость имеет место во всех практически значимых ситуациях, хотя, вероятно, можно сконструировать искусственные примеры, когда зависимости нет.

В отличие от величин ζ_i , которые зависели только от одного объекта (контрольной) выборки, величины ξ_i являются функцией почти всей (обучающей) выборки (за исключением других объектов из того же фолда). При этом мы можем рассмотреть условную независимость величин ξ_i и ξ_j , когда все объекты выборки, кроме (x^i, y^i) и (x^j, y^j) , фиксированы. Величины, соответствующие объектам из одного фолда, будут условно независимы. Величины, соответствующие объектам, принадлежащим разным фолдам, вообще говоря, условно зависимы. Иными словами, величины ξ_i и ξ_j условно независимы, если $k(i) = k(j)$, и, как правило, зависимы, если $k(i) \neq k(j)$.

Может показаться, что зависимость между величинами ξ_i слабая и ею можно пренебречь. Это в корне неверно. Зависимость в некотором смысле действительно слабая, но пренебречь ею нельзя, и она существенно влияет на дисперсию оценки скользящего экзамена. Аналогично можно заметить, что в эмпирическом риске и скользящем экзамене выборки, по которым ведётся обучение, различаются всего одним объектом. Но получившиеся оценки кардинально разные в плане смещения.

Математическое ожидание $\mu = E\xi_i$ есть средний риск решения, построенного по выборке объёма $N - K$. Для дисперсии введём обозначение $\sigma^2 = D\xi_i$.

Имеет место разложение

$$\sigma^2 = \varsigma \left(N - \frac{N}{K}, Q \right) = \bar{\varsigma} \left(N - \frac{N}{K}, Q \right) + \zeta \left(N - \frac{N}{K}, Q \right),$$

где $\bar{\varsigma}(N, Q) = DR(S, Q)$, $\zeta(N, Q) = E_S D_{x,y|S} \mathcal{L}(y, \lambda_{Q,S}(x))$.

Оценка скользящего экзамена в текущих обозначениях принимает вид

$$\bar{\mu} = \frac{1}{N} \sum_{i=1}^N \xi_i.$$

Вычислим её дисперсию

$$D\bar{\mu} = E\bar{\mu}^2 - \mu^2 = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N (E\xi_i \xi_j - \mu^2).$$

На рисунке 6.2 показана структура получившийся двойной суммы.

При $i = j$ под суммой получаем просто σ^2 , т. е. $\varsigma \left(N - \frac{N}{K}, Q \right)$. Если $i \neq j$, но принадлежат одному и тому же фолду, то

$$E\xi_i \xi_j - \mu^2 = \bar{\varsigma} \left(N - \frac{N}{K}, Q \right).$$

Если i и j соответствуют разным фолдам, т. е. $k(i) \neq k(j)$, то

$$E\xi_i \xi_j - \mu^2 = \ddot{\omega} \left(N - \frac{N}{K}, N - \frac{2N}{K}, Q \right).$$

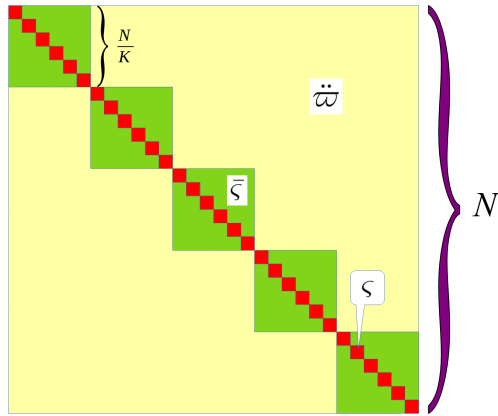


Рис. 6.2. Структура ковариационной матрицы потерь при кроссвалидации

Таким образом,

$$D\bar{\mu} = \frac{1}{N} \cdot \hat{\zeta} \left(N - \frac{N}{K}, Q \right) + \frac{1}{K} \cdot \bar{\zeta} \left(N - \frac{N}{K}, Q \right) + \frac{K-1}{K} \cdot \ddot{\omega} \left(N - \frac{N}{K}, N - \frac{2N}{K}, Q \right). \quad (6.11)$$

При группировке слагаемых мы предположили, что N делится на K и все фолды одного размера.

Теперь (6.10) можем записать в виде

$$E \left(\check{R}^K(S, Q) - R(S'_1, Q) \right)^2 = \frac{1}{N} \cdot \hat{\zeta} \left(N - \frac{N}{K}, Q \right) + \frac{K-1}{K} \cdot \left(\bar{\zeta} \left(N - \frac{N}{K}, Q \right) + \ddot{\omega} \left(N - \frac{N}{K}, N - \frac{2N}{K}, Q \right) - 2\dot{\omega} \left(N - \frac{N}{K}, N - \frac{2N}{K}, Q \right) \right). \quad (6.12)$$

Получили довольно громоздкое выражение, из чего можно заключить, что оценить погрешность скользящего экзамена непросто.

Из характеристик ζ , $\bar{\zeta}$, $\check{\omega}$, $\check{\omega}$ приемлемым образом (по одной выборке) можно оценить лишь ζ . Для остальных нужно множество выборок. А в реальных задачах выборка у нас одна.

В некоторых источниках, где говорится про дисперсию кроссвалидации (и утверждается что-то про её поведение) делается терминологическая ошибка (дисперсией называют то, что ею не является, например разброс по фолдам).

Польза выражения (6.12) в том, что можно понять, какой эффект дают модификации скользящего экзамена.

Например, можно видеть, что число фолдов практически несущественно влияет на погрешность (дисперсию) кроссвалидации. На практике использовать меньше пяти фолдов нецелесообразно, а при $K \geq 5$ выражение (6.12) от K зависит слабо.

Также можно сделать вывод, что повторная кроссвалидация тоже не даёт значительного эффекта. Для неё можно получить формулу, аналогичную (6.12), но более сложную (в ней будут присутствовать величины $\check{\zeta}$, $\check{\omega}$). При этом отличие будет практически несущественным.

6.4.5 Пример одинаково распределённых классов

В задаче классификации существует некоторый аналог «нулевой гипотезы», а именно класс вероятностных моделей, в которых распределения переменных для обоих классов полностью совпадают.

В этом случае любая решающая функция имеет вероятность ошибочной классификации 0,5, а дисперсия оценки риска по контрольной выборке равна $\frac{1}{4N^*}$.

Для таких распределений выражение (6.12) превращается в

$$\begin{aligned} E \left(\check{R}^K(S, Q) - R(S'_1, Q) \right)^2 &= D\bar{\mu} = \\ &= \frac{1}{4N} + \frac{K-1}{K} \cdot \check{\omega} \left(N - \frac{N}{K}, N - \frac{2N}{K}, Q \right). \end{aligned}$$

Для одинаково распределённых классов очень просто провести исследование методом статистического моделирования. Достаточно

но задать, например, равномерное распределение по всем переменным. Далее можно генерировать выборки и проводить процедуру кроссвалидации.

Такое моделирование позволяет оценить, насколько дисперсия скользящего экзамена превосходит дисперсию по контрольной выборке.

Для одинаково распределённых классов это различие объясняется исключительно величиной $\ddot{\omega}$.

6.4.6 Статистические оценки

Рассмотрим, какие статистические оценки можно построить в рамках процедуры кроссвалидации. В частности, нас интересует, возможно ли оценить дисперсию $\sigma^2 = D\xi_i = \varsigma$.

Рассмотрим оценку

$$\tilde{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^N (\xi_i - \bar{\mu})^2.$$

Аналогичная оценка в случае контрольной выборки оказалась несмещенной оценкой для σ^2 , однако при вычислении её математического ожидания мы существенно использовали независимость величин ζ_i . Но величины ξ_i зависимы, поэтому для кроссвалидации результаты будут иными.

Вычислим математическое ожидание оценки $\tilde{\sigma}^2$. Можно показать, что

$$E\tilde{\sigma}^2 = \frac{N}{N-1} \cdot (\sigma^2 - D\bar{\mu}).$$

Для ξ_i дисперсия суммы не равна сумме дисперсий, и $D\bar{\mu} \neq \frac{1}{N}\sigma^2$.

Таким образом, $\tilde{\sigma}^2$ позволяет оценить разность величин σ^2 и $D\bar{\mu}$. Поскольку связь между этими величинами нам неизвестна, но σ^2 намного больше $D\bar{\mu}$, мы можем использовать $\tilde{\sigma}^2$ как несколько смещённую оценку для σ^2 .

При этом оценить $D\bar{\mu}$ мы не можем.

Для скользящего экзамена в общем случае неизвестен доверительный интервал и нет хороших оценок дисперсии.

Эмпирические исследования показывают, что можно использовать следующую эвристическую оценку

$$D\bar{\mu} \approx \frac{C}{N} \tilde{\sigma}^2, \quad (6.13)$$

где $C \geq 1$ – некоторый поправочный коэффициент, значение которого зависит от метода и задачи. На практике бывает несколько больше 1, но не превосходит 2.

Пусть J_k – множество индексов объектов, принадлежащих k -му фолду, а N_k – размер фолда.

Обозначим

$$\bar{\mu}_k = \frac{1}{N_k} \sum_{i \in J_k} \xi_i.$$

Для удобства будем считать, что N делится на K . Положим все N_k одинаковыми, т. е. $N_k \equiv \frac{N}{K}$.

Тогда все $\bar{\mu}_k$ будут, вообще говоря, зависимыми одинаково распределёнными случайными величинами со средним μ .

Исследуем, может ли быть полезной оценка

$$\tilde{\sigma}_K^2 = \frac{1}{K-1} \sum_{k=1}^K (\bar{\mu}_k - \bar{\mu})^2.$$

Вычислим математическое ожидание и узнаем, что же она оценивает

$$E\tilde{\sigma}_K^2 = \frac{K}{N} \cdot \zeta\left(N - \frac{N}{K}, Q\right) + \bar{\zeta}\left(N - \frac{N}{K}, Q\right) - \ddot{\omega}\left(N - \frac{N}{K}, N - \frac{2N}{K}, Q\right).$$

Получается, что $\tilde{\sigma}_K^2$ оценивает некоторую комбинацию характеристик, которая не имеет содержательной интерпретации.

Кроме того, $\tilde{\sigma}_K^2$ – плохая оценка. В параграфе о контрольной выборке мы выяснили, что аналогичная оценка при $K < N$ имеет большую погрешность и её использование не имеет рационального оправдания. Использовать её, например, при $K = 10$ – это

практически то же самое, что взять «с потолка» какие-то произвольные числа.

При $K = N$ получаем $\tilde{\sigma}_K^2 = \tilde{\sigma}^2$. Но в этом случае нужно использовать поправку, учитывающую, что дисперсия скользящего экзамена существенно выше, чем у контрольной выборки.

Таким образом, использовать в качестве оценки для $D\bar{\mu}$ величину $\frac{\tilde{\sigma}_K^2}{K}$ — это грубая ошибка.

Можно ли по скользящему экзамену построить хоть какую-то несмещённую оценку дисперсии? Отчасти, да. Оценка

$$\check{\sigma}_K^2 = \frac{1}{N - K} \sum_{j=1}^N (\xi_j - \bar{\mu}_{k(j)})^2$$

является несмещённой оценкой для $\zeta(N - \frac{N}{K}, Q)$ — средней дисперсии потерь для решения, построенного по выборке объёма $N - \frac{N}{K}$. Однако эта величина также не поможет нам оценить $D\bar{\mu}$.

Под средней дисперсией здесь понимается величина $ED_{\lambda_k(i)}\xi_i$, т. е. дисперсия берётся при фиксированной решающей функции. Это понятие аналогично среднему риску. Заметим, что средняя дисперсия — это не то же, что σ^2 .

6.4.7 Разновидности скользящего экзамена

Идея скользящего экзамена допускает множество реализаций и вариантов. Перечислим некоторые из них:

- ✧ leave-one-out;
- ✧ k -fold crossvalidation;
- ✧ случайные подвыборки;
- ✧ повторные разбиения;
- ✧ кроссвалидация со стратификацией;
- ✧ bootstrap.

Про первые варианты мы уже рассказали, кратко опишем остальные.

Идея случайных подвыборок заключается в том, что выборка разделяется на тестовую и обучающую части путём случайного

выбора (без повторения) объектов для теста. При этом долю объектов для теста выбирают не слишком большой (до 20%), чтобы не было заметного смещения. На оставшихся объектах проводится обучение, и для выбранных делается прогноз. Данные шаги продельваются многократно. Для каждого прогноза вычисляется функция потерь, и все потери усредняются.

Данный метод имеет свойства, схожие с методом кроссвалидации, однако требует больших вычислительных затрат (нужно проводить итерации, пока почти все объекты не побывают в роли тестовых), по-видимому, не давая заметного повышения точности.

Более предпочтительной выглядит кроссвалидация с повторением. Идея повторных разбиений заключается в том, кроссвалидация проводится многократно, причём на каждой итерации выборка случайным образом «перемешивается». В качестве оценки риска берётся среднее значение потерь по всем итерациям.

Повторные разбиения обычно позволяют несколько повысить точность оценки риска, но эффект, как правило, незначителен. С другой стороны, вреда от такой процедуры быть не может, поэтому провести несколько итераций бывает оправдано.

Встречается грубое заблуждение, будто большим числом повторений можно сколь угодно уменьшать дисперсию оценки. На самом деле, все повторы делаются на одной и той же выборке, поэтому получаемые оценки сильно зависимы и дисперсия оценки практически не уменьшается. Иными словами, чудес не бывает, и никому ещё не удалось успешно применить метод Мюнхгаузена по вытаскиванию себя из трясины.

Также встречается заблуждение, что точность оценки можно существенно повысить, увеличивая число фолдов. Оно основано, видимо, на том, что при независимых измерениях ошибка уменьшается как $\frac{1}{N}$, поэтому ожидают, что при K фолдах ошибка будет порядка $\frac{1}{K}$. На самом деле, дисперсия оценки если и зависит от K , то несущественно.

Здесь необходимо сделать важную оговорку. В некоторых случаях увеличение K или же применение повторных разбиений всё же даёт заметный эффект.

Он объясняется следующим фактором. Многие (пожалуй, даже большинство) методов имеют не вполне детерминированное поведение. Многие в явном виде используют случайность (которой можно управлять через `random_seed`). У многих получаемое решение зависит от порядка следования объектов выборки.

В результате по одной и той же выборке мы можем получить одним методом разные решения. Понятно, что такой разброс даёт «лишнюю прибавку» к погрешности (дисперсии) оценок. И именно эта добавка уменьшается за счёт повторных разбиений.

Но такой же эффект мы можем получить, «усовершенствовав» сам метод, а именно включив в него усреднение решений с разным `random_seed` и с разным порядком объектов в обучении.

Если в задаче классификации классы несбалансированы, т. е. вероятности классов сильно отличаются, то при разбиении выборки в разных фолдах частота объектов «редкого» класса может сильно различаться (например, в какой-то фолд могут вообще не попасть объекты одного из классов).

Идея стратификации заключается в том, чтобы разбивать выборку так, чтобы соотношение классов по фолдам было примерно одинаковым. Этого легко добиться, например отдельно разбить на фолды объекты каждого класса, а затем соответствующие фолды объединить.

Сложно сказать, стоит ли ожидать от стратификации значительного эффекта. Нет доказательства, что стратификация уменьшает дисперсию оценки. Обучение при кроссвалидации проводится на выборках, мало отличающихся от исходной, т. е. баланс классов в них сохранён, и кажется, что стратификация не должна заметно влиять на оценку.

Наиболее полезные выводы относительно разных вариантов кроссвалидации можно сделать на основе статистического моделирования. Пример такого моделирования приведён в конце данной главы.

Автор рекомендует читателям провести аналогичные эксперименты и сделать собственные выводы относительно лучшего способа оценки риска.

Существует вариант кроссвалидации, при котором в обучении не участвуют сразу два фолда: один из них используется как тестовый, а второй — для раннего останова (например, для выбора числа эпох или количества деревьев, т. е. для настройки некоторого гиперпараметра). Этот способ внешне выглядит привлекательным, но он сомнителен, поскольку решение об останове принимается по очень малому объёму данных.

Более правильным вариантом для подбора гиперпараметров будет вложенная кроссвалидация, когда на каждой итерации на обучающей подвыборке проводится «внутренняя» кроссвалидация. Однако такой вариант во многих случаях будет чрезмерно трудоёмким, поэтому можно ограничиться простой кроссвалидацией, осуществляя по ней и подбор гиперпараметра (останова), и финальную оценку риска. Да, такая оценка риска будет несколько смещённой, но погрешность из-за этого смещения будет незначительной. Вполне вероятно, что смещение здесь будет «меньшим злом», чем промах с моментом останова (при настройке гиперпараметра по единственному фолду).

Оценка bootstrap есть

$$\hat{R}(S, Q) = \hat{E} \frac{1}{|J_0|} \sum_{i \in J_0} \mathcal{L}(y^i, \lambda_{Q, \hat{S}}(x^i)),$$

где \hat{S} — выборка, получаемая из S путем N -кратного случайного (равновероятного) выбора ее значений с повторениями, J_0 — множество индексов объектов из S , ни разу не выбранных в \hat{S} , математическое ожидание \hat{E} подразумевает усреднение по бутстреп-выборкам \hat{S} .

Ввиду того, что оценка bootstrap является смещенной (поскольку для обучения используется лишь $1 - e^{-1}$ доля объектов), чаще используют ее в комбинации с эмпирическим риском

$$\hat{R}'(S, Q) = e^{-1} \tilde{R}(S, Q) + (1 - e^{-1}) \hat{R}(S, Q).$$

Автору неизвестны убедительные свидетельства о преимуществах оценки bootstrap по сравнению с кроссвалидацией.

6.4.8 Оценка out-of-bag

Чаще всего скользящий экзамен используется в виде K -fold кроссвалидации, когда обучающая выборка разбивается на K непересекающихся подмножеств (фолдов) и при обучении тестируемый объект удаляется из выборки вместе со всем фолдом, которому он принадлежит.

Как правило, после проведения скользящего экзамена проводится обучение метода уже на всей выборке, и полученная решающая функция используется в качестве итогового решения.

Однако в качестве итогового решения можно использовать решающие функции, уже построенные в ходе кроссвалидации, попросту объединив их в ансамбль.

В обоих случаях оценкой качества выступает одно и то же выражение — функционал скользящего экзамена. Однако решающие функции, качество которых он оценивает, разные. Поэтому с формальной точки зрения это разные оценки. И во втором случае будем называть этот функционал оценкой out-of-bag.

Таким образом, сравнение оценок out-of-bag и скользящего экзамена фактически означает сравнение двух способов построения итогового решения (а функционал качества в обоих случаях одинаковый).

Заметим, что первоначально оценка out-of-bag появилась в методе бэггинга (RandomForest), где она выглядит несколько иначе, чем мы определили, однако основная идея та же.

Оценка out-of-bag является, вообще говоря, смещённой.

Например, в методе RandomForest для каждого объекта выборки число деревьев, которые не использовали этот объект для обучения, составляет примерно треть, т. е. $\frac{1}{e}$, от общего их количества. Получается, что оценка out-of-bag соответствует качеству леса, примерно втрое меньшего размера, поэтому такая оценка риска несколько завышена.

В целом, использование out-of-bag представляется удачным сочетанием идей скользящего экзамена и ансамблирования.

6.4.9 Свойства оценки скользящего экзамена

Мы установили следующие свойства скользящего экзамена:

- ✧ относительная простота вычисления;
- ✧ несмещённость (не в том смысле, в котором хотелось бы);
- ✧ состоятельность (может быть не доказана);
- ✧ большая дисперсия, для которой нет хороших оценок;
- ✧ среднеквадратичный разброс по фолдам (практически) никак не связан с дисперсией скользящего экзамена и не может выступать в роли её оценки.

Несмещённость данной оценки не означает, что, получив некоторое значение на кроссвалидации, мы вправе оценить риск этим значением. Как и в случае контрольной выборки, его следует скорректировать в большую сторону.

Что касается состоятельности, то для кроссвалидации с фиксированным K она очевидна (следует из состоятельности оценки hold-out), но при $K = N$ простого доказательства не получается (в виду зависимости получаемых величин).

Любой вариант скользящего экзамена имеет большую дисперсию. Есть эмпирические свидетельства, что дисперсия скользящего экзамена бывает до двух раз выше дисперсии оценки по контрольной выборке.

У скользящего экзамена есть множество вариантов, однако опыт свидетельствует, что они примерно равноценны по качеству, а значит, имеет смысл использовать что проще.

Если вычислительные ресурсы ограничены, то разумный выбор — кроссвалидация с $K = 5$. Если ресурсов с избытком, то имеет смысл взять $K = 10$. Предварительно объекты следует «перемешать», поскольку в исходной таблице они могли идти в неслучайном порядке.

6.5 Скользящий экзамен и дисперсионный анализ ☆

При первом чтении параграф можно пропустить.

Скольльзящий экзамен на практике используется, в том числе, для подбора гиперпараметров модели, а также для выбора самой модели (метода машинного обучения). Однако его можно использовать [35] и для выбора вероятностной модели, т. е. для проверки статистических гипотез.

6.5.1 Задача оценивания зависимостей

Задачей оценивания зависимостей будем называть задачу построения решающей функции в случае, когда в качестве значений целевой переменной Y выступает множество действительных чисел.

В задаче оценивания регрессии используется квадратичная функция потерь $\mathcal{L}(y, y') = (y - y')^2$. В этом случае риск есть средний квадрат отклонения.

Обозначим $\bar{y} = \frac{1}{N} \sum_{i=0}^N y_i$ – среднее значение y по всей выборке.

Пусть метод Q_0 в качестве решения возвращает \bar{y} . Тогда оценка качества этого метода посредством скользящего экзамена даст величину

$$\check{R}(S, Q_0) = \frac{1}{N} \sum_{i=1}^N \left(y_i - \frac{N\bar{y} - y_i}{N-1} \right)^2 = \frac{N}{(N-1)^2} \sum_{i=1}^N (y_i - \bar{y})^2.$$

Здесь выражение $\frac{N\bar{y} - y_i}{N-1}$ есть прогноз, получаемый методом Q_0 при удалении из выборки i -го объекта.

Оценка скользящего экзамена оказалась связанной со стандартной оценкой дисперсии $\tilde{D}y = \frac{1}{N-1} \sum_{i=1}^N (y_i - \bar{y})^2$, а именно:

$$\check{R}(S, Q_0) = \frac{N}{N-1} \tilde{D}y.$$

Для демонстрации техники работы с оценкой скользящего экзамена проведём некоторые выкладки. Вычислим

$$ER(c, \lambda_{Q_0, V}) = E(y - \bar{y})^2 = Dy + D\bar{y} = Dy \cdot \frac{N+1}{N}.$$

Используя несмещённость ((6.8)) оценки скользящего экзамена, получаем

$$E\check{R}(S, Q_0) = ER(c, \lambda_{Q_0, V_{N-1}}) = Dy \cdot \frac{N}{N-1}.$$

Таким образом, устанавливаем общеизвестный факт

$$Dy = E\check{D}y.$$

Мы доказали несмещённость стандартной оценки дисперсии $\check{D}y$ как следствие несмещённости оценки скользящего экзамена. Это доказательство не проще общепринятого, однако полезно как пример использования свойств скользящего экзамена.

6.5.2 Гипотеза однородности

Задача дисперсионного анализа по большей части сводится к проверке гипотезы однородности.

Предположим, что единственная переменная X является категориальной и принимает значения, которые мы закодируем целыми числами $\{1, \dots, M\}$.

Обозначим $N_x = \sum_{i=1}^N I(x_i = x)$ – количество выборочных точек с $x_i = x$ и $\bar{y}_x = \frac{1}{N_x} \sum_{i=1}^N y_i \cdot I(x_i = x)$ – среднее по выборке значение y в точке x . Здесь $I(\cdot)$ – индикаторная функция (равна 1, когда условие истинно, и 0, когда ложно).

Обозначим $\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$ – среднее значение y по всей выборке.

Гипотеза однородности заключается в предположении, что математическое ожидание целевой переменной не зависит от X . Этой гипотезе соответствует метод прогнозирования Q_0 .

Статистика Фишера для проверки гипотезы однородности во введённых обозначениях запишется как

$$F(S) = \frac{(N - M) \sum_{x=1}^M N_x (\bar{y}_x - \bar{y})^2}{(M - 1) \sum_{i=1}^N (y_i - \bar{y}_{x_i})^2}.$$

Альтернативной гипотезе (неоднородности) соответствует метод Q_1 , который делает разные прогнозы в зависимости от X . Метод Q_1 в качестве решения возвращает $\lambda(x) = \bar{y}_x$. Оценка качества этого метода посредством скользящего экзамена даст величину

$$\begin{aligned} \check{R}(S, Q_1) &= \frac{1}{N} \sum_{i=1}^N \left(y_i - \frac{N_{x_i} \bar{y}_{x_i} - y_i}{N_{x_i} - 1} \right)^2 = \\ &= \frac{1}{N} \sum_{i=1}^N \left(\frac{N_{x_i}}{N_{x_i} - 1} \right)^2 (y_i - \bar{y}_{x_i})^2. \end{aligned}$$

Введём обозначения:

$$z_i = y_i - \bar{y}_{x_i}, \quad \bar{z}_x = \bar{y}_x - \bar{y}, \quad d_x^2 = \sum_{i=1}^N z_i^2 \cdot I(x_i = x).$$

В этих обозначениях

$$\begin{aligned} \check{R}(S, Q_0) &= \frac{N}{(N - 1)^2} \sum_{i=1}^N (z_i + \bar{z}_{x_i})^2 = \frac{N}{(N - 1)^2} \sum_{i=1}^N (z_i^2 + \bar{z}_{x_i}^2) = \\ &= \frac{N}{(N - 1)^2} \sum_{x=1}^M (d_x^2 + N_x \bar{z}_x^2). \end{aligned}$$

Аналогично

$$\check{R}(S, Q_1) = \frac{1}{N} \sum_{i=1}^N \left(\frac{N_{x_i}}{N_{x_i} - 1} \right)^2 z_i^2 = \frac{1}{N} \sum_{x=1}^M \left(\frac{N_x}{N_x - 1} \right)^2 d_x^2.$$

Также преобразуем выражение для статистики Фишера

$$F(S) \cdot (M - 1) \sum_{x=1}^M d_x^2 = (N - M) \sum_{x=1}^M N_x \bar{z}_x^2.$$

Рассмотрим статистику

$$\begin{aligned} \check{S}(S) &= N\check{R}(S, Q_0) - N\check{R}(S, Q_1) = \\ &= \left(\frac{N}{N-1} \right)^2 \cdot \sum_{x=1}^M (d_x^2 + N_x \bar{z}_x^2) - \sum_{x=1}^M \left(\frac{N_x}{N_x - 1} \right)^2 \cdot d_x^2. \end{aligned}$$

После подстановки статистики Фишера

$$\begin{aligned} \check{S}(S) &= \left(\frac{N}{N-1} \right)^2 \cdot \left(1 + F(S) \cdot \frac{M-1}{N-M} \right) \cdot \sum_{x=1}^M d_x^2 - \\ &\quad - \sum_{x=1}^M \left(\frac{N_x}{N_x - 1} \right)^2 \cdot d_x^2. \end{aligned} \tag{6.14}$$

Скольльзящий экзамен является общепринятым способом выбора метода Q при решении практических задач. Из нескольких методов выбирается тот, который даёт наименьшую ошибку на скольльзящем контроле.

Заметим, однако, что методы Q_0 и Q_1 фактически основаны на гипотезах соответственно равенства и неравенства средних. Иными словами, выбор метода Q_1 , по сути, означает, что гипотеза однородности выборки отвергается.

Таким образом, метод скольльзящего экзамена даёт статистический критерий, который отвергает гипотезу равенства средних при $\check{S}(S) > 0$.

Теорема 1 Пусть $N_x = \frac{N}{M}$. Тогда условие $\check{S}(S) > 0$ равносильно

$$F(S) > 1 + \frac{N-1}{N-M}.$$

Доказательство непосредственно следует из (6.14).

Мы выявили достаточно любопытный факт, а именно, что метод скользящего экзамена эквивалентен некоторому статистическому критерию, уровень значимости для которого определяется параметрами данных и не может устанавливаться произвольно.

В случае, когда все N_x равны, что метод скользящего экзамена эквивалентен критерию Фишера, с пороговым значением статистики $1 + \frac{N-1}{N-M} \approx 2$, что при $M = 2$ соответствует уровню значимости около 0,157.

Заметим, что с ростом M уровень значимости уменьшается (стремится к 0), так как число степеней свободы растёт, а пороговое значение остаётся примерно одинаковым.

6.5.3 Байесовская модель

Для содержательной интерпретации дальнейших выкладок полезно рассмотреть пример с реальными данными.

Рассмотрим задачу Zestimate (Zillow Prize: Zillow's Home Value Prediction, <https://www.kaggle.com/c/zillow-prize-1>).

Данные представляют собой значения 58 переменных, измеренные для 86795 объектов.

Целевая переменная Y' представляет собой некоторую характеристику, вычисляемую на основе цены объекта недвижимости.

В качестве переменной X будет выступать код региона, который принимает 317 различных значений.

К переменной Y' для удобства применим квантильное преобразование, т. е. введём переменную $Y = \tilde{F}(Y')$, $\tilde{F}(\cdot)$ – эмпирическая функция распределения.

Как и раньше, $\bar{y}_x = \frac{1}{N_x} \sum_{i=1}^N y_i \cdot I(x_i = x)$ – среднее по выборке значение y в точке x .

Введём величины $\bar{u}_x = \sqrt{12 \cdot N_x} \cdot (\bar{y}_x - 0,5)$. «Магическое» число 12 связано с дисперсией равномерного распределения.

Если справедлива гипотеза однородности (т. е. если Y не зависит от X), то величины \bar{u}_x имеют распределение, близкое к нормальному с параметрами 0, 1.

На рис. 6.3 приведена (круглые серые маркеры) эмпирическая функция распределения $\tilde{F}(\bar{u}_x)$. Сплошная чёрная кривая изображает функцию нормального распределения с параметрами $-0,06$, $2,07$, оцененными по обучающей выборке.

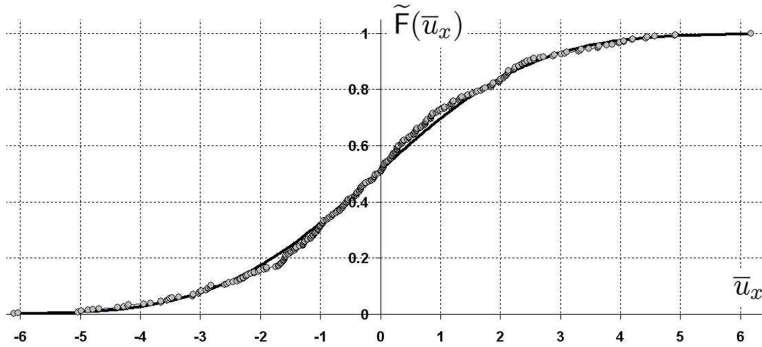


Рис. 6.3. Эмпирическая функция распределения оценок среднего в задаче Zestimate

Видим, что распределение очень похоже на нормальное, но дисперсия значимо больше 1, поэтому поведение \bar{u}_x не может быть объяснено случайными флуктуациями частот.

Введём обозначения $\xi_x = E[Y | x]$ – условное математическое ожидание целевой переменной в точке x и $\omega_x = \bar{u}_x - \xi_x$.

Предположим, что целевая переменная подчиняется модели $Y_x = \xi_x + \omega$, $E\omega = 0$, $D\omega = \sigma^2$.

Тогда наблюдаемые значения \bar{y}_x являются реализациями случайной величины $\zeta_x = \xi_x + \omega_x$, $E\omega_x = 0$, $D\omega_x = \frac{\sigma^2}{N_x}$.

В силу центральной предельной теоремы величина ω_x имеет приближённо нормально распределение.

В исходной постановке задачи значения ξ_x не являются случайными, но \bar{y}_x эмпирически выглядит как величина с нормальным распределением.

Это даёт основания рассмотреть байесовскую постановку, в которой ξ_x будет случайной величиной с нормальным распреде-

лением.

В этой модели оптимальное решение есть

$$\lambda(x) = E[\xi_x | \zeta_x = \bar{y}_x] = EY + (\bar{y}_x - EY) \cdot \alpha_x = EY \cdot (1 - \alpha_x) + \bar{y}_x \alpha_x,$$

где

$$\alpha_x = 1 - \frac{D\omega_x}{D\zeta_x} = 1 - \frac{\sigma^2}{N_x D\zeta_x}.$$

При построении решающей функции по выборке естественно взять несмещённые оценки неизвестных параметров:

$$EY \approx \bar{y}, \quad \sigma^2 \approx \frac{1}{N - M} \sum_{i=1}^N (y_i - \bar{y}_{x_i})^2, \quad D\zeta_x \approx \frac{1}{M - 1} \sum_{x=1}^M (\bar{y}_x - \bar{y})^2.$$

В случае, когда все N_x одинаковы, при подстановке этих оценок в выражение для α_x получаем $\alpha_x = 1 - \frac{1}{F(S)}$. Последнее выражение представляется достаточно интересным. В следующем подразделе мы его получим ещё раз, из других соображений.

6.5.4 Усреднённое решение

Методы Q_0 и Q_1 строят решения, исходя из того, принимается или отвергается гипотеза однородности. Поскольку мы не можем быть полностью уверены в справедливости гипотез, представляется естественным рассмотреть решения, основанные на усреднении базовых решающих функций.

Пусть метод Q_α в качестве решения возвращает $\lambda(x) = \alpha \bar{y}_x + (1 - \alpha) \bar{y}$. Тогда оценка качества этого метода посредством скользящего экзамена даст величину

$$\check{R}(S, Q_\alpha) = \frac{1}{N} \sum_{i=1}^N \left(y_i - \frac{N_{x_i} \bar{y}_{x_i} - y_i}{N_{x_i} - 1} \cdot \alpha - \frac{N \bar{y} - y_i}{N - 1} \cdot (1 - \alpha) \right)^2.$$

После преобразований имеем

$$\check{R}(S, Q_\alpha) = \frac{1}{N} \sum_{i=1}^N \left(\frac{N_{x_i}}{N_{x_i} - 1} (y_i - \bar{y}_{x_i}) \alpha + \frac{N}{N - 1} (y_i - \bar{y}) (1 - \alpha) \right)^2.$$

Выражаем через z_i , \bar{z}_x , d_x^2 , которые введены в предыдущем подразделе.

$$\begin{aligned}\check{R}(S, Q_\alpha) &= \frac{N}{(N-1)^2} \sum_{i=1}^N (C_{x_i} z_i \cdot \alpha + (z_i + \bar{z}_{x_i}) \cdot (1-\alpha))^2 = \\ &= \frac{N}{(N-1)^2} \sum_{x=1}^M ((C_x \alpha + 1 - \alpha)^2 d_x^2 + \bar{z}_x^2 N_x \cdot (1-\alpha)^2),\end{aligned}$$

где $C_x = \frac{N_x \cdot (N-1)}{(N_x-1) \cdot N}$.

Подставим статистику Фишера

$$\begin{aligned}\check{R}(S, Q_\alpha) &= \\ &= \frac{N}{(N-1)^2} \sum_{x=1}^M \left((C_x \alpha + 1 - \alpha)^2 d_x^2 + F(S) \cdot \frac{M-1}{N-M} \cdot (1-\alpha)^2 d_x^2 \right).\end{aligned}$$

Обозначим $\alpha^* = \arg \min_{\alpha} \check{R}(S, Q_\alpha)$.

Теорема 2 При $N_x = \frac{N}{M}$ имеет место

$$\alpha^* = \frac{F(S) - 1}{F(S) + \frac{M-1}{N-M}} \approx 1 - \frac{1}{F(S)}.$$

Доказательство.

Из $N_x = \frac{N}{M}$ находим $C_x = \frac{N-1}{N-M}$.

Теперь

$$\begin{aligned}\check{R}(S, Q_\alpha) &= \\ &= \frac{N}{(N-1)^2} \left(\left(\frac{M-1}{N-M} \alpha + 1 \right)^2 + F(S) \cdot \frac{M-1}{N-M} (1-\alpha)^2 \right) \cdot \sum_{x=1}^M d_x^2.\end{aligned}$$

Дифференцируя выражение по α и приравнявая производную к 0, получаем искомое.

Заметим, что при $F(S) = 2$, что соответствует пороговому значению, при котором нулевая гипотеза отвергается, имеем $\alpha^* \approx 0.5$. Это вполне логично: когда гипотезы одинаково правдоподобны, решения усредняются с равными весами.

6.5.5 Обсуждение результатов

В разделе рассматривалась классическая задача дисперсионного анализа. Пусть, например, задано некоторое множество городов, для каждого из которых имеется выборка объектов недвижимости с известными ценами. Требуется проверить гипотезу, одинакова ли средняя цена недвижимости в этих городах.

Один из стандартных методов решения этой задачи заключается в применении критерия Фишера.

Однако вместо явной проверки гипотез мы можем построить две решающие функции: первую в предположении равенства ожидаемой цены недвижимости по городам, вторую — в предположении, что цены разные. Предположение, при котором решение будет оценено как более точное, мы и примем в качестве гипотезы.

Для оценки точности решения одним из общепринятых подходов является метод скользящего экзамена.

Оказывается, что применение скользящего экзамена для задачи дисперсионного анализа эквивалентно классическому критерию Фишера, причём с «магическим» числом в качестве уровня значимости (которое соответствует значению статистики Фишера, равному 2). Например, для случая двух подвыборок этот уровень значимости примерно равен 0.16. При этом с ростом числа подвыборок (числа городов в примере) этот уровень значимости стремится к 0.

Этот факт выглядит непривычно. В классическом подходе к проверке гипотез уровень значимости выбирается произвольно. Полученный результат можно интерпретировать как то, что некоторые значения уровня значимости могут считаться в определённом смысле «более обоснованными». С другой стороны, слишком малые значения уровня значимости (при больших M) могут свидетельствовать о том, что оценка скользящего экзамена в этих условиях не является адекватной.

Далее рассматривалась задача, которую можно считать одной из постановок задачи регрессионного анализа. В качестве примера можно рассмотреть те же выборки недвижимости по городам, но цель теперь в том, чтобы оценить математическое ожидание цены

недвижимости в каждом городе.

Решающая функция, оптимальная (в классе линейных комбинаций двух базовых моделей) с точки зрения критерия скользящего экзамена, оказывается совпадающей (при некоторых условиях) с решением, оптимальным в рамках некоторой байесовской постановки с нормальным распределением параметров модели.

Полученные результаты демонстрируют связь методов машинного обучения с методами проверки статистических гипотез.

6.6 Практические рекомендации

Кроссвалидацию всегда следует проводить в режиме out-of-fold, т. е. объединять предсказания по фолдам, получив прогнозы для всей выборки, а затем по всем прогнозам оценивать риск. При таком способе можно оценить даже неаддитивные метрики, например, ROC-AUC.

Значительным ресурсом для повышения точности решения является калибровка прогноза под целевую метрику.

Например, если используется такая экзотическая метрика как квадратичное взвешенное каппа, то повысить значение этой метрики можно просто путём масштабирования, сравнив дисперсию прогноза с дисперсией целевой переменной.

6.6.1 Оценивание дисперсии

Некоторые пользователи методов машинного обучения вычисляют среднеквадратичный разброс по фолдам.

Полученная величина не имеет никакого смысла. По крайней мере её категорически нельзя использовать как оценку дисперсии скользящего экзамена (т. е. как оценку его точности).

Вместо этого можно использовать другую оценку (6.13), которая тоже эвристическая, но она имеет хоть какие-то эмпирические обоснования.

Точнее, чем на отдельной контрольной выборке, риск оценить невозможно. Погрешность такой оценки порядка $\frac{1}{\sqrt{N^*}}$.

В некоторых случаях смещение эмпирического риска оказывается даже меньше этой величины, и тогда можно использовать эмпирический риск (а в кроссвалидации нет необходимости). На практике этим способом следует пользоваться с осторожностью, поскольку легко ошибиться в оценке сложности модели. Однако есть достаточно очевидные примеры, когда число параметров модели много меньше объёма выборки. В этих редких случаях можно обходиться без тестовой выборки.

6.6.2 Валидация и тест

Встречается путаница в терминах: «контрольная», «тестовая» и «валидационная» выборки.

Термины «контрольная», «тестовая» выборки будем использовать как синонимы.

Валидационная выборка — та, по которой настраиваются гиперпараметры.

В виде исключения, процедуру кроссвалидации мы будем всегда называть кроссвалидацией (а не кросс-тестом), даже если по ней настройка гиперпараметров не производилась.

Оценка по валидационной выборке, вообще говоря, смещённая. Обычно это смещение небольшое. Однако на соревнованиях может оказаться существенным.

Важно понимать, что выбор метода — это тоже подбор гиперпараметра. Если по тестовой выборке принимается решение о выборе метода, то выборка становится валидационной. Сам метод (его название) — это тоже гиперпараметр модели.

На соревнованиях: публичный лидерборд — валидационная выборка, приватный лидерборд — тестовая выборка. При этом для выбора гиперпараметров результат публичного лидерборда стоит усреднять с оценкой по кроссвалидации.

При больших выборках (от 100000) можно выделить обучающую, валидационную и тестовую подвыборки.

При небольших выборках (до 5000 объектов) разумнее использовать кроссвалидацию как для подбора гиперпараметров, так и

для финального оценивания. Несмотря на смещённость, это лучше, чем выделять часть данных на тест (погрешность из-за дисперсии будет выше смещения).

При несбалансированных классах «реальным» объёмом выборки является число объектов самого редкого класса (умноженное на число классов).

При кроссвалидации для детерминированных алгоритмов достаточно 5 фолдов, для стохастических лучше взять 10.

Целесообразность повторных разбиений, как и стратификации, вызывает сомнения. Однако «вреда» от них быть не может. А поскольку отсутствие «пользы» не доказано, применение имеет некоторый смысл.

Использование бутстреп выборок кажется неоправданным, поскольку точность такой оценки вполне может быть ниже, чем для кроссвалидации.

6.7 Выводы

По материалам главы можно сделать следующие выводы:

- ✧ наилучший способ оценивания риска неизвестен;
- ✧ на практике обычно используют скользящий контроль;
- ✧ все разновидности скользящего контроля примерно равноценны;
- ✧ нет точных оценок доверительного интервала для риска по обучающей выборке;
- ✧ полезно использовать статистическое моделирование.

Оценивание качества решений — одна из самых важных и сложных проблем машинного обучения, которая до настоящего времени остаётся открытой.

Accuracy							
	train	true	test	cv5	cv10	cv5_rep	cv10_str
mean	0.9335	0.8429	0.8423	0.8379	0.8422	0.838	0.8386
std	0.0091	0.0068	0.0122	0.0121	0.0129	0.012	0.0129

Logloss							
	train	true	test	cv5	cv10	cv5_rep	cv10_str
mean	0.2641	0.4047	0.3984	0.4045	0.4017	0.4047	0.4033
std	0.0201	0.0078	0.0208	0.0229	0.0230	0.0219	0.0219

Рис. 6.4. Результаты моделирования. В верхней таблице — значения точности, в нижней — логарифмические потери

6.8 Примеры кода

Приведём пример кода для моделирования построения решений на случайных выборках.

Автор настоятельно рекомендует читателям провести подобный эксперимент, чтобы непосредственно понаблюдать поведение оценок скользящего экзамена.

Сначала рассмотрим функцию генерации данных.

Аргументами функции являются размер выборки, число переменных и байесовский уровень ошибки. Все переменные, за исключением первых двух, являются шумовыми (не зависят от целевой переменной).

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold,
    StratifiedKFold, RepeatedKFold

def gen_data(N,n,by_err):
    df = pd.DataFrame(np.random.uniform(-1.0, 1.0, (N,n)),
        columns=['X%i' % (i+1) for i in range(n)])

```

```

A = (df['X1']*df['X1'] - df['X1']*df['X2']
      + df['X2']*df['X2'] < 0.7**2).astype(int)
df['Y'] = np.random.uniform(0.0, 1.0, N)
df['Y'] = (df['Y']>=(by_err+(1-2*by_err)*A)).astype(int)
return df

```

```

df = gen_data(1000,3,0.1)
ax = df.loc[df['Y']==0].plot.scatter(x='X1',y='X2',
                                     figsize=(8,8))
df.loc[df['Y']==1].plot.scatter(x='X1',y='X2',c='red',ax=ax)

```

В конце приведённого фрагмента сгенерированная выборка визуализируется в пространстве информативных переменных, и можно наглядно посмотреть форму распределения.

Ниже приведены функция вычисления оценки точности по кроссвалидации и функция моделирования.

```

def crossval_score(X,y,model,kf):
    scor = []
    for train, test in kf.split(X,y):
        X_train, X_test = X.iloc[train], X.iloc[test]
        y_train, y_test = y.iloc[train], y.iloc[test]
        clf = model.fit(X_train,y_train)
        scor.append(clf.score(X_test,y_test))
    return np.array(scor).mean()

def simulate():
    model = GradientBoostingClassifier(n_estimators=150,
                                       learning_rate=0.025, max_depth=5, random_state=0)
    df = gen_data(1000,5,0.1)
    df1 = gen_data(1000,5,0.1)
    dft = gen_data(100000,5,0.1)
    X = df.iloc[:, :-1]
    y = df.iloc[:, -1]
    model.fit(X,y)
    scores = {
        'train': model.score(X,y),
        'true': model.score(dft.iloc[:, :-1], dft.iloc[:, -1]),
        'test': model.score(df1.iloc[:, :-1], df1.iloc[:, -1]),
        'cv5': crossval_score(X,y,model,KFold(5)),
        'cv10': crossval_score(X,y,model,KFold(10)),
        'cv5_rep': crossval_score(X,y,model,
                                   RepeatedKFold(n_splits=5,n_repeats=3)),
    }

```

```

'cv10_str': crossval_score(X,y,model,
                           StratifiedKFold(n_splits=5))}
return scores

```

```

dr100 = pd.DataFrame([simulate() for i in range(100)])
dr100.agg(['mean', 'std']).round(4)

```

Функция `simulate` генерирует случайную выборку и на ней проводит обучение модели, а также вычисление различных оценок качества, в том числе методом кроссвалидации. Функция вызывается 100 раз, и полученные результаты помещаются в таблицу `dr100`.

Результат выполнения кода приведён в верхней таблице на рис. 6.4.

Столбец `train` содержит эмпирический риск, `true` — истинное значение риска (оценено по выборке очень большого размера, и его можно считать точным), `test` — оценка риска по дополнительной контрольной выборке того же размера, `cv5` — оценка методом кроссвалидации при $K = 5$, `cv10` — оценка методом кроссвалидации при $K = 10$, `cv5_rep` — оценка методом кроссвалидации при $K = 5$ с тремя повторениями, `cv10_str` — оценка методом кроссвалидации при $K = 10$ со стратификацией.

Строка `mean` содержит средние значения величин. Видим, что смещение всех оценок (кроме `train`) находятся в пределах статистической погрешности, т. е. несущественны.

Строка `std` содержит среднеквадратические (стандартные) отклонения величин. Видим, что между разными вариантами кроссвалидации нет существенных отличий.

Несколько нетипичен факт, что дисперсия кроссвалидации оказалась не выше, чем у контрольной выборки. Как правило, она оказывается всё же выше (что показывает уже следующий пример). Но даже при равенстве дисперсий оценка по контрольной выборке будет точнее, так как погрешность кроссвалидации включает ещё дисперсию величины `true`.

Поскольку ожидается, что стратификация имеет больший эффект при несбалансированных классах, следующий пример использует несколько изменённую модель (уменьшен размер эллип-

са), в которой частоты классов соотносятся как 1 к 10. Поскольку точность в таких задачах вычислять бессмысленно, в качестве критерия использована логарифмическая функция потерь.

Результат приведён в нижней таблице рис. 6.4. Как видим, стратификация опять не даёт эффекта.

По результатам моделирования можно вычислить также корреляцию между величинами. Полученные значения трудно интерпретировать, однако они могут представлять интерес. В некоторых примерах корреляция между tme и cv5 оказывается даже отрицательной.

Подобные эксперименты, разумеется, не доказывают равноценность разных вариантов кроссвалидации. Зато они потенциально могут продемонстрировать их неравноценность. Автору не удалось встретить или построить примеров, когда какой-то вариант кроссвалидации (при $K \geq 5$) был бы существенно лучше других. Возможно, такие примеры удастся найти читателям.

Контрольные вопросы

1. Как построить доверительный интервал для точности классификации на основе контрольной выборки?
2. Является ли оценка методом кроссвалидации несмещённой?
3. Как строится оценка out-of-fold?
4. Как строится оценка out-of-bag?
5. Как отличаются дисперсии для оценок по кроссвалидации и по контрольной выборке?
6. В чём отличие валидационной и тестовой выборок?
7. Всегда ли оправдано разбивать выборку на обучающую, валидационную и тестовую?
8. Какие существуют варианты кроссвалидации? Какой вариант можно рекомендовать по умолчанию?
9. Какая оценка называется состоятельной?

Глава 7

Эмпирический риск

7.1 Понятие эмпирического риска

7.1.1 Основные понятия

Пусть X – пространство значений переменных, используемых для прогноза, а Y – пространство значений прогнозируемых переменных.

Предположим, что все переменные являются случайными величинами с некоторой неизвестной функцией совместного распределения $F_c(x, y)$.

Индекс c здесь — это формально введённый параметр, чтобы различать распределения.

Решающей функцией (алгоритмом классификации) называется соответствие $\lambda : X \rightarrow Y$, причём λ должна быть измеримой функцией.

Качество принятого решения оценивается заданной функцией потерь $\mathcal{L} : Y^2 \rightarrow [0, \infty)$.

Под риском будем понимать средние потери:

$$R(c, \lambda) = \mathbb{E}\mathcal{L}(y, \lambda(x)) = \int_D \mathcal{L}(y, \lambda(x)) dF_c(x, y).$$

Пусть

$$S = ((x^i, y^i) \in D \mid i = 1, \dots, N)$$

– случайная независимая выборка из распределения P_c , $S \in D^N$.

Методом построения решающих функций назовём отображение

$$Q : D^N \rightarrow \Lambda,$$

где Λ – заданный класс решающих функций.

Тогда риск будет величиной $R(c, \lambda_{Q,S})$, где $\lambda_{Q,S}$ – функция, построенная по выборке S методом Q .

Эмпирический риск определим как средние потери на обучающей выборке:

$$\tilde{R}(S, \lambda) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y^i, \lambda(x^i)).$$

7.1.2 Смещённость эмпирического риска

Чтобы понять причины смещённости эмпирического риска, рассмотрим простой пример.

Предположим, что мы производим некоторые изделия и хотим убедить клиентов, что продукция имеет очень низкую долю брака. Для этого мы берём партию из $N = 100$ изделий и проверяем её.

Предположим, что обнаружилось 5 бракованных изделий. Тогда мы должны объявить клиентам, что наша продукция имеет приблизительно 5% брака.

Однако нас это не устраивает и мы проверяем следующую партию из $N = 100$ изделий. В ней оказывается 6 бракованных изделий. Проверяем дальше, и вот, наконец, в 198-й партии все изделия оказываются качественными. Публикуем видео с процессом проверки этой партии и объявляем, что наша продукция имеет 0% брака.

Легко посчитать, что если вероятность выпуска бракованного изделия составляет 0,05, то вероятность получить партию из 100 исправных изделий составляет 0,006. При этом вероятность получить из 200 таких партий хотя бы одну, состоящую только из исправных изделий, составляет уже 0,7.

Получается, что реальная доля брака в продукции, скорее всего, и составляет 5 %, а то, что нам удалось (за 198 попыток) найти партию без брака, этому не противоречит. Таким образом, оценка, что доля брака составляет 0 %, является смещённой, т. е. систематически заниженной. Это вполне очевидно, поскольку мы специально делали «подгонку» под нужный результат.

Практически такая же по смыслу «подгонка» имеет место в машинном обучении.

Предположим, что мы строим линейный классификатор. Если бы мы выбрали разделяющую плоскость наугад, то эмпирический риск был бы несмещённой оценкой риска. Но поскольку мы из всех гиперплоскостей выбираем именно ту, которая лучше всего разобьёт обучающую выборку, происходит «подгонка», и эмпирический риск становится смещённым.

Однако пример выше мы привели не только для того, чтобы проиллюстрировать причину смещённости эмпирического риска, но и для того, чтобы показать, что даже по эмпирическому риску возможно оценивать риск.

Для этого поставим себя на место потенциального клиента, который увидел ролик с исправными 100 приборами, но при этом узнал, что для этого ролика выбрана лучшая из 200 партий изделий.

Заметим, что если вероятность брака равна 0,08, то вероятность получить 100 исправных в лучшей из 200 партий составляет менее 0,05, т. е. меньше типичного уровня значимости, используемого при проверке гипотез. Исходя из этого, гипотезу, что вероятность брака больше 0,08, мы можем отвергнуть.

Таким образом, даже на основе смещённой оценки удаётся получить содержательную оценку доверительного интервала для вероятности брака. Для этого потребовалось знать «число попыток» (число партий).

Аналогичная идея используется при получении оценок Вапника–Червоненкиса.

7.2 Число разбиений выборки

Пусть имеется N объектов (точек в пространстве). Если каждый из них мы можем отнести к любому из двух классов, то всего способов классифицировать это множество объектов (выборку) будет 2^N .

При этом число возможных разбиений, т.е. число способов разбить N точек на два подмножества равно 2^{N-1} . Это вдвое меньше числа классификаций, поскольку каждому разбиению можно приписать классы двумя способами.

Заметим, что разбиение, в котором одно из подмножеств пусто, мы также учитываем.

7.2.1 Разбиения линейными функциями

Будем говорить, что N точек в n -мерном пространстве ($N > n$) находятся в *общем положении*, если никакие $n+1$ из этих точек не лежат в одной гиперплоскости. При $N = 2$ это значит, что никакие три точки не лежат на одной прямой.

Если три точки находятся в общем положении, то их можно разбить прямыми на два подмножества всеми $2^{N-1} = 4$ способами.

Если таких точек 4, то все разбиения получить с помощью прямых не получится. Можно получить только 7 из 8 разбиений. На рис. 7.1 приведены две существенно различные конфигурации точек, в каждой из которых одно из разбиений нереализуемо. Слева невозможно отделить прямой две красные точки от двух синих, справа невозможно отделить одну красную точку от трёх синих. Все реализуемые разбиения показаны прямыми.

Если на плоскости 5 точек в общем положении, то их можно разбить прямыми 11 способами (из 16 возможных).

Если мы зададимся вопросом, а на какое максимальное число областей можно поделить плоскость прямыми, то окажется, что это число связано с числом разбиений множества точек. Так, тремя прямыми мы получаем 7 областей, а четырьмя — 11 областей (рис. 7.2).

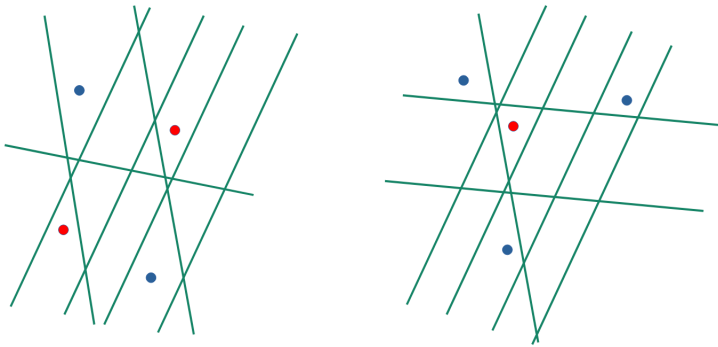


Рис. 7.1. Способы разбиения множества из 4 точек прямыми. Красные точки невозможно отделить от синих

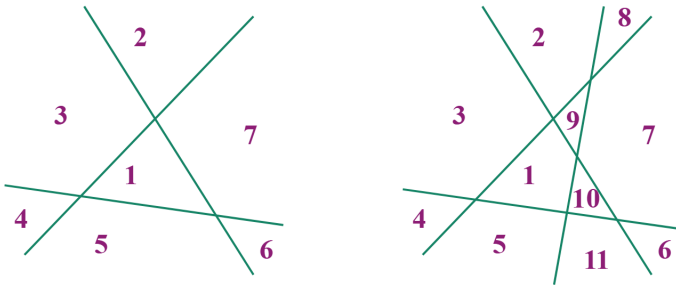


Рис. 7.2. Максимальное число областей, на которое можно поделить плоскость тремя и четырьмя прямыми

Это совпадение неслучайно. Соответствующие задачи можно назвать двойственными, и доказана их эквивалентность.

Число различных разбиений гиперплоскостями N точек в n -мерном пространстве не превосходит величины, которая выражается простой и изящной формулой

$$\mathcal{H}(n, N) = \sum_{i=0}^n C_{N-1}^i. \quad (7.1)$$

Если точки находятся в общем положении, то число их разбиений гиперплоскостями равно $\mathcal{H}(n, N)$.

Читатель может посмотреть вывод подобного выражения в [8]. При этом следует иметь в виду, что в указанном источнике рассматриваются только гиперплоскости, проходящие через начало координат. Мы же рассматриваем гиперплоскости общего вида (поэтому у нас суммирование не до $n - 1$, а до n).

7.2.2 Комбинаторная размерность

Пусть Λ – некоторый класс решающих функций для классификации на 2 класса.

Обозначим через $\hat{L}_\Lambda(N)$ максимальное число способов классифицировать выборку объёма N функциями класса Λ . Эта величина называется *функцией роста*.

Максимальное число способов приписать различные классы N объектам составляет 2^N . Поэтому $\hat{L}_\Lambda(N) \leq 2^N$. Но не всегда все эти варианты реализуемы функциями выбранного класса.

Например, 4 точки (в общем положении) в пространстве можно плоскостью разбить на 2 множества всеми способами, а 5 точек — уже не всеми.

Если точки лежат не в общем положении (например, 4 точки в пространстве оказались на одной плоскости), то число способов разбить такую выборку будет меньше.

Для гиперплоскостей в n -мерном пространстве функция роста равна $2 \cdot \mathcal{H}(n, N)$.

Ёмкостью h класса решающих функций называется максимальный размер выборки, которая может быть классифицирована этими функциями всеми 2^N способами. Таким образом, для линейных функций ёмкость равна $h = n + 1$.

Говорят, что класс имеет бесконечную ёмкость, если $\hat{L}_\Lambda(N) = 2^N$ для всех N .

При $N > h$ для функции роста в общем случае справедлива следующая оценка

$$\hat{L}_\Lambda(N) \leq \sum_{i=0}^{h-1} C_{N-1}^i. \quad (7.2)$$

Оценка (7.2) справедлива вообще для любых классов разбиений пространства на два подмножества, не только для линейных.

Ёмкость класса решающих функций также называют *комбинаторной размерностью* или (в контексте задач машинного обучения) *размерностью Валника–Червоненкиса (VC-dimension)*.

7.3 Проверка нулевой гипотезы

Под нулевой гипотезой будем понимать предположение, что классы имеют одинаковое распределение и безусловные вероятности классов равны.

При таком предположении для совершенно любой решающей функции вероятность ошибочной классификации равна 0,5.

Нулевая гипотеза является сложной гипотезой, поскольку ей удовлетворяет множество распределений.

7.3.1 Статистический критерий

Будем полагать совместное распределение в пространстве переменных (абсолютно) непрерывным (имеющим плотность).

Предположим, что на некоторой выборке линейный классификатор дал нулевой эмпирический риск. Нас интересует, можно ли на основании этого факта сделать вывод о риске, т. е. оценить вероятность ошибочной классификации.

Для начала выясним, сможем ли мы хотя бы отвергнуть нулевую гипотезу.

Обозначим получение нулевого эмпирического риска как событие A^0 .

Отвергнуть гипотезу мы сможем, если окажется, что вероятность получить нулевой эмпирический риск при справедливости нулевой гипотезы достаточно мала.

Если $N \leq n + 1$, то нулевой эмпирический риск мы получим с вероятностью 1, поскольку при любом непрерывном распределении выборочные точки с вероятностью 1 окажутся в общем положении.

Нулевую гипотезу можно представить так, что сначала в пространстве X расставляются точки, а затем каждой из них равновероятно приписывается класс y . Последнее эквивалентно тому, что равновероятно выбирается одна из 2^N классификаций.

Пусть

$$S = ((x^i, y^i) \in X \times Y \mid i = 1, \dots, N)$$

– случайная независимая выборка из некоторого распределения, удовлетворяющего нулевой гипотезе.

Обозначим

$$S_X = (x^i \in X \mid i = 1, \dots, N)$$

– последовательность точек этой выборки в пространстве X .

Обозначим через \mathcal{G}_X множество всех последовательностей из N точек пространства X , находящихся в общем положении.

Пусть $N > n + 1$ и выборочные точки оказались в общем положении, т. е. $S_X \in \mathcal{G}_X$.

Линейный классификатор может разбить эти точки $\mathcal{H}(n, N)$ способами. В результате метод выберет наилучшую из $2 \cdot \mathcal{H}(n, N)$ доступных классификаций. Если правильная классификация окажется среди доступных, то эмпирический риск будет нулевой.

Но правильная классификация в соответствии с нулевой гипотезой выбирается равновероятно из всех 2^N классификаций. Поэтому вероятность получения нулевого эмпирического риска при

заданном расположении точек S_X равна отношению числа линейных классификаций этого множества точек к общему числу классификаций, т. е.

$$P(A^0 | S_X) = \frac{2 \cdot \mathcal{H}(n, N)}{2^N}.$$

Эта вероятность одинакова при любом расположении точек в X при условии их общего положения.

Но если условная вероятность не зависит от расположения точек, то она равна безусловной вероятности, поэтому

$$P(A^0) = \frac{2 \cdot \mathcal{H}(n, N)}{2^N} = \frac{1}{2^{N-1}} \cdot \sum_{i=0}^n C_{N-1}^i. \quad (7.3)$$

Это следует из формулы полной вероятности, где с учётом $P(\mathcal{G}_X) = 1$ имеем

$$P(A^0) = \int_{X^N} P(A^0 | S_X) P(dS_X) = \int_{\mathcal{G}_X} P(A^0 | S_X) P(dS_X),$$

и константа выносится из интеграла.

Формула (7.3) позволяет точно вычислить вероятность получить линейно разделимую выборку при нулевой гипотезе. Так, в двумерном пространстве при $N = 6$ получаем вероятность 0,5, при $N = 10$ получаем 0,089844, при $N = 14$ вероятность равна 0,01123.

Эта вероятность есть p-value, т. е. достигаемый уровень значимости при проверке нулевой гипотезы. Так, если мы безошибочно классифицировали 10 точек в двумерном пространстве, то на уровне значимости 0,1 нулевую гипотезу отвергаем, а на уровне значимости 0,05 не отвергаем.

Недостатком полученного критерия является то, что он не учитывает соотношение количества объектов каждого класса в выборке. Например, если все объекты выборки окажутся одного класса, то они классифицируются заведомо без ошибок. А если объектов обоих классов окажется поровну, то при таком условии вероятность события A^0 будет меньше безусловной.

В реальных задачах мы, разумеется, знаем, сколько объектов каждого класса попало в выборку. Используя эту информацию, можно было бы получить более «сильный» критерий для проверки нулевой гипотезы. Но для этого пришлось бы делать предположения о конкретной форме распределения, и выкладки потребуются несравнимо более сложные [19].

Моделирование, проведённое в параграфе 7.6, показывает, что отличие вероятностей при неизвестном и при равном соотношении частот классов существенно, но не очень значительно (в разы, но не на порядки).

7.3.2 Неполная разделимость

Выведем формулу (7.3) несколько иными рассуждениями.

Зафиксируем точки выборки в некотором общем положении.

Имеем $L = 2 \cdot \mathcal{H}(n, N)$ различных классификаций этих точек линейными функциями.

Обозначим через A_l^0 событие, что выборка безошибочно классифицирована l -м классификатором. Тогда событие безошибочной классификации хотя бы одним из классификаторов есть

$$A^0 = \sum_{l=1}^L A_l^0.$$

Вероятность каждого из таких событий есть $P(A_l^0) = 2^{-N}$.

События несовместны (два различных на выборке классификатора не могут оба классифицировать эту выборку безошибочно), поэтому вероятность суммы есть сумма вероятностей. Откуда

$$P(A^0) = P(A^0 | S_X) = L \cdot P(A_l^0 | S_X) = \frac{L}{2^N}.$$

Получили формулу (7.3).

Но представление искомого события суммой позволяют получить критерий и в случае, когда выборка классифицируется с ошибками.

Обозначим через A_l^r событие, что l -й классификатор на выборке сделал не более r ошибок.

Вероятность такого события (при справедливости нулевой гипотезы) легко считается, а именно

$$P(A_l^r) = \frac{1}{2^N} \cdot \sum_{m=0}^r C_N^m.$$

События A_l^r уже не являются несовместными (при $r > 0$), но вероятность суммы всегда не превосходит суммы вероятностей, поэтому

$$P(A^r | S_X) \leq L \cdot P(A_l^r | S_X) = \frac{L}{2^N} \cdot \sum_{m=0}^r C_N^m.$$

Если оценка справедлива для всех условных вероятностей, то она справедлива и без условия, поэтому

$$P(A^r) \leq \frac{2 \cdot \mathcal{H}(n, N)}{2^N} \cdot \sum_{m=0}^r C_N^m. \quad (7.4)$$

Представление о степени завышенности оценки (7.4) можно получить, проведя статистическое моделирование с помощью кода из параграфа 7.6.

Выражение (7.4) можно использовать в качестве критерия проверки нулевой гипотезы, но этот критерий при $r > 0$ не будет точным (выражение будет завышать p-value).

7.4 Оценки Вапника–Червоненкиса

7.4.1 Доверительный интервал

Повторим рассуждения подраздела 7.1.2, но вместо деталей будут объекты выборки, а вместо брака — факт ошибочной классификации.

Обозначим через p вероятность ошибочной классификации, которая нам не известна и которую мы хотим оценить.

Пусть N_e – число ошибочно классифицированных объектов выборки объёма N , обозначим долю ошибок как $\nu = \frac{N_e}{N}$.

Оценим вероятность того, что ν будет отличаться от p более чем на ε .

Как известно, N_e подчиняется биномиальному распределению с параметром p . В силу центральной предельной теоремы распределение величины ν может быть приближено нормальным распределением с математическим ожиданием p и дисперсией $\frac{p(1-p)}{N}$. Имеем

$$P(|\nu - p| \geq \varepsilon) \approx 1 - \Phi\left(\varepsilon \sqrt{\frac{N}{p \cdot (1-p)}}\right) \leq 1 - \Phi(2\varepsilon\sqrt{N}),$$

где $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-z}^z e^{-\frac{t^2}{2}} dt$.

Мы воспользовались фактом, что $\max_{0 \leq p \leq 1} \sqrt{p \cdot (1-p)} = \frac{1}{2}$.

Интеграл от нормальной плотности не выражается аналитически

$$1 - \Phi(z) = 2 \frac{1}{\sqrt{2\pi}} \int_z^\infty e^{-\frac{t^2}{2}} dt,$$

но мы его можем оценить:

$$\begin{aligned} \int_z^\infty e^{-\frac{t^2}{2}} dt &= \int_0^\infty e^{-\frac{(u+z)^2}{2}} du = \int_0^\infty e^{-\frac{u^2+2uz+z^2}{2}} du \leq \\ &\leq e^{-\frac{z^2}{2}} \int_0^\infty e^{-uz} du = \frac{1}{z} e^{-\frac{z^2}{2}}. \end{aligned}$$

По ходу выкладок мы сделали замену $t = u + z$ и воспользовались тем, что $e^{-\frac{u^2}{2}} \leq 1$.

В результате при $z > 1$ имеем

$$1 - \Phi(z) \leq 2 \frac{1}{z\sqrt{2\pi}} e^{-\frac{z^2}{2}} < e^{-\frac{z^2}{2}}.$$

Если $2\varepsilon\sqrt{N} > 1$, то

$$P(|\nu - p| \geq \varepsilon) < e^{-2\varepsilon^2 N}.$$

Мы оценили для заданной решающей функции вероятность того, что эмпирический риск будет отличаться от вероятности ошибки больше чем на ε . Фактически мы получили частный случай неравенства Хёфдинга, но приведённые выкладки более наглядны и позволяют оценить степень огрубления.

Предположим теперь, что мы заранее (независимо от выборки) зафиксировали L каких-то решающих функций в пространстве X .

Оценим вероятность того, что эмпирический риск будет отличаться от вероятности ошибки больше чем на ε хотя бы для одной из этих решающих функций. Эта вероятность больше предыдущей, но не более чем в L раз, поскольку вероятность суммы событий не превосходит сумму вероятностей. Поэтому

$$P\left(\bigvee_{\lambda \in \Lambda} |\nu(\lambda) - p(\lambda)| \geq \varepsilon\right) < Le^{-2\varepsilon^2 N}.$$

Приведённые выкладки аналогичны тому как мы оценивали вероятность получения серии изделий без брака хотя бы в одной из множества партий.

Нам нужно, чтобы вероятность выхода за окрестность была достаточно маленькой. Пусть эта вероятность равна $1 - \eta$, где η – выбранная доверительная вероятность (которая равна, например, 0,95). Приравняем полученную оценку к этой вероятности. Имеем

$$Le^{-2\varepsilon^2 N} = 1 - \eta, \quad \ln L - 2\varepsilon^2 N = \ln(1 - \eta),$$

откуда

$$\varepsilon = \sqrt{\frac{\ln L - \ln(1 - \eta)}{2N}} \approx \sqrt{\frac{\ln L}{2N}}. \quad (7.5)$$

Приближение корректно, если $\ln L \gg -\ln(1 - \eta)$.

Выражение (7.5) даёт оценку величины отклонения эмпирического риска от риска.

Как и в законе больших чисел, отклонение обратно пропорционально корню из объёма выборки. Однако в числителе теперь стоит логарифм числа решающих функций.

7.4.2 Использование функции роста

Практически все актуальные классы решающих функций бесконечны. Например, мощность множества всех гиперплоскостей — континуум. И в этом случае мы не можем непосредственно применить выражение (7.5), поскольку придётся подставить в него $L = \infty$.

Чтобы применить оценки предыдущего подраздела в такой ситуации, нужно вместо количества функций взять количество классов эквивалентности. Эквивалентными мы считаем функции, которые одинаково классифицируют точки выборки. Такое понятие определено по отношению к конкретной выборке. Но существует максимальное количество классов эквивалентности по всем выборкам (заданного размера). Это количество и есть функция роста.

Интуиция подсказывает, что для бесконечных классов решающих правил можно в (7.5) вместо L просто подставить функцию роста $\hat{L}_\Lambda(N)$. Но корректность такой подмены в общем случае пока не доказана. При этом в [8] доказаны для общего случая более слабые оценки, аналогичные (7.5).

Однако в предположении нулевой гипотезы справедливость выражения (7.5) с функцией роста вместо L доказать несложно, и мы это сейчас сделаем.

Для этого, как уже делали недавно, зафиксируем некоторую конфигурацию точек S_X , оставив случайными только метки классов для выборки. Имеем

$$\begin{aligned} \mathbb{P} \left(\bigvee_{\lambda \in \Lambda} |\nu(\lambda) - p(\lambda)| \geq \varepsilon \mid S_X \right) &= \\ &= \mathbb{P} \left(\bigvee_{\lambda \in \Lambda_{S_X}} |\nu(\lambda) - 0,5| \geq \varepsilon \mid S_X \right) < \hat{L}_\Lambda(N) \cdot e^{-2\varepsilon^2 N}, \end{aligned}$$

где Λ_{S_X} — способы классифицировать точки S_X .

Полученная оценка справедлива при любом S_X , значит, она справедлива и безусловно.

Поэтому в случае нулевой гипотезы для разности между риском и эмпирическим риском справедлива оценка

$$\varepsilon = \sqrt{\frac{\ln \hat{L}_\Lambda(N) - \ln(1 - \eta)}{2N}} \approx \sqrt{\frac{\ln \hat{L}_\Lambda(N)}{2N}}. \quad (7.6)$$

Нулевая гипотеза является в некотором роде «худшим случаем», поэтому если оценка справедлива для неё, то естественно ожидать, что она справедлива и в общем случае. Скорее всего, это так, но приведённая схема доказательства не пройдёт, поскольку вне нулевой гипотезы $p(\lambda)$ будет зависеть от S_X .

При $N > h$ для функции роста в общем случае справедлива грубая оценка

$$\hat{L}_\Lambda(N) \leq 1,5 \cdot \frac{N^h}{h!}.$$

Отсюда следует, что $\ln \hat{L}_\Lambda(N)$ может быть оценен сверху величиной порядка $h \ln N$, из чего заключаем, что для классов конечной ёмкости эмпирический риск является состоятельной оценкой риска.

Приведённый результат важен с теоретической точки зрения, однако практическая значимость его ограничена.

Во-первых, методы с ограниченной ёмкостью класса решений сейчас применяются существенно реже, чем так называемые универсальные методы, в которых сложность решения подстраивается под размер выборки и может быть сколь угодно большой. Таким свойством обладают деревья решений и построенные на их основе ансамбли.

Во-вторых, близость эмпирического риска к риску сама по себе не имеет практического значения. Иными словами, нам не важно, насколько близко качество решения на обучении и на контроле (на новых объектах). На практике важно лишь само значение риска на новых объектах.

7.4.3 Неуллучшаемость оценок ☆

Данный подраздел имеет повышенную сложность и при первом чтении может быть пропущен.

Будем рассматривать случай конечного множества классификаторов.

Пусть p — вероятность «успеха» в схеме Бернулли. Для фиксированного классификатора в роли p будет выступать вероятность ошибочной классификации.

Обозначим через $\xi = \frac{N_e}{N}$ случайную величину, представляющую собой долю ошибочно классифицированных объектов обучающей выборки,

$$B(\gamma, N, p) = P(\xi \leq \gamma) = \sum_{0 \leq i \leq N\gamma} C_N^i p^i (1-p)^{N-i}$$

— кумулятивное биномиальное распределение.

Имеем: $P(\xi \leq \gamma) = B(\gamma, N, p)$ — вероятность, что доля ошибочно классифицированных объектов (эмпирический риск) будет меньше γ .

Если приравнять данную вероятность заданному уровню значимости α , то получим уравнение, связывающее p и γ . Выразив p как функцию γ , получим границу доверительного интервала для вероятности ошибочной классификации.

Пусть $\hat{p}(\gamma)$ — функция, задаваемая уравнением

$$B(\gamma, N, \hat{p}(\gamma)) = \alpha.$$

Очевидно, что для любого p выполняется $P(p > \hat{p}(\xi)) \leq \alpha$.

Рассмотрим конечное множество классификаторов Λ , $|\Lambda| = L$.

Для каждого $\lambda \in \Lambda$ определена вероятность ошибочной классификации $p(\lambda)$.

Обозначим $A(\lambda)$ — событие $p(\lambda) > \hat{p}(\gamma)$.

Имеем:

$$P(p(\lambda(\nu)) > \hat{p}(\gamma)) \leq P\left(\sum_{\lambda \in \Lambda} A(\lambda)\right) \leq \sum_{\lambda \in \Lambda} P(A(\lambda)) \leq \alpha L. \quad (7.7)$$

Таким образом, при доверительной вероятности $\eta = 1 - \alpha L$ функция $\hat{p}(\gamma)$ является доверительным интервалом для вероятности ошибочной классификации при любом методе $\lambda(\nu)$ выбора решающей функции из Λ . Получили один из вариантов оценок Вапника–Червоненкиса.

В оценке присутствуют три неравенства. На первый взгляд может показаться, что первое из них несущественно влияет на точность оценки для метода минимизации эмпирического риска по всему классу Λ , поскольку при минимизации эмпирического риска, как правило, максимизируется разность между риском и эмпирическим риском. Однако классификатор, минимизирующий эмпирический риск, не обязательно оказывается среди нарушающих доверительный интервал. Фактически это есть эффект «расслоения» [12]. Помимо «расслоения», погрешность оценки этим неравенством может быть вызвана тем, что метод классификации не всегда минимизирует эмпирический риск (использует более сложный критерий качества или приближенный метод оптимизации).

Второе неравенство (известное также как «union bound») обращается в равенство только в случае, когда вероятность произведения любой пары событий равна нулю. В случае независимых событий, вероятность которых отлична от 0 и 1, неравенство становится строгим. Однако тогда его вклад в погрешность оценок оказывается несущественным. Так, если A_j независимы (в совокупности) и $P(A_j) = p$, то

$$P\left(\sum_{j=1}^L A_j\right) = 1 - (1 - p)^L \xrightarrow[L \rightarrow \infty]{pL = \text{const}} 1 - e^{-pL}.$$

При $1 - e^{-pL} = 0,2$ получим $pL \approx 0,22$.

Случай независимых событий представляет собой пример, когда неравенство union bound не вносит существенной погрешности в оценку риска, и именно на независимых событиях строится пример, показывающий, что сложностная оценка является асимптотически точной (рассмотрен далее). Вместе с тем в большинстве практических ситуаций события $A(\lambda)$ зависимы, причем зависи-

мость такова, что погрешность union bound становится весьма значительной. Это эффект «сходства» классификаторов [12].

Заметим, что эффекты «расслоения» и «сходства» связаны между собой, в частности для независимых $A(\lambda)$ эффект «расслоения» также не имеет места (далее иллюстрируется на примере).

Последнее неравенство в оценке (7.7) есть следствие дискретности биномиального распределения и не дает существенного вклада в погрешность.

Рассмотрим простейшие примеры.

Пусть $L = 1$. Оценка риска превращается в классический односторонний доверительный интервал для вероятности успеха в схеме Бернулли. Следующий пример: $L = 2$, причем λ_2 является инверсией λ_1 , т. е. $\lambda_2(x) = 1 - \lambda_1(x)$. Можно для определенности считать, что первый классификатор всегда приписывает класс 0, второй — класс 1. Это соответствует методу классификации объектов по оценкам безусловных вероятностей классов (т. е. в соответствии с выборочными частотами классов, без использования признакового пространства X). Вероятностная модель задается единственным параметром $p = P(y = 0)$, и пусть m — количество объектов с $y = 0$ в выборке.

Оценка по формуле (7.7) совпадает с оценкой, полученной из двустороннего доверительного интервала для вероятности успеха в схеме Бернулли. Однако такой интервал неоптимален, поскольку включает в критическое множество лишние области (рис. 7.3). За счет «расслоения» часть критической области двустороннего доверительного интервала не входит в критическую область оценки риска. Это происходит, когда выход за границу двустороннего интервала случается на классификаторе, который имеет не наилучший эмпирический риск. На рис. 7.3 эта область показана двойной штриховкой (критические множества — одинарной).

Будем рассматривать асимптотическое поведение оценок (7.7), когда абсолютный объем выборки стремится к бесконечности, но при этом относительно сложности остается малым [31–33], т. е. $N \rightarrow \infty$, $\varkappa = \frac{N}{\ln L} = const$.

Хотя рассматривается асимптотический случай, полученные

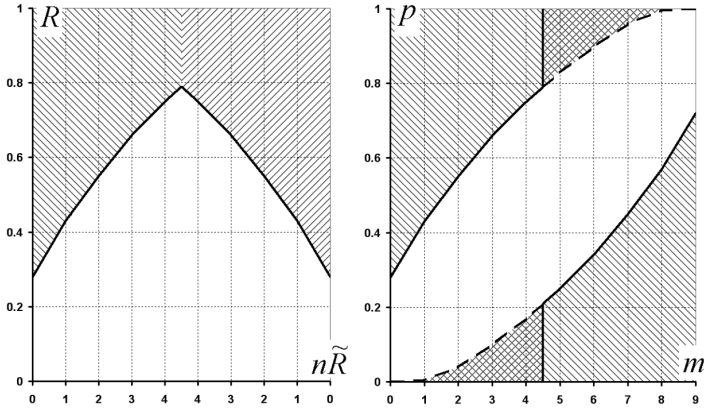


Рис. 7.3. Критические области

выводы будут на качественном уровне применимы и к случаю достаточно малых выборок $N \approx 50$. Рассмотрим простейший вариант сложностной оценки [8]: $\hat{p}(\gamma) = \gamma + \varepsilon$, где

$$\varepsilon = \sqrt{\frac{\ln L - \ln(1 - \eta)}{2N}} = \sqrt{\frac{1}{2\kappa} \left(1 + \frac{\kappa}{N} \ln(1 - \eta)\right)}.$$

При $\kappa \ll N$, что, как правило, выполняется на практике, оценка определяется главным образом значением κ , в то время как N и η не существенны. Например, при $\kappa = 5$ и $N = \infty$ имеем $\varepsilon = 0,316$, при $N = 50$ и $\eta = 0,5$ получим $\varepsilon = 0,327$, при $\eta = 0,9$ получим $\varepsilon = 0,351$.

Таким образом, при типичных значениях параметров оценка доверительного интервала слабо зависит от доверительной вероятности (если не выбирать значения, близкие к 1), а объем выборки влияет главным образом опосредованно через κ . Аналогичные рассуждения можно провести и для более тонких сложностных оценок. Это оправдывает рассмотрение предложенной асимптотики, позволяющей ограничиться наиболее существенным параметром — κ .

Чтобы на основе (7.7) построить доверительный интервал рис-

ка, нужно найти $\hat{p}(\gamma)$, удовлетворяющую уравнению

$$LB(\gamma, N, \hat{p}(\gamma)) = 1 - \eta. \quad (7.8)$$

Справедлива известная [6] оценка

$$\frac{1}{\sqrt{2\pi N\gamma(1-\gamma)}} \exp\left(-NH(\gamma, p) - \frac{1}{12N\gamma(1-\gamma)}\right) \leqslant B(\gamma, N, p) \leqslant \exp(-NH(\gamma, p)),$$

где $H(\gamma, p) = \gamma \ln \frac{\gamma}{p} + (1-\gamma) \ln \frac{1-\gamma}{1-p}$ – дивергенция Кульбака–Лейблера.

Умножая неравенства на L , получаем, что решения уравнения (7.8) при $N \rightarrow \infty$, $\varkappa = \frac{N}{\ln L}$ сходятся к решению уравнения

$$H(\gamma, \hat{p}(\gamma)) = 1/\varkappa. \quad (7.9)$$

Решение $\hat{p}_\varkappa(\gamma)$ уравнения (7.9) оказывается наилучшей (без использования дополнительной информации) асимптотической оценкой риска на основе эмпирического риска.

Для доказательства этого факта рассмотрим пример, приведенный в [24]. Пусть даны набор переменных X_1, \dots, X_L и множество классификаторов $\lambda_1, \dots, \lambda_L$, причем λ_j приписывает объекту класс, номер которого равен значению j -й переменной, т. е. $f_{\lambda_j}(x) = x_j$. Распределение в D задается следующим образом:

$$\begin{aligned} P(x, y) &= P(x|y)P(y); \quad P(x|y) = \sum_{j=1}^L P(x_j|y); \\ P(y=0) &= p_0; \quad P(x_j \neq y|y) = p, \end{aligned}$$

где p_0 и p – параметры распределения, причем выбор p_0 не имеет значения. По построению, риск для любого классификатора λ_j равен p . Вместо (7.7) для алгоритма $\lambda(\nu)$, минимизирующего эмпирический риск, в рассмотренной модели можем выписать точную вероятность выхода из доверительного интервала:

$$P(p(\lambda(\nu)) > \hat{p}(\gamma)) = 1 - (1 - B(\gamma, N, p))^L = 1 - \eta.$$

Аппроксимируя степень экспонентой, получаем:

$$LB(\gamma, N, \hat{p}(\gamma)) \approx -\ln \eta.$$

От (7.8) данное уравнение отличается только правой частью. Но так как асимптотическое решение не зависит от правой части, заключаем, что для полученного уравнения решение есть $\hat{p}_*(\gamma)$. Если взять любую $\hat{p}(\gamma)$, строго меньшую $\hat{p}_*(\gamma)$ на некотором, ненулевой длины, интервале значений аргумента, то найдутся такие параметры рассмотренной модели, что вероятность выхода из доверительного интервала $\hat{p}(\gamma)$ будет больше любого $1 - \eta < 1$. Это доказывает, что оценка $\hat{p}_*(\gamma)$ является асимптотически наилучшаемой.

То, что без дополнительной информации нельзя добиться существенного улучшения оценок Вапника–Червоненкиса, общеизвестно. В данном подразделе показано, что в рассмотренной асимптотике они не улучшаемы вовсе.

7.4.4 Точные оценки для гистограммного классификатора

Пусть $X = \{1, \dots, k\}$ – единственная дискретная переменная, и метод классификации минимизирует эмпирический риск в каждой точке.

Вероятностная мера s задается набором вероятностей

$$\rho_j = P(x = j), \quad p_j = P(y = 1 | x = j).$$

Данный пример соответствует использованию histogram classifier, когда решение принимается по гистограммам частот [7].

Практическая ценность такого классификатора, конечно, невысока, но он хотя бы реалистичен, т. е. может (пусть и с весьма посредственным результатом) применяться в реальных задачах (в отличие от примера, иллюстрирующего неулучшаемость оценок Вапника–Червоненкиса).

При этом для гистограммного классификатора удаётся построить [31] точные асимптотические оценки смещения эмпирического риска, и эти оценки дают представление о возможной величине смещения в реальных задачах.

Введём параметр $\mu = \frac{N}{k}$.

Для нахождения максимального смещения эмпирического риска нужно решить следующую оптимизационную задачу:

$$\begin{cases} E_S R(c, \lambda_S) \rightarrow \max_c, \\ E_S \tilde{R}(S) = \gamma. \end{cases} \quad (7.10)$$

Найденный максимум обозначим как $\hat{F}_\mu(\gamma)$, тогда разность $\hat{F}_\mu(\gamma) - \gamma$ есть максимальное смещение эмпирического риска.

Смысл задачи (7.10) в том, что мы ищем не просто «худшее» распределение (на котором риск максимален), а «условно худшее» распределение — то, которое даёт максимальный ожидаемый риск при заданном значении ожидаемого эмпирического риска. Это как раз отражает реальные условия, когда мы получили какой-то эмпирический риск и хотим оценить риск.

Зафиксируем параметр $\mu = \frac{N}{k}$ и устремим N и k к бесконечности.

При больших N и k распределение, доставляющее максимальное смещение эмпирического риска, имеет простую структуру: либо все ρ_j равны (между собой) и все p_j равны, либо

$$\begin{cases} p_j = 0,5, & \rho_j = a, & j < k, \\ p_k = 0, & \rho_k = 1 - (k - 1)a. \end{cases}$$

Здесь a — некоторая константа, зависящая от μ и γ .

Пример такого распределения приведён на рис. 7.4.

Сами функции максимального смещения эмпирического риска приведены на рис. 7.5. Зависимости построены с использованием численных методов, подробности можно посмотреть в [31].

В рассмотренном случае существенную роль играет эффект «сходства» классификаторов. Их множество продуцирует всего

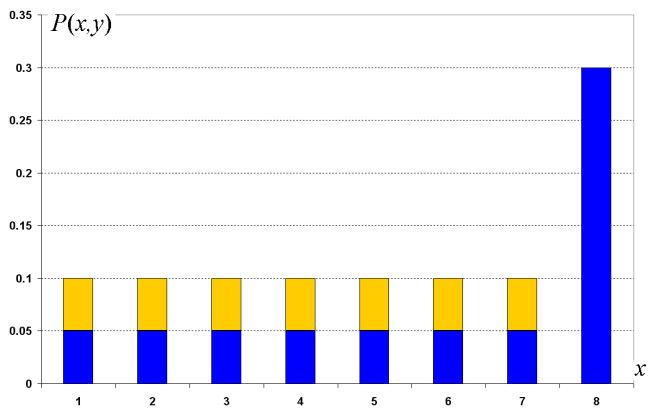


Рис. 7.4. «Условно худшее» распределение

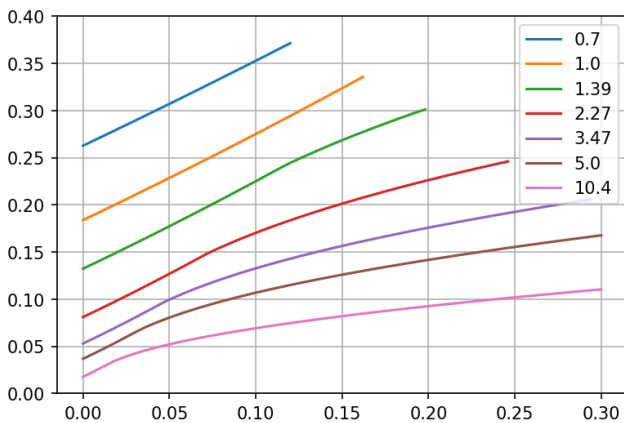


Рис. 7.5. Максимальное смещение $\hat{F}_\mu(\gamma) - \gamma$ как функция γ при различных μ

$\log_2 L = k$ «ячеек», в то время как максимально возможное число «ячеек» (имеющее место в примере, иллюстрирующем достижимость сложностных оценок) составляет 2^L . Под «ячейками» здесь понимаются области порождаемого классификаторами разбиения признакового пространства. Число таких областей представляется перспективной для дальнейшего исследования характеристикой сложности семейства классификаторов, которая может использоваться наряду с ёмкостью.

7.5 Доверительный интервал для риска

7.5.1 Качество точечных оценок

Вернёмся к вопросу о том, какая из оценок риска лучше (точнее). Для начала нужно понять, что значит, одна оценка точнее другой, и как эту точность определять.

В общем случае оценочный функционал — это некоторая функция выборки.

Качество оценок обычно характеризуют средним квадратом отклонения. Для эмпирического функционала $\bar{R}(V, Q)$ можем записать

$$\Delta = \mathbf{E} (\bar{R}(S, Q) - R(c, \lambda_{Q,S}))^2.$$

Существенная проблема заключается в том, что выражения зависят от c — распределения, которое неизвестно.

Чтобы получить гарантированную оценку точности, можно взять максимум Δ по всем c .

Но «цена погрешности» зачастую бывает разной для разных значений риска. Например, если для вероятности ошибки мы имеем оценку 0,6 при истинном значении 0,5 — это вполне несущественное отличие. Но если мы оценили вероятность ошибки как 0,1, а на самом деле она равна 0,2 — такое отклонение очень значительно. Иными словами, одно и то же отклонение при разных значениях риска имеет разную значимость.

Кажется, что нужно оценивать погрешность оценки при условии полученного значения этой оценки, например использовать

$$E_{\bar{R}}R(c, \lambda_{Q,S}).$$

Смысл данного выражения — это, например, какая ожидаемая вероятность ошибки будет у решения при условии, что на кроссвалидации получили ноль ошибок.

В случае контрольной выборки мы сумели вычислить подобное выражение, но использовали байесовский подход (который включает необоснованный произвол), и на другие функционалы этот подход не переносится.

Если же взять максимум по всем распределениям, то результат получится вырожденным. В частности, вероятность ошибки будет 0,5 независимо от значения функционала в условии.

Дело в том, что при совпадающих распределениях двух классов вероятность ошибки будет 0,5, но при этом вероятность получить ноль ошибок на кроссвалидации будет ненулевой. То, что она при этом ничтожно малая, никак не влияет на возможность вычислить условное математическое ожидание риска, которое будет, очевидно, 0,5.

Здесь естественно применить идею проверки гипотез, а именно исключить (при взятии максимума) те распределения, при которых вероятность получить наблюдаемый результат слишком мала.

В этом варианте конструкция получается осмысленной (содержательной), но слишком сложной. Раз уж мы поняли неизбежность использования идей проверки гипотез, естественно строить доверительные интервалы.

7.5.2 Доверительный интервал для риска

Доверительный интервал для R зададим в виде $[0, \hat{R}_\eta(S)]$. Мы ограничиваемся односторонними оценками, поскольку на практике для риска важны именно оценки сверху. Таким образом, в данном случае построение доверительного интервала эквивалентно выбору функции $\hat{R}_\eta(S)$, которую назовём оценочной функцией.

При этом должно выполняться условие:

$$\forall c, P_c(R \leq \hat{R}(S)) \geq \eta, \quad (7.11)$$

где η – заданная доверительная вероятность.

Строго говоря, функцию $\hat{R}(S)$ правильнее называть оценкой доверительного интервала, поскольку в классическом определении требуется, чтобы вероятность попадания оцениваемой величины в интервал должна быть равна доверительной вероятности η , а у нас в условии стоит «больше или равно». Однако в данном случае обеспечить для всех распределений равенство невозможно, поэтому доверительного интервала в таком классическом понимании не существует. Говорить об оценке того, что не существует, очевидно некорректно. Поэтому будем называть $\hat{R}(S)$ доверительным интервалом, помня о различии с классическим определением.

На практике интервальную оценку будем строить как $\hat{R}(\bar{R}(S))$, т. е. как функцию точечной оценки.

Условие (7.11) не определяет доверительный интервал однозначно, и остаётся произвол в выборе конкретного вида оценочной функции, но эта проблема решаема. Предположим, что мы выбрали какую-то функцию $\hat{R}(\bar{R}(S))$ эвристически. Вопрос, как проверить выполнение условия (7.11), которое требует перебрать все возможные распределения P_c .

Пожалуй, самый реалистичный вариант — это опять же эвристически выбрать некоторое множество семейств распределений. Так мы приходим к идее эмпирических доверительных интервалов для риска. Идея заключается в том, что если некоторое условие выполняется при всех распределениях, которые мы только смогли выдумать, то можем надеяться, что оно окажется выполненным и для реального распределения, с которым мы столкнёмся в реальной задаче.

Эмпирический доверительный интервал для R зададим в виде $[0, \hat{R}(\bar{R}(S))]$.

При этом должно выполняться условие:

$$\forall c \in \tilde{C}, P_c(R \leq \hat{R}(\bar{R}(S))) \geq \eta,$$

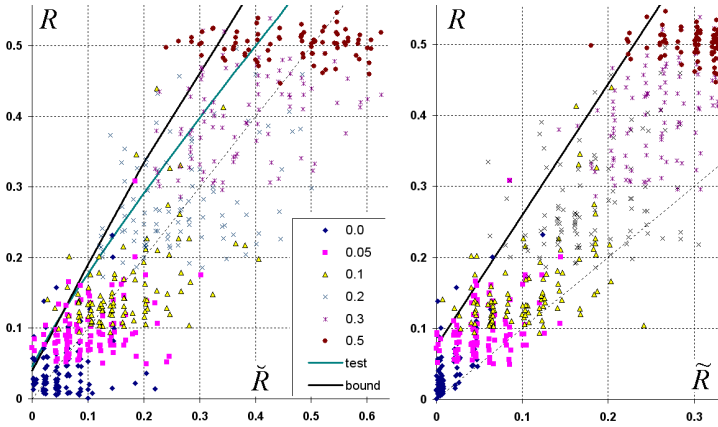


Рис. 7.6. Эмпирические доверительные интервалы на основе скользящего экзамена (слева) и эмпирического риска (справа)

где η – заданная доверительная вероятность, а \tilde{C} – эвристически выбранное множество распределений.

На рис. 7.6 приведена иллюстрация идеи построения эмпирических доверительных интервалов. Каждая точка на графике соответствует некоторой случайной выборке, по которой построена решающая функция. Для этой решающей функции вычисляются риск и его оценка через скользящий экзамен (слева) или эмпирический риск (справа). Разные цвета точек (виды маркеров) соответствуют разным распределениям.

Кривые, помеченные как «bound», проведены таким образом, чтобы выше них оказывалось бы не более 10% точек из каждого распределения. Они могут выступать в роли оценочной функции \hat{R} .

Поскольку оценки типа (7.11) подразумевают справедливость при любых распределениях, фактически они ориентированы «на худший случай». Часто это отмечают как недостаток рассмотренного подхода.

Действительно, известно много примеров, когда оценки для худшего случая оказывались далеки от типичных значений в ре-

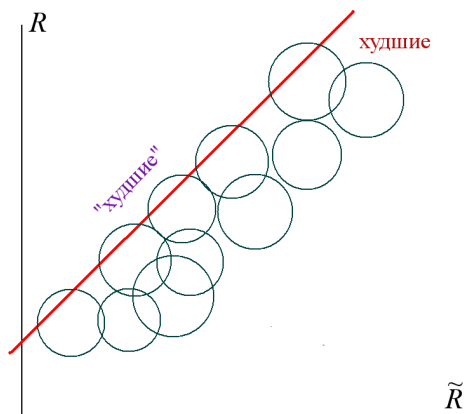


Рис. 7.7. Иллюстрация понятия «худшее распределение»

альных ситуациях. Например, симплекс-метод в худшем случае имеет экспоненциальную трудоёмкость, но в реальных задачах, как правило, находит решение быстро.

Рис. 7.7 иллюстрирует, что распределение может быть «плохим» в разных смыслах. Чаще всего плохим называют распределение, при котором любой метод даёт решение низкого качества (с большим значением риска). Это, например, распределение, в котором распределения переменных при условии каждого класса совпадают. Подобные распределения в реальных задачах действительно редко встречаются, поэтому ориентироваться на них при конструировании оценок — неразумно.

Но оценка (7.11) ориентирована на распределения, которые плохие в совсем другом смысле. Это распределения, для которых расхождение эмпирической оценки риска и риска максимально. Но значения самого риска при этом могут быть небольшими.

Подобные «плохие» распределения уже не являются чем-то экзотическим и вполне типичны для реальных данных.

В частных случаях удалось даже явно найти эти «худшие» распределения, и оказалось, что они обладают такой особенностью, что в некоторой подобласти пространства переменных клас-

сы перекрываются, а в других областях — хорошо разделимы.

Эмпирические доверительные интервалы можно строить не только по одной статистике, но и по нескольким, например одновременно используя эмпирический риск и скользящий экзамен.

7.5.3 Свойства эмпирического риска

Для оценки эмпирического риска можно отметить следующие свойства:

- ✧ простота вычисления;
- ✧ сильная смещённость;
- ✧ состоятельность при соответствующем ограничении на сложность метода (оценки Вапника–Червоненкиса);
- ✧ малая дисперсия.

Все они подробно обсуждались и не требуют дополнительных комментариев.

Следует сделать общий вывод относительно области применения эмпирического риска.

Несмотря на то, что на его основе мы не знаем способа с приемлемой точностью оценить риск (за редким исключением), вычисление эмпирического риска имеет смысл в качестве верхней оценки качества, а также для контроля процесса обучения.

При использовании методов малой ёмкости (например, линейных или наивного байесовского) и большом объёме выборки смещение эмпирического риска становится незначительным, и его можно использовать непосредственно как оценку риска.

7.6 Примеры кода

Приведём пример кода для оценки вероятности безошибочной классификации выборки при совпадающих распределениях классов.

Сначала генерируем выборку из нормального распределения. Затем каждой точке приписываем равновероятно один из двух

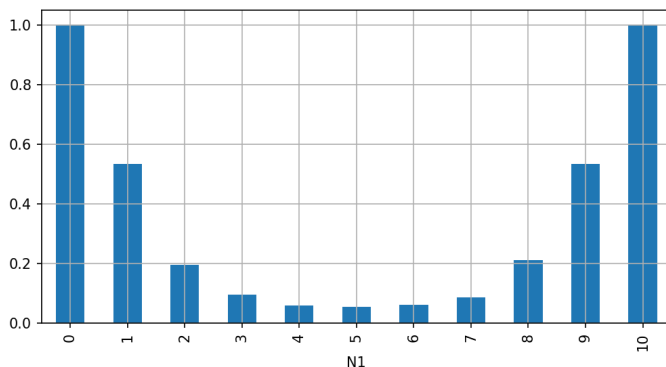


Рис. 7.8. Зависимость частоты безошибочной классификации от числа объектов первого класса

классов. Полученную выборку классифицируем методом опорных векторов (гиперпараметры нацелены на минимизацию ошибок).

Данные действия повторяются многократно и вычисляется доля выборок, которые удалось классифицировать без ошибок.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import LinearSVC
```

```
def gen_val(N,n=2):
    X = np.random.normal(size=(N,n))
    y = np.random.randint(0,2,size=N)

    ys = y.sum()
    sc = 1.0
    if ys>0 and ys<N:
        clf = LinearSVC(C=10000.0,
                        max_iter=500000,tol=1e-8).fit(X,y)
        sc = clf.score(X,y)
    return sc, int(ys)
```

```
df = pd.DataFrame((gen_val(10) for i in range(50000)),
                  columns=['score', 'N1'])
df['perfect'] = (df['score']==1).astype(int)
```

```
print(df['perfect'].mean())  
  
df.groupby('N1')['perfect'].mean().plot.bar(  
    figsize=(8,4), grid='on')
```

При запуске программы доля безошибочно классифицированных выборок составила 0,089. Это хорошо согласуется со значением вероятности 0,089844, которое получено по формуле (7.3).

При равных частотах классов в выборке частота безошибочной классификации составила 0,053.

Полная зависимость приведена на рис. 7.8.

Контрольные вопросы

1. Почему эмпирический риск является смещённой оценкой риска?
2. Свидетельствует ли значительное смещение эмпирического риска о переобучении?
3. При каком условии эмпирический риск является состоятельной оценкой риска?
4. Что такое комбинаторная размерность?
5. Что такое нулевая гипотеза применительно к задаче классификации?
6. Каковы причины завышенности оценок Вапника–Червоненкиса?
7. Какова польза от вычисления (и анализа) эмпирического риска?

Глава 8

Задача машинного обучения

В данной главе мы, наконец, попытаемся выяснить, чем же до сих пор занимались, а именно, как формально поставить задачу, которую решали. В этом мы следуем логике движения от частного к общему. До этого не было смысла делать общую формальную постановку задачи, поскольку для её понимания нужно было получить содержательное представление о задаче.

Но перед общей постановкой требуется рассмотреть ещё ряд вопросов.

8.1 Критерии качества

8.1.1 Целевая функция потерь

Под качеством моделей машинного обучения в первую очередь понимают точность прогноза, которую они обеспечивают. Однако точность можно определять разными выражениями, поэтому в каждой задаче стоит выбор подходящей меры качества (точности).

Можно ли ввести объективную меру качества? Кажется, что да.

Заметим, что прогноз нам нужен в конечном итоге для принятия каких-то решений. Например, на основе прогноза погоды принимаются решения, брать ли на улицу зонт, когда сеять и т. п. И каждое решение имеет свою объективную цену в случае, если оно окажется ошибочным. В большинстве случаев эту цену ошибки очень трудно точно измерить. Например, поставлен ошибочный диагноз. Пострадавший может подать иск о денежной компенсации, и в случае его удовлетворения это будут реальные денежные потери того, кто допустил ошибку. Но отражает ли эта компенсация реальный ущерб? И как быть, если из-за этой ошибки подавать иск уже некому?

В подавляющем большинстве случаев с приемлемой точностью оценить ущерб от ошибочного прогноза не представляется возможным. Но нужно понимать, что этот ущерб объективно существует.

Иными словами, можно считать, что во всех задачах машинного обучения существует объективная функция потерь, которую мы не знаем. Тем не менее качественное представление о величине потерь при разных исходах обычно имеется. И именно эти представления и нужно использовать при выборе функции потерь, т. е. выбирать выражение, которое хотя бы примерно соотносится с ожидаемым ущербом.

Виды критериев качества.

- ✧ Целевой критерий: функция потерь отражает реальные потери от ошибочно принятого решения. Мы не выбираем этот критерий, он — объективная данность.
- ✧ Критерий для обучения: может использовать значение целевой функции потерь на выборке, а может включать любую другую функцию потерь. Не обязательно вообще выражается через потери (дискриминант Фишера). Обычно включает регуляризацию.

Целевой критерий качества в инженерной среде принято называть *метрикой качества*. Это, конечно, неудачный термин, поскольку метрика в математике — это расстояние, т. е. имеет со-

вершено другой смысл.

На самом деле, «критерий качества» — тоже неудачный термин, поскольку в математике термин «критерий» так или иначе подразумевает предикат, т. е. логическую функцию, а не числовую.

За неимением лучшего, будем использовать данные термины, и под целевым критерием (метрикой) качества будем понимать математическое ожидание целевой функции потерь.

Критерий для обучения не обязательно совпадает с целевым. Зачастую реализации методов машинного обучения умеют оптимизировать только небольшое число функций, поэтому выбор критериев для обучения довольно узок.

В инженерной среде критерий для обучения часто называют «лоссом» (калька с английского). Во многих библиотеках параметры, задающие обучающий и целевой критерий называются «loss» и «metric» соответственно.

Однако, если в устной речи такой жаргонизм ещё приемлем, то в письменной он выглядит неуместным, поэтому термин «лосс» мы использовать не будем.

8.1.2 Критерии точности классификации

До сих пор в задаче классификации мы чаще всего упоминали индикаторную функцию потерь, усреднение которой даёт вероятность ошибочной классификации.

Противоположной величиной будет точность (ассигасу), т. е. вероятность правильной классификации.

Во многих прикладных задачах индикаторная функция потерь не подходит, поскольку объективно потери несимметричны для разных классов. Например, принять больного человека за здорового обычно более опасно, чем здорового посчитать больным.

Таким образом, нужно учесть все варианты отдельно. Имеем 4 комбинации для предсказанного и истинного класса.

Мы одинаковыми терминами будем называть как математические ожидания, так и выборочные средние.

Пусть нам нужно оценить качество решения на выборке объёма $N = N_1 + N_{-1}$.

Посчитав количество объектов для каждой комбинации предсказанного и истинного класса, получим так называемую матрицу ошибок.

True Positive (TP) False Positive (FP)
False Negative (FN) True Negative (TN)

Здесь «Positive» и «Negative» означает, что были предсказаны соответственно классы 1 и -1 . Слова «True» и «False» указывают на то, было ли предсказание ошибочным. Например, «False Negative» означает, что был предсказан класс -1 , и это оказалось неверным, т. е. истинный класс был 1.

Имеет место $N_1 = TP + FN$, $N_{-1} = FP + TN$.

По матрице ошибок мы можем вычислить отдельно относительные ошибки в каждом классе:

$$FPR = \frac{FP}{N_{-1}}, \quad FNR = \frac{FN}{N_1}.$$

Если принадлежность объекта классу -1 назвать «нулевой гипотезой», то величины «False positive rate» (FPR) и «False negative rate» (FNR) будут соответствовать ошибкам первого и второго рода. Так, ошибка первого рода (FPR) — ситуация, когда отвергнута верная нулевая гипотеза.

Величина $TPR = \frac{TP}{N_1}$ — точность предсказания относительно первого класса, её называют *recall* (полнота).

Точность предсказания относительно другого класса определяется как $TNR = \frac{TN}{N_{-1}}$ и называется *specificity*.

Если классы в задаче несбалансированы (один встречается намного чаще другого), то в качестве усреднённой характеристики точности решения можно использовать сбалансированную точность (*balanced accuracy*)

$$BA = \frac{TPR + TNR}{2}.$$

Можно подумать, что введённых характеристик достаточно на все случаи жизни, но нет.

Представим, что мы разработали метод диагностики некоторой достаточно редкой болезни, для которого ошибки первого и второго рода равны по 0,001, т. е. сбалансированная точность равна 0,999.

Кажется, что это прекрасный результат. Но предположим, что в нашем городе 1 млн человек, среди которых 100 больных. Наш метод с большой вероятностью обнаружит всех этих больных. Однако на масштабе города он также посчитает больными примерно 1000 здоровых человек.

Если мы поделим число правильно диагностированных больных на количество людей, отнесённых к больным, то получим $\frac{100}{1100}$.

Такая величина называется *precision* (Positive predictive value)

$$PPV = \frac{TP}{TP + FP}.$$

Другой типичный пример — системы автоматизированного допуска на территорию. Здесь важна не столько доля ложных распознаваний по отношению к числу попыток входа, сколько доля посторонних, которые были запущены на территорию, относительно общего числа вошедших. Это снова *precision*.

Заметим, что на русский язык термины *accuracy* и *precision* переводятся одинаково, поэтому под точностью будем понимать *accuracy*, а «*precision*» использовать без перевода.

Также часто используется так называемая F1-метрика (F1-score), которая представляет собой среднее гармоническое от *precision* и *recall*

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

У канадских эскимосов существует около 50 слов для описания воды в твёрдом агрегатном состоянии. На основе таблицы из 4 клеток было придумано более полутора десятков терминов, для выражения различных комбинаций этих чисел. Кажется, что

нет смысла пытаться их целенаправленно запоминать. И во избежание путаницы кажется предпочтительным использовать их англоязычные формы.

Также нужно понимать, что приведённые «метрики» являются скорее косвенными. Реальное качество решения определяется объективной функцией потерь. Однако поскольку объективные потери от ошибочного решения довольно трудно измерить, косвенные метрики качества актуальны и полезны.

8.1.3 Кривая ошибок

Все методы классификации позволяют получить в качестве предсказания не только номер класса, но и оценку уверенности в правильности прогноза. Во многих случаях такая оценка делается в форме условной вероятности принадлежности классу. Иногда исходная оценка не имеет вероятностной интерпретации. Например, метод SVM даёт оценку в виде отступа. В любом случае, существует некоторая величина, по которой мы можем относить объект к тому или другому классу, сравнивая эту оценку с порогом.

До сих пор мы по умолчанию предполагали, что относим объект к тому классу, для которого оценка условной вероятности максимальна. Для двух классов это эквивалентно пороговой вероятности 0,5.

Однако во многих прикладных задачах порог требуется иной. Например, если мы осуществляем контроль качества изделий, то вряд ли нас устроит порог 0,5. Скорее всего, мы будем принимать изделие, только если уверенность в его качестве будет достаточно высокой, например выше 0,99.

Изменяя порог принятия решения, мы меняем соотношение ошибок разного типа. Так, увеличивая порог отнесения объекта к первому классу, мы уменьшим ошибку первого рода за счёт увеличения ошибки второго рода.

Мы всегда можем повысить точность прогноза одного класса за счёт увеличения числа ошибок на другом классе, поэтому более

полезной информацией будет именно зависимость между данными величинами, а не их частные значения.

Если мы будем перебирать все варианты порогового значения (с некоторым шагом), то получим множество пар ошибок, которое можно изобразить в виде кривой.

Традиционно принято строить зависимость доли правильно классифицированных объектов первого класса от доли неправильно классифицированных объектов второго класса. Такую зависимость называют ROC-кривой (receiver operating characteristic, или кривая ошибок).

Можно проинтерпретировать эту кривую в терминах математической статистики. Вероятность ошибки первого рода при проверке статистических гипотез называют уровнем значимости. В таких терминах ROC-кривая отражает зависимость мощности критерия от уровня значимости.

Дадим формальное определение ROC-кривой. Кривая определена только для случая двух классов.

Пусть $\tilde{g}(x)$ – решающая функция, приписывающая каждой точке x некоторое число, которое можно интерпретировать как оценку условной вероятности первого класса в точке x либо как степень уверенности в принадлежности объекта первому классу.

Пусть $\tilde{F}_y(\cdot)$ – условная функция распределения случайной величины $1 - \tilde{g}(x)$.

Тогда ROC-кривая определяется как кривая, заданная параметрически множеством точек $(\tilde{F}_{-1}(z), \tilde{F}_1(z))$, когда z изменяется от $-\infty$ до $+\infty$, и отрезков, соединяющих последовательные точки в случае разрывов функций распределения.

Кривая может строиться как по самим распределениям, так и по выборке. Во втором случае $\tilde{F}_y(\cdot)$ будет эмпирической функцией условного распределения.

Существует простой и наглядный алгоритм построения ROC-кривой, который состоит из следующих шагов.

Запрашиваем у метода прогноз в форме вероятностей, отступа или «сырых» значений.

Упорядочиваем выборку по убыванию прогноза $\tilde{g}(x)$.

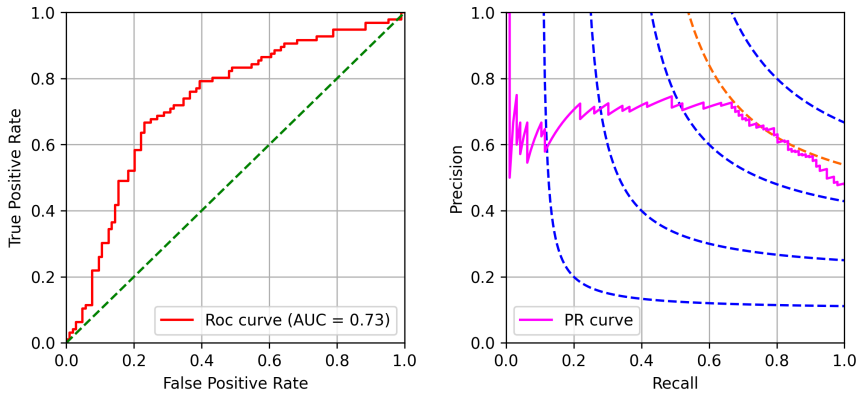


Рис. 8.1. Пример кривой ошибок (слева) и PR-кривой (справа)

Ставим «перо» в точку $(0, 0)$ и двигаемся по упорядоченной выборке. Если встречаем в выборке объект класса 1, то смещаем «перо» вверх на величину $\frac{1}{N_1}$. Если встречаем в выборке объект класса -1 , то смещаем «перо» вправо на величину $\frac{1}{N_{-1}}$. В итоге приходим в точку $(1, 1)$, нарисовав искомую кривую.

Если для каких-то точек выборки прогнозы $\tilde{g}(x)$ одинаковы, то мы не сможем упорядочить эти точки между собой. В таком случае мы на графике просто соединяем (наклонным отрезком) крайние точки участка графика, где точная траектория «пера» не определена.

Таким образом, если решающая функция задаёт строгий порядок на выборке, то ROC-кривая состоит только из прямоугольных «ступенек». Если порядок нестрогий, то появляются наклонные звенья ломаной.

Начало ROC-кривой всегда в точке $(0, 0)$, конец — в $(1, 1)$.

На рис. 8.1 слева приведён пример ROC-кривой, иллюстрирующий её типичный вид.

Распространённым критерием качества оценки $\tilde{g}(x)$ является AUC — area under the curve, т. е. площадь под ROC-кривой.

Чем больше значение AUC, тем лучше решение $\tilde{g}(x)$.

Значение 0,5 соответствует наихудшему качеству $\tilde{g}(x)$, учитывая что $AUC(1 - \tilde{g}(x)) = 1 - AUC(\tilde{g}(x))$.

Если мы получили AUC существенно меньше 0,5, то нужно просто инвертировать прогноз. Однако подобного не должно происходить, поскольку корректно применённый метод не должен «прогнозировать наоборот».

Строго монотонное преобразование функции $\tilde{g}(x)$ не меняет AUC, поэтому данная метрика хорошо подходит для задач упорядочивания объектов.

Если для построения ROC-кривой взять прогноз в виде меток классов (без вероятностей и уверенности), то ROC-кривая станет вырожденной и будет представлять собой ломаную из двух звеньев. Площадь под ней будет просто сбалансированная точность, т. е. среднее арифметическое между вероятностями правильного прогнозирования каждого класса.

Аналогично ROC-кривой можно построить PR-кривую, для которой по осям откладываются precision и recall.

Такая кривая позволит наглядно увидеть все комбинации precision и recall, которые мы можем получить за счёт изменения порога.

Свойства PR-кривой заметно отличаются от ROC-кривой.

Так площадь под PR-кривой вполне может быть меньше 0,5, и это не будет означать, что решение плохое.

На рис. 8.1 справа приведён пример PR-кривой. Дополнительно пунктирными линиями показаны линии, на которых F1-метрика принимает постоянное значение. Эти линии являются гиперболой. Оранжевая линия соответствует $F1 = 0,7$. Это максимальное значение F1-метрики, которое можно получить в приведённом решении, выбрав оптимальный порог.

8.1.4 Функция потерь для оценки вероятности

Мы рассмотрели критерии качества, которые применимы для случаев, когда прогнозируется класс либо когда требуется упорядочить объекты по вероятности принадлежности к заданному

классу. Однако во многих задачах требуется как можно точнее оценить саму вероятность.

В этом случае под решающей функцией понимается оценка $\tilde{g}(x)$ условной вероятности

$$g(x) = P_c(y = 1 | x) = \frac{P_c(dx, y = 1)}{P_c(dx)}.$$

Качество решения $\tilde{g}(x)$ можно выражать следующей функцией потерь

$$\mathcal{L}(y, \tilde{g}(x)) = -I(y = 1) \cdot \ln \tilde{g}(x) - I(y = -1) \cdot \ln(1 - \tilde{g}(x)).$$

Эта функция называется \log loss.

Выборочное среднее данной функции потерь есть взятая со знаком минус функция правдоподобия выборки по отношению к оценке условной вероятности.

Как известно, метод максимального правдоподобия — это один из мощнейших инструментов математической статистики, имеющий глубокое обоснование.

По большому счёту \log loss — единственная функция потерь, которая адекватна для оценки точности вероятностного прогноза. Причём она работает даже для несбалансированных классов.

Функция \log loss может обращаться в бесконечность. Это произойдёт, если вероятность класса будет оценена нулём, но объект окажется этого класса.

Такое свойство «метрики» является не недостатком, а достоинством. Дело в том, что в подавляющем большинстве задач оценивать вероятность нулём или единицей совершенно недопустимо. Исключение возникает, только если имеется достоверная априорная информация (о невозможности для объектов каких-то классов иметь некоторые значения каких-то признаков).

Если вся информация в задаче — это таблица данных, то 0 и 1 — это «запрещённые» значения для оценок вероятности. Поэтому бесконечные значения потерь свидетельствуют не о неадекватности метрики, а о некорректности решения, т. е. построенной модели.

Технически для вероятности (поскольку это число) можно применить даже средний квадрат отклонения, но в этом мало смысла (если речь о целевой функции потерь). Однако в качестве функции потерь для обучения средний квадрат отклонения в задаче классификации всё же используется.

Мы рассмотрели наиболее распространённые критерии качества классификации: точность (accuracy), balanced accuracy, F1 score, ROC-AUC, log loss.

8.1.5 Функция потерь для непрерывной целевой переменной

Казалось бы, переменная с континуальным множеством значений должна давать не меньшее, а то и большее разнообразие постановок задач по сравнению с категориальной переменной. Однако на практике всё наоборот: задачи с непрерывной целевой переменной в некотором смысле беднее задач классификации (именно поэтому мы почти всё внимание уделяем последним).

Основная причина в том, что в задачах классификации почти всегда нужны оценки вероятности, а для непрерывной целевой переменной прогнозируется только её значение.

Теоретически можно прогнозировать условное распределение. Но на практике такие постановки почти не встречаются и мало исследуются.

Обычно используют среднеквадратичное или среднее абсолютное отклонение. В первом случае оптимальный прогноз — функция регрессии (условное математическое ожидание), во втором — условная медиана.

Существуют задачи с несимметричной функцией потерь.

Характерный пример: прогнозирование потребительского спроса с целью формирования товарного запаса. Если реальный спрос окажется больше прогнозируемого, то часть клиентов не получит товар (поскольку запас товара мы формируем в соответствии с прогнозом), что приведёт к недополучению прибыли (и, возможно, к снижению лояльности клиентов). Если фактиче-

ский спрос окажется меньше прогноза, то часть товара останется нераспроданной, что может привести, например, к его списанию, т. е. к прямым убыткам.

8.2 Теория статистической устойчивости

8.2.1 Разложение ошибки

Зачастую при обучении моделей возникает необходимость проанализировать зависимость точности решения от сложности модели. При этом возникает желание разложить ошибку (риск) на сумму каких-нибудь компонент, поведение которых было бы максимально простым.

Существует (как минимум) два подхода к разложению риска для решающих функций.

- ✧ Разложение на смещение и разброс (bias-variance decomposition).
- ✧ Разложение на меру адекватности и меру статистической устойчивости [27].

Первый подход обсудим позднее.

Второй подход к разложению ошибки мы уже фактически обсуждали, когда сравнивали линейный и квадратичный дискриминант. Тогда мы говорили, что при неравных матрицах ковариации линейная разделяющая функция будет иметь неустранимую погрешность округления, но это может компенсироваться меньшей статистической погрешностью при оценивании параметров модели, поскольку линейная модель имеет меньше параметров, чем квадратичная.

Фактически в этом рассуждении мы разложили ошибку на компоненты: погрешность аппроксимации и статистическую погрешность.

Данные компоненты зависят от сложности монотонно: погрешность аппроксимации монотонно невозрастающая, статистическая погрешность монотонно неубывающая. Это поведение проиллю-

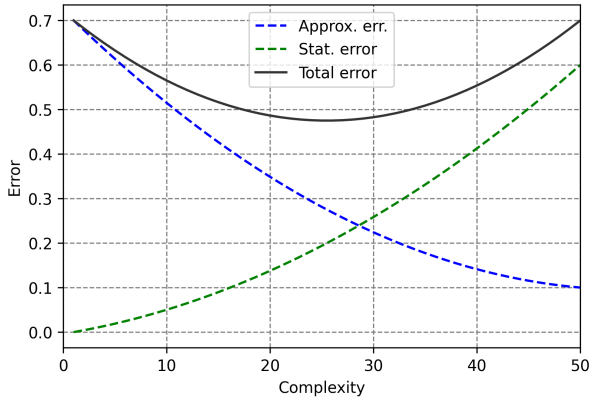


Рис. 8.2. Компоненты ошибки

стрировано на рис. 8.2.

Строго говоря, предыдущее утверждение нужно доказывать. Но мы поступим хитрее, а именно: определим понятие сложности так, чтобы первая часть утверждения выполнялась автоматически.

Теперь первая часть утверждения верна по определению. Вторую часть утверждения, вероятно, можно доказать, но пока доказательство неизвестно. Примем её как эмпирический факт.

8.2.2 Меры адекватности и устойчивости

Введём формальные определения компонент разложения.

Идея подхода [27] в том, чтобы разложить ошибку на погрешность аппроксимации и статистическую погрешность.

Напомним, что байесовским уровнем ошибки называется величина риска для оптимальной решающей функции. Другое название для этой величины — «шум», или неустранимая погрешность.

Если мы зафиксируем сложность модели, то величина риска будет уменьшаться с ростом объема выборки, стремясь к некоторому пределу.

Определим величину асимптотического риска

$$R_Q^* = \lim_{N \rightarrow \infty} \mathbb{E} \mathcal{L}(y, \lambda_{Q, S_N}(x)).$$

Эту же величину можно определить, не устремляя объём выборки к бесконечности, а применив метод построения решающей функции непосредственно к распределениям.

Все методы, которые мы рассматривали, выбирают решающую функцию в соответствии с некоторым критерием качества, который вычисляется по обучающей выборке. Но все эти критерии могут быть вычислены и на распределении, поэтому все рассмотренные методы могут быть «запущены» на распределениях. Однако такое действие требует дополнительных усилий, поскольку имеющиеся программные реализации на вход принимают всё же только выборку. Один из приёмов обхода этого ограничения был рассмотрен в примерах к бустингу.

Не всегда способ применения метода к распределениям очевиден. Например, чтобы применить к распределениям метод опорных векторов, его нужно сформулировать как задачу безусловной оптимизации.

Мерой адекватности [27] метода называется разность между асимптотическим риском и байесовским уровнем ошибки (шумом).

Мерой статистической устойчивости метода называется разность между средним (по выборкам заданного размера) риском и асимптотическим риском.

Меру адекватности будем называть погрешностью аппроксимации, а меру статистической устойчивости — статистической погрешностью.

Таким образом, мы определили разложение ошибки (риска) на шум, погрешность аппроксимации и статистическую погрешность.

8.2.3 Сложность модели

Класс решающих функций называется универсальным, если функциями этого класса возможно достичь риска, сколь угодно близкого к байесовскому. Иными словами, универсальный класс может обеспечить сколь угодно малую погрешность аппроксимации.

Примеры универсальных классов: решающие деревья, нейросети.

Определение 1 *Мерой сложности метода будем называть любой его гиперпараметр, при увеличении которого погрешность аппроксимации монотонно убывает (или, как минимум, не возрастает).*

Такому определению удовлетворяют множество гиперпараметров. Например, для бустинга основные параметры сложности — это количество деревьев и их глубина (или иной параметр, ограничивающий размер).

Возникает проблема со сравнением сложности моделей, ограниченных разными параметрами: например, непонятно, как сравнить модель из 1000 деревьев глубины 2 с моделью из 100 деревьев глубины 4.

Требуется придумать универсальные меры сложности. Одной из таких мер является ёмкость, но это характеристика класса решающих функций, а не метода.

В качестве универсальной меры сложности решения можно использовать отступ (ненормированный).

Изначально понятие отступа возникло для метода SVM. Там отступ означает попросту расстояние от точки до границы чужого класса.

После понятие отступа было перенесено на ансамбли деревьев, для которых отступ определяется как $y\beta(x)$, где $\beta(x)$ — это собственно агрегированный прогноз. В документации к sklearn величина $\beta(x)$ называется «raw values predicted» и возвращается методом `decision_function`. Мы будем называть её «сырой прогноз».

Заметим, что такой отступ (для ансамблей) уже не имеет наглядного геометрического смысла (как расстояние до чужого класса). Его можно интерпретировать, пожалуй, как степень уверенности (в неограниченной шкале) модели в ответе.

Для методов, в которых понятие отступа не возникает естественным образом, его можно определить искусственно, выразив через прогнозируемые вероятности. Для этого естественно использовать функцию, обратную к сигмоиду.

Иногда проводят нормировку отступа, разделив его на сумму весов всех компонент ансамбля. Нормированный отступ лежит в интервале $[0, 1]$ и фигурирует в оценках обобщающей способности. Эти оценки, однако, как и оценки Вапника–Червоненкиса, имеют скорее теоретическую, чем практическую ценность.

Ненормированный отступ характеризует степень обучения.

В качестве универсальной меры сложности решения также можно использовать величину, обратную эмпирическому риску.

В случае индикаторной функции потерь такая величина не очень удобна, поскольку эмпирический риск может обратиться в ноль, однако для логарифмических потерь вполне годится.

8.2.4 Процесс обучения

Процессом обучения будем называть подгонку модели под обучающую выборку.

В некоторых методах этот процесс можно прерывать досрочно. Типичный пример — нейросети.

С одной стороны, сложность нейросети однозначно определяется её архитектурой. С другой стороны, при досрочном останове параметры не успеют принять оптимальные значения. Это значит, что потенциальная сложность решения не будет реализована.

В процесс обучения мы будем включать также подбор сложности модели (в рамках одного метода).

Кривая обучения — зависимость риска от сложности модели. На практике кривая обучения строится с использованием оценки риска по тестовой выборке или кроссвалидации.

Типичный вид кривой обучения показан на рис. 8.2.

Как правило, с ростом сложности модели риск сначала убывает, затем возрастает. Участок убывания риска соответствует недообучению, участок возрастания — переобучению. Иными словами, недообучение — это использование модели недостаточной сложности, а переобучение — модель избыточной сложности.

В некоторых источниках переобучением называют ситуацию, когда качество на обучении и контроле сильно различается. Однако смысл такого способа определения непонятен, поскольку нас интересует непосредственно значение риска, а разность между риском и эмпирическим риском ни о чём не говорит. Большая разность — это не обязательно плохо и не обязательно переобучение.

В некоторых случаях переобучение вообще не происходит, т. е. риск не начинает расти, а выходит на константу.

В любом случае можно сформулировать правило, что следует выбирать такое значение сложности, после которого риск перестаёт существенно уменьшаться.

В большинстве экспериментов риск имеет только один локальный минимум, т. е. если риск начал расти, то при дальнейшем увеличении сложности он уменьшиться уже не может.

Существуют свидетельства о наблюдении двух минимумов у кривой обучения [3]. Однако есть основания усомниться в выводах этой работы. Например, в работе [3] есть график зависимости риска от сложности для случайного леса. При этом на данном графике сначала растёт глубина дерева, а затем глубина фиксируется, и растёт количество деревьев. Фактически здесь просто соединяются два разных графика, и по одной оси последовательно перебираются два разных параметра. Поэтому говорить о двух минимумах не имеет смысла, так как это минимумы разных функций, искусственно «сшитых» в одну.

Кроме того, число деревьев в случайном лесе вообще не является параметром сложности (по нашему определению), поскольку не влияет на аппроксимирующую способность модели.

На других графиках в этой работе параметром сложности

принимается число параметров нейросети. Но сложность модели определяется не только числом обучаемых параметров, но и алгоритмом обучения, а также регуляризаторами. Вполне вероятно, что неунимодальность риска в указанных примерах обусловлена как раз влиянием этих факторов.

Таким образом, возможная неунимодальность кривой обучения — это безусловно интересный феномен, заслуживающий дальнейшего изучения. Однако, скорее всего, множественность экстремумов можно устранить за счёт более корректного выбора характеристики сложности.

В целом, можно выдвинуть гипотезу (и автор не встречал убедительных примеров, её опровергающих), что риск как многомерная функция всех параметров сложности имеет один локальный минимум на всём пространстве этих параметров. Из этого, в частности, следует, что оптимизация параметров сложности — относительно простая в вычислительном плане процедура, и для неё достаточно будет, например, алгоритма покоординатного спуска. А, например, простой перебор по сетке (grid search) здесь катастрофически неэффективен.

Помимо кривой обучения имеет смысл строить зависимость эмпирического риска. Эта кривая не позволит оценить реальный риск, но она помогает контролировать процесс. Например, увидеть, не достигнут ли минимум эмпирического риска, и происходит ли вообще обучение (обучение может «застопориться», например, из-за чрезмерной регуляризации). Нужно строить график для эмпирического риска также, чтобы убедиться, что обучение не пошло «вспять» (часто происходит с нейросетями).

Иногда можно обучаться и после достижения нулевой ошибки (на обучающей выборке).

Некоторая трудность в подборе параметров возникает, если процесс обучения немонотонен. Типичный пример — обучение нейросети, когда с ростом числа эпох эмпирический риск не убывает монотонно, а демонстрирует значительные колебания.

В такой ситуации, если мы подберём число эпох по кроссвалидации, а потом с этим числом эпох обучим модель на всей вы-

борке, то можем попасть как раз в «провал» по точности (когда оптимизатор пытается преодолеть локальный минимум).

Таким образом, число эпох — это не вполне подходящий параметр для управления сложностью. Поэтому для нейросети в качестве параметра сложности лучше использовать, например, эмпирический риск (взятый с обратным знаком).

Также параметром сложности для нейросети может служить отступ.

Кстати отступ (для некоторого объекта), взятый со знаком «минус», — это вполне разумная функция потерь. А значит, средний отступ (со знаком минус) на обучающей выборке — это разновидность эмпирического риска.

8.3 Смещение и разброс

8.3.1 Bias-variance decomposition

Данное разложение является адаптацией к задаче машинного обучения используемого с давних времён разложения ошибки на систематическую и «случайную».

Вывод разложения элементарен.

Для любых (имеющих требуемые моменты) независимых случайных величин u и v имеет место

$$E(u - v)^2 = Du + (Eu - Ev)^2 + Dv.$$

Зафиксируем точку x признакового пространства и подставим $u = y | x$, $v = \lambda_{S,Q}(x)$. Здесь $y | x$ обозначает случайную величину, соответствующую условному распределению y в точке x . Получаем

$$E_{S,y|x} (y - \lambda_{S,Q}(x))^2 = D_{y|x}y + (E_{y|x}y - E_S \lambda_{S,Q}(x))^2 + D_S \lambda_{S,Q}(x).$$

Усреднение ведётся по условному распределению на целевой переменной и по выборкам заданного объёма.

В полученном разложении слагаемые принято называть соответственно: шум, смещение и разброс.

Разложение справедливо для каждой точки x . Можно провести усреднение по всему пространству, взяв математическое ожидание от обеих частей. Средний шум — то же самое, что байесовский уровень ошибки.

8.3.2 Универсальное разложение

Термин «variance» на самом деле переводится как «дисперсия». Однако мы будем использовать термин «разброс», и этому есть две причины.

Первая состоит в том, что «дисперсия» — слишком общее понятие, и требуется постоянно уточнять, о какой дисперсии идёт речь. А термин «разброс» сразу конкретизирует контекст.

Вторая, и главная, причина в том, что подобное разложение может быть реализовано [16] не только для квадратичной функции потерь, и тогда последняя компонента уже не будет дисперсией.

Итак, пусть $\mathcal{L}(\cdot, \cdot)$ — произвольная функция потерь.

Оптимальный прогноз в точке x есть

$$\bar{\lambda}(x) = \min_{y'} \mathbf{E}_{y|x} \mathcal{L}(y, y').$$

Компоненту шума, или байесовский уровень риска, определим как

$$\mathbf{E}_{x,y} \mathcal{L}(y, \bar{\lambda}(x)).$$

Теперь посмотрим на потери «с другой стороны». Пусть у нас есть какой-то случайный прогноз, и мы найдём такое значение целевой переменной, при котором этот прогноз бы имел минимальную ошибку.

Определим функцию (в литературе она называется «главное предсказание»)

$$\bar{y}(x) = \min_{y'} \mathbf{E}_S \mathcal{L}(y', \lambda_{S,Q}(x)).$$

Компоненту разброса определим как

$$\mathbf{E}_{x,S} \mathcal{L}(\bar{y}(x), \lambda_{S,Q}(x)).$$

Компоненту смещения естественно было бы определить как

$$E_x \mathcal{L}(\bar{y}(x), \bar{\lambda}(x)).$$

Однако не для любых функций потерь определённые таким образом компоненты в сумме дадут риск. Легко проверить, что «баланс не сойдётся», например, для модуля отклонения (MAE).

Чтобы имело место именно разложение риска, можно компоненту смещения «принудительно» определить как разность между риском и суммой смещения и разброса. При таком определении смещение — это «то, что осталось».

Данный подход позволяет ввести разложение риска для любой функции потерь.

8.3.3 Логарифмическая функция потерь

Для задачи классификации наиболее содержательный вид разложение на смещение и разброс принимает в случае логарифмической функции потерь. Соответствующее разложение предложено в работе [42]. Воспроизведём его, несколько упростив выкладки (их можно пропустить, поскольку они не влияют на понимание остального материала).

Для двух классов логарифмическая функция потерь выглядит как

$$L(y, u_1) = -y \ln u_1 - (1 - y) \ln(1 - u_1),$$

где u_1 — решение в виде оценки вероятности события $y = 1$, т. е. вероятности принадлежности объекта первому классу.

Для случая нескольких классов, когда $Y = \{1, \dots, k\}$, решение представляет собой вектор $u = (u_1, \dots, u_k)$, $\sum_{\omega=1}^k u_\omega = 1$. Тогда логарифмическая функция потерь примет вид

$$L(y, u) = -\ln u_y,$$

где u_y — компонента решения, соответствующая номеру класса объекта.

В таком виде функция потерь не подходит для разложения, поскольку вместо y в неё нельзя подставить распределение. Однако логарифмическую функцию потерь можно записать в виде дивергенции Кульбака–Лейблера (будем обозначать как $K(\cdot, \cdot)$).

Поскольку дивергенция определена только для распределений, нам нужно переменной Y сопоставить распределение. В роли последнего может использоваться двоичный вектор. Для этого введём случайные величины (v_1, \dots, v_k) , где v_ω является индикаторной функцией того, что $y = \omega$.

Получаем

$$L(y, u) = - \sum_{\omega=1}^k v_\omega \ln u_\omega = \sum_{\omega=1}^k v_\omega \ln \frac{v_\omega}{u_\omega} = K(v, u).$$

Мы полагаем $0 \ln 0 = 0$, поэтому $v_\omega \ln v_\omega \equiv 0$.

В силу независимости v и u имеем

$$\begin{aligned} \mathbb{E}_{y,u} L(y, u) &= - \sum_{\omega=1}^k \mathbb{E}_y v_\omega \mathbb{E}_u \ln u_\omega = \\ &= - \sum_{\omega=1}^k p_\omega \mathbb{E}_u \ln u_\omega = \mathbb{E}_u K(p, u) + H(p), \end{aligned}$$

где $p_\omega = \mathbb{E}_y v_\omega$ – вероятность того, что $y = \omega$, а

$$H(p) = - \sum_{\omega=1}^k p_\omega \ln p_\omega.$$

Введём вектор $\bar{u} = (\bar{u}_1, \dots, \bar{u}_k)$, компоненты которого вычисляются как

$$\bar{u}_\omega = \frac{1}{C} e^{\mathbb{E}_u \ln u_\omega}, \quad C = \sum_{\omega=1}^k e^{\mathbb{E}_u \ln u_\omega},$$

где C – нормировочная константа, которая обеспечивает выполнение условия $\sum_{\omega=1}^k \bar{u}_\omega = 1$. Фактически \bar{u} – это просто нормированное среднее геометрическое вектора u .

Логарифмируя выражение, получаем

$$\ln C = \mathbb{E}_u \ln u_\omega - \ln \bar{u}_\omega.$$

Умножим обе части равенства на \bar{u}_ω и просуммируем по ω .
Получим

$$\ln C = \sum_{\omega=1}^k \bar{u}_\omega (\mathbb{E}_u \ln u_\omega - \ln \bar{u}_\omega) = -\mathbb{E}_u K(\bar{u}, u).$$

Аналогично умножая на p_ω и суммируя, имеем

$$\ln C = \sum_{\omega=1}^k p_\omega (\mathbb{E}_u \ln u_\omega - \ln \bar{u}_\omega) = K(p, \bar{u}) - \mathbb{E}_u K(p, u).$$

Приравнивая полученные выражения для $\ln C$, получаем искомое разложение

$$\mathbb{E}_u K(p, u) = K(p, \bar{u}) + \mathbb{E}_u K(\bar{u}, u).$$

Окончательно

$$\mathbb{E}_{y,u} L(y, u) = H(p) + K(p, \bar{u}) + \mathbb{E}_u K(\bar{u}, u).$$

Это разложение на шум, смещение и разброс.

8.3.4 Разложение для kNN

Вернёмся к квадратичной функции потерь.

Рассмотрим очень важный частный пример, на котором можно проанализировать некоторые свойства разложения bias-variance, а именно применение метода kNN к задаче регрессионного анализа.

Обычно ожидается, что с ростом сложности модели смещение уменьшается, а разброс увеличивается. Сейчас мы покажем, что это не всегда так.

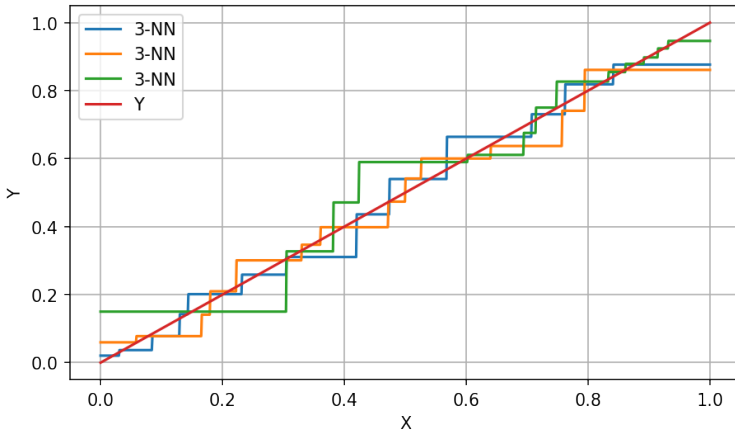


Рис. 8.3. Решения kNN для одномерной регрессии

В ряде источников (например, [41]) приводится следующая формула разложения для метода kNN

$$E(y - f(x))^2 | x = \left(f(x) - \frac{1}{k} \sum_{i=1}^k f(N_i(x)) \right)^2 + \frac{\sigma^2}{k} + \sigma^2. \quad (8.1)$$

В классической постановке задачи регрессии, т.е. когда координаты X в обучающей выборке фиксированы (не случайны), первое слагаемое в (8.1) есть bias.

В общей постановке, когда X случайны, первое слагаемое содержит как смещение, так и разброс.

Выясним, какой вид примет (8.1) в постановке со случайными «объясняющими» переменными.

На рис. 8.3 приведены примеры решений методом kNN в задаче одномерной регрессии. Здесь Y – истинная регрессионная функция, а три ступенчатые кривые – решения методом kNN, полученные для разных обучающих выборок.

Легко заметить, что средний прогноз почти совпадает с Y . Смещение существенно только на краях области и им можно вообще пренебречь.

Для данной задачи риск можно вычислить аналитически. Без учёта краевых эффектов имеем разложение [36]

$$E(y(x) - f(x))^2 \approx \frac{(k+1)(k+2)}{12N^2k} + \frac{\sigma^2}{k} + \sigma^2. \quad (8.2)$$

В отличие от (8.1) здесь и первое, и второе слагаемые — оба variance, причём первое слагаемое практически линейно растёт с ростом k .

Однако чем больше k , тем меньше аппроксимирующая способность метода kNN, а значит, параметр сложности модели для kNN должен вводиться как величина, обратная к k .

Получается, что в рассмотренном примере с ростом сложности разброс не растёт, а уменьшается.

Данный пример показывает, что с ростом сложности модели variance может не только расти (как это обычно ожидают), но и уменьшаться. Примечательно то, что этот факт установлен аналитически.

Путём статистического моделирования можно обнаружить и другие случаи, когда поведение компонент разложения отличается от «канонического».

8.3.5 Свойства разложения

Отметим важные факты:

- ✧ смещение \neq ошибка аппроксимации;
- ✧ разброс \neq ошибка статистического оценивания.

Эти факты стоит подчеркнуть, поскольку зачастую упомянутые понятия как раз смешиваются, чего делать нельзя (хотя часто они действительно довольно близки).

В литературе часто упоминается так называемый bias-variance tradeoff. Слово «tradeoff» означает такое отношение между величинами, когда увеличение одной ведёт к уменьшению другой.

Заметим, что между смещением и разбросом такое отношение выполняется не вполне строго.

Строгий tradeoff имеет место между аппроксимирующей способностью и статистической устойчивостью, т. е. между компонентами разложения, которое мы рассмотрели первым.

Результаты исследований (проведённых численно и аналитически) показывают, что в ряде примеров как смещение, так и разброс могут вести себя немонотонно при увеличении сложности (сначала уменьшаться, а затем расти).

Вследствие этого разложение на смещение и разброс не даёт полноценного объяснения свойств кривой обучения, поскольку смещение и разброс лишь косвенно характеризуют аппроксимационную способность и статистическую устойчивость метода построения решающих функций.

Пожалуй, наиболее известное применение разложение bias-variance состоит в обосновании метода случайного леса.

Легко показать, что для данного метода смещение ансамбля равно смещению каждого дерева и не зависит от числа деревьев. При этом разброс с ростом числа деревьев уменьшается. Поэтому риск для случайного леса с ростом числа деревьев строго уменьшается (конечно, не до нуля, а быстро сходится к некоторому пределу).

При этом с ростом числа деревьев T разброс уменьшается вовсе не в T раз. В T раз он бы уменьшался, если бы деревья были некоррелированными. Но деревья случайного леса зависимы, поскольку построены по пересекающимся подвыборкам, и нет оснований ожидать от них некоррелированности. А то, что они строятся независимо друг от друга означает лишь условную независимость (при фиксированной выборке). Поэтому на практике даже при неограниченном росте числа деревьев разброс может уменьшиться, хорошо если в два раза по итогу. Однако два раза — это уже хорошо, и ансамбль действительно повышает точность.

Но здесь нужно понимать важную тонкость. Разложение bias-variance позволяет доказать, что ансамбль деревьев случайного леса лучше, чем одно дерево, построенное алгоритмом случайного леса. Но из этого не следует, что случайный лес должен быть точнее, чем одно дерево, построенное «обычным» алгоритмом для

деревьев. Дело в том, что «обычный» алгоритм строит дерево по всей выборке, а в случайном лесе каждое дерево строится только на части данных.

Таким образом, случайный лес использует деревья, имеющие более низкую точность, чем дерево, построенное по всей выборке. На практике эта потеря точности с лихвой компенсируется за счёт ансамблирования. Но как это доказать — непонятно. Можно сказать, что разложение объясняет, но не доказывает эффективность случайного леса.

В заключение заметим, что оба рассмотренных разложения могут быть построены только на распределениях. Их компоненты невозможно оценить, имея лишь обучающую выборку. Поэтому разложения актуальны для теоретических исследований, но почти «неприменимы» в реальных задачах.

8.4 Общая постановка задачи машинного обучения

8.4.1 Модели

В машинном обучении очень широко встречается термин «модель». Обычно он используется как синоним решающей функции, либо под моделью понимают класс решающих функций.

Мы тоже будем использовать это значение термина, но также будем говорить о вероятностных моделях.

Под вероятностной моделью будем понимать, по сути, просто распределение, из которого получены данные как случайная независимая выборка.

Такая вероятностная модель является полной описательной моделью. Она позволяет смоделировать задачу, т. е. сгенерировать любой объём данных. Также по полной вероятностной модели всегда можно принять оптимальное решение (минимизирующее ожидаемые потери).

Есть описательные модели, а есть модели принятия решения. Вторые всегда неполные.

Фактически, если известны распределения, то нет никакой проблемы машинного обучения, есть разве что чисто техническая проблема превращения полного знания в оптимальное решение (например, путём использования формулы Байеса для нахождения условной вероятности класса в заданной точке). Тем не менее мы можем тестировать прогностические модели, «обучая» их на распределениях, оценивая тем самым их аппроксимирующую способность.

По полной описательной модели (вероятностной) можно принять оптимальное решение. Но можно строить решение и без описательной модели. Достаточно наличия объективной функции потерь и обучающей выборки.

8.4.2 Замечания о терминах

Часто задачу машинного обучения формулируют как задачу поиска скрытой зависимости.

Но объективно никакой зависимости нет, есть только совместное распределение.

Есть оптимальное решение, но оно определяется критерием. Критерий в задаче первичен, зависимость — вторична.

Задача же определяется критерием качества. Например, для квадратичной функции потерь оптимальным решением будет регрессия, а для абсолютных потерь — условная медиана.

В случае категориальной целевой переменной задачу называем задачей классификации. Для неё существует также устаревший термин «распознавание образов».

Варианты названия задачи для вещественной целевой переменной:

- ✧ регрессионный анализ — не всегда строим регрессию;
- ✧ восстановление зависимостей — объективной зависимости в общей постановке нет;
- ✧ построение решающих функций — самый точный термин.

Часто статистические подходы к машинному обучению назы-

вают байесовской теорией. Это крайне неудачная терминология. Настоящий байесовский подход подразумевает задание априорных вероятностей на гипотезах.

Вероятностная интерпретация задач машинного обучения вовсе не подразумевает использование байесовского подхода. И она никак не ограничивает исследователя в выборе решаемых задач, т. е. может применяться в любых задачах машинного обучения (хотя есть задачи, где это нецелесообразно).

8.4.3 Задача построения решающей функции

Приведём максимально общую формальную постановку задачи.

Пусть X – пространство значений переменных, используемых для прогноза, Y – пространство значений прогнозируемых переменных, \mathcal{C} – множество всех вероятностных мер на заданной σ -алгебре подмножеств множества $D = X \times Y$. При каждом $c \in \mathcal{C}$ имеем вероятностное пространство: $\langle D, B, P_c \rangle$, где B – σ -алгебра, P_c – вероятностная мера. Параметр c будем называть *стратегией природы*.

Решающей функцией (в некоторых источниках её называют алгоритмом классификации) называется соответствие $\lambda : X \rightarrow Y$.

Качество принятого решения оценивается заданной функцией потерь $\mathcal{L} : Y^2 \rightarrow [0, \infty)$.

Под риском будем понимать средние потери:

$$R(c, \lambda) = \mathbf{E}\mathcal{L}(y, \lambda(x)) = \int_D \mathcal{L}(y, \lambda(x)) P_c(dx, dy),$$

$x \in X, y \in Y$.

Цель – построить решающую функцию, которая бы минимизировала риск, но риск зависит от распределения, которое неизвестно.

Пусть $Q : D^N \rightarrow \Lambda$ – метод (алгоритм) построения решающих функций, $\lambda_{Q,S}$ – функция, построенная по выборке S методом Q , Λ – заданный класс решающих функций.

Требуется найти метод, который бы минимизировал средний риск:

$$F(c, Q) = E_{D^N} R(c, \lambda_{Q,S}).$$

Теперь ещё больше обобщим постановку, чтобы в неё укладывались в том числе задачи кластерного анализа.

Для этого не будем выделять целевые переменные, а будем считать, что имеем D – некоторое пространство значений каких-то переменных. На D имеется неизвестное вероятностное распределение. Можно считать, что имеем систему случайных величин с неизвестным совместным распределением.

Под решающей функцией будем понимать функцию, которая позволяет принимать нам нужное решение. Например, в задаче кластерного анализа решающая функция должна для любой пары точек из D сказать, относятся они к одному кластеру или к разным.

Риск — некоторый функционал, аргументами которого выступают решающая функция λ и распределение c .

В задаче кластерного анализа в качестве риска можно взять, например, отношение среднего разброса внутри кластеров к среднему расстоянию между кластерами.

Понятие метода построения решающих функций определяется как и раньше.

8.4.4 Связь с задачей проверки гипотез

Неочевидным образом, задача машинного обучения оказывается близка проверке статистических гипотез.

Самые первые методы машинного обучения, как правило, проверяли гипотезу (например, нормальности), и в соответствии с нею строили решение.

Но уже Фишер предложил метод, не делающий предположений о распределении.

- ✧ Классическая статистическая гипотеза: распределение принадлежит заданному классу.

- ✧ Статистическая гипотеза в машинном обучении: распределение таково, что заданный метод построения решающих функций обеспечивает требуемое качество.

Статистический критерий можно считать частным случаем метода построения решающих функций, когда в роли решающей функции выступает предикат, а в роли риска — ошибка второго рода. При этом функция потерь оказывается зависящей от распределения, что отличает проверку гипотез от задач машинного обучения.

Известно множество статистических критериев, но нет понятия наилучшего критерия (кроме случая известной простой альтернативы). И в машинном обучении так же отсутствует понятие наилучшего метода.

8.4.5 Классификация методов машинного обучения

В данном изложении методы машинного обучения были структурированы следующим образом:

- ✧ метрические (kNN);
- ✧ (квази)линейные;
- ✧ логические или иерархические (решающие деревья);
- ✧ композиционные, которые разделяются на:
 - общие композиции (нейросети, стэкинг);
 - ансамблевые (бустинг, случайный лес, блендинг).

Такая классификация методов весьма компактна и отражает рассмотренные нами особенности методов и их связь.

Можно классифицировать методы и другими способами, например по некоторым признакам.

Так, можно выделить методы, строящие вероятностные модели, и методы, строящие непосредственно решающие функции.

Правда, при таком делении логистическая регрессия попадёт в оба типа, поскольку она строит (частичную) вероятностную модель, которая и является решающей функцией.

Можно выделить методы, использующие вероятностные предположения: это линейный и квадратичный дискриминант, логистическая регрессия, наивный байесовский классификатор.

Традиционно в математической статистике принято разделять параметрические и непараметрические методы.

В параметрических методах предполагается, что оцениваемое распределение принадлежит заданному параметрическому семейству, например классу нормальных распределений. В этом случае для восстановления распределения достаточно оценить параметры. Частным случаем параметрического подхода является байесовский подход. В этом случае на параметрах постулируется некоторое априорное распределение и по формуле Байеса вычисляется апостериорное (при условии выборки) распределение, по которому можно строить решение.

Непараметрические методы основаны на непараметрическом оценивании распределения. Пример — парзеновские оценки плотности, которые можно рассматривать как обобщение одного из базовых методов оценивания плотности — с помощью гистограммы.

Методы машинного обучения большей частью находятся вне этой классификации, поскольку большинство методов не пытаются оценить полную вероятностную модель.

Естественно было бы классифицировать методы машинного обучения по признаку, является ли параметрическим класс используемых ими решающих функций.

Тогда подавляющее большинство методов окажутся параметрическими. Эти методы конструируют решающую функцию на основе оптимизации некоторого эмпирического функционала качества в некотором классе решающих функций, например метод опорных векторов, решающие деревья, нейронные сети. Обучение модели как раз и заключается в подборе её параметров.

Непараметрическими окажутся только метрические методы.

Однако во многих источниках все методы, которые не используют параметрические вероятностные модели, относят к непараметрическим.

Чтобы избежать путаницы, лучше вообще не использовать термины «параметрический» и «непараметрический» применительно к методам машинного обучения.

8.5 Сравнение методов построения решающих функций

Методов машинного обучения придумано большое множество. Возникает естественное желание как-то ранжировать эти методы по их актуальности и полезности, чтобы применять в первую очередь наиболее перспективные.

Для сравнения методов построения решающих функций можно выделить два подхода:

- ✧ выбор эталонного набора тестовых задач;
- ✧ формальное введение понятия оптимальности метода.

Обсудим результаты, достигнутые в рамках первого подхода.

8.5.1 «Полигон» — 1980-е

Система «Полигон» [26] была одной из первых с мире попыток систематизировать сравнение разных методов построения решающих функций (первая версия репозитория UCI появилась примерно в то же время).

В то время не было ни языка Python, ни каких-либо библиотек методов анализа данных. Были только библиотеки с бумажными журналами, где в статьях описывались разрабатываемые методы. Также были авторы этих методов, которые реализовывали свои методы в виде какого-то кода (который в теории работал). Чтобы сравнить методы, нужно было договариваться с авторами об обмене кодом либо делать свою реализацию на основе описания из статей.

В таких условиях построение подобной системы было серьезным вызовом.

Принципы и ключевые идеи:

- ✧ для каждого метода включается «эталонная» задача;
- ✧ на «своей» задаче метод должен работать лучше других;
- ✧ возможно оценить степень универсальности метода;
- ✧ тестовая единица — таблица данных.

Данные могли быть как реальные, так и синтетические.

Работы над системой способствовали выработке принципов сравнения методов машинного обучения.

8.5.2 Архив UCI

Первая версия архива UCI появилась в 1987 году.

Данный репозиторий долгое время был единственным широко используемым инструментом для сравнения методов анализа данных.

Архив актуален и в настоящее время. Он содержит большое число задач разного типа.

Задача включает сами данные и их описание с постановкой цели.

Многие задачи стали практически общеизвестны, поэтому достаточно упомянуть только название задачи, чтобы читатель понял, о чём речь.

Минусом репозитория является низкая оперативность. Сравнение полученных результатов проводится исключительно в научных публикациях, что занимает долгое время.

8.5.3 «Полигон» — 2000-е

В систему «Полигон» [13] заложены следующие принципы:

- ✧ использование реальных задач;
- ✧ большое число характеристик качества;
- ✧ основной критерий — скользящий экзамен.

Данная система является напоминанием о том, что в Kaggle отсутствует механизм сравнения методов и вычисления их «рейтинга».

В этой системе «Полигон» такой механизм был как раз реализован. Заложенные в систему идеи полезны для разработки подобного инструментария в будущем.

8.5.4 Платформа Kaggle

Платформа Kaggle появилась в 2010 году и в настоящее время является наиболее актуальным полигоном для методов машинного обучения, содержащим самую большую коллекцию задач машинного обучения. Здесь представлены практически все типы задач, которые имеют объективный критерий для сравнения результатов (решений).

Решающим достоинством платформы в этой роли является высокая мотивация участников соревнований в получении наиболее точных решений при полном отсутствии заинтересованности в продвижении каких-то конкретных методов, технологий и подходов. Это обеспечивает максимально возможную объективность в сравнении методов.

Недостатком платформы является то, что для сравнения методов приходится вручную просматривать решения победителей конкурсов и собирать статистику по используемым подходам.

Помимо функции полигона методов Kaggle выполняет функцию главного образовательного ресурса в области машинного обучения, а также самой полной базы готовых решений и примеров кода. Можно заметить, что в настоящее время самые эффективные и полезные инструменты машинного обучения и анализа данных являются полностью открытыми и бесплатными.

8.5.5 Синтетические данные

Использование синтетических данных часто встречает скептическое отношение, поскольку искусственные данные не заменяют реальные, и непонятно, насколько их свойства близки к реальным.

Однако реальные данные также не заменяют синтетические.

Их достоинства — неограниченный объём данных и управляемые свойства: байесовский уровень ошибки, степень независимо-

сти, доля шумовых переменных и т. п.

Только на синтетических данных можно убедительно продемонстрировать, за счёт чего один метод работает лучше другого.

Некоторые синтетические модели доступны в библиотеке `sklearn.datasets`, однако для более-менее серьёзных исследований приходится разрабатывать свои модели.

8.6 Понятие оптимальности метода

8.6.1 Проблема оптимальности методов построения решающих функций

При сравнении методов в роли тестовых единиц могут выступать:

- ✧ таблица данных;
- ✧ распределение;
- ✧ класс распределений.

Напомним, что метод — это отображение выборок в решения.

Для таблицы данных понятие оптимального метода не имеет смысла.

Утверждение может оказаться неожиданным, поскольку мы только что утверждали, что Kaggle позволяет сравнивать методы, а он как раз оперирует таблицами данных.

Но мы говорили о сравнении «разумных» методов.

Если же вводить понятие оптимального метода, то нужно либо строго формально определить понятие «разумности» метода, либо рассматривать все методы как любые отображения выборок в решения.

А среди любых отображений есть подкласс вырожденных методов — таких, которые все выборки отображают в одно и то же решение.

Но для любой таблицы данных всегда найдётся решающая функция, которая классифицирует её безошибочно. Значит, и соответствующий вырожденный метод также не будет делать оши-

бок на этих данных. И на кроссвалидации будет нулевая ошибка.

Метод не обучается, он классифицирует только одну таблицу, но для неё он оптимален.

Для заданного распределения оптимальный метод также вырожден — он любой выборке сопоставляет байесовское решающее правило.

Об оптимальности метода можно говорить только для класса распределений.

Но, как мы выяснили ранее, даже для нормальных распределений оптимальный метод неизвестен.

Более того, даже само понятие оптимальности метода не введено.

8.6.2 Минимаксный подход к оцениванию качества

Введём обозначение для среднего риска

$$\mathcal{R}(c, Q) = E_S R(c, \lambda_{Q,S}),$$

где математическое ожидание берётся по всем выборкам.

Мы не можем определить лучший метод с точки зрения среднего риска, поскольку $\mathcal{R}(c, Q)$ зависит от распределения, поэтому задача выбора многокритериальна.

Для таких случаев существует минимаксный подход, когда оптимальным считается метод, лучший в худшем случае, т. е.

$$Q^* = \arg \min_Q \max_c \mathcal{R}(c, Q).$$

Однако максимальный по всем распределениям средний риск одинаков для любого метода и достигается при нулевой гипотезе (когда распределения классов одинаковы).

Чтобы получить невырожденный результат, нужно ограничить класс распределений.

Может показаться, что достаточно ввести ограничение сверху на байесовский уровень ошибки. Но это не имеет смысла, если не

ограничить вид распределений. Легко привести пример вероятностной модели, когда байесовский уровень ошибки нулевой, но все существующие методы дадут решение с ошибкой 0,5.

Представляется перспективной идея ввести понятие достижимого уровня риска. Его необходимо задавать для каждого распределения из класса. Конкретное значение может определяться на основе эталонного метода или из других соображений.

Использование максимума не абсолютного риска, а отнесённого к достижимому уровню риска, позволяет (на основе минимаксного подхода) ввести осмысленное понятие метода, оптимального на классе распределений.

8.6.3 Выбор классов распределений

Класс распределений, используемый в качестве тестовой единицы для сравнения качества методов, не должен быть ни узким ни широким — иначе получаем соответственно вырожденный метод или аналог NFL (теоремы о бесплатном завтраке).

Класс должен быть достаточно богатым (представительным) для исследований: обеспечивать все значения достижимого риска, а также содержать модели, дающие максимальное смещение эмпирического риска.

Желательно, чтобы класс поддерживал параметр сложности и был универсальным по сложности.

Также желательная замкнутость относительно допустимых преобразований пространства переменных.

Возможные варианты классов распределений:

- ✧ класс кусочно-постоянных распределений;
- ✧ класс нормальных распределений;
- ✧ класс, сформированный случайными решающими деревьями;
- ✧ ядерные функции для условных вероятностей.

8.7 Некоторые задачи машинного обучения

Перечислим типичные задачи машинного обучения.

Распознавание образов (классификация «с учителем») — задача, которую рассматривали наиболее подробно.

Регрессионный анализ (восстановление зависимостей) — точнее было бы построение зависимостей.

Кластерный анализ (таксономия, автоматическая группировка, классификация «без учителя») — позднее рассмотрим кратко. Похоже на оценивание смеси распределений.

Задача упорядочивания объектов — частный случай классификации, когда оцениваем вероятность принадлежности классу и упорядочиваем по вероятности.

Задача обнаружения закономерностей — нет целевой переменной. Похожа на кластеризацию, но более общая.

Прогнозирование временных рядов — можно сводить к другим задачам, но есть специализированные подходы.

Планирование эксперимента — обобщение задачи регрессии на случай, когда можем выбирать точки. Частный случай обучения с подкреплением.

Поиск глобального экстремума — частный случай планирования эксперимента, когда нужно аппроксимировать не всю функцию, а только найти максимум.

Анализ клиентских сред — выявление связей между объектами, например покупателями и товарами.

Тематическое моделирование — классификация текстов по темам.

Генеративные модели — сейчас наиболее популярны. Демонстрируют поведение, которое в них не закладывалось.

Эти задачи можно сгруппировать по следующим типам:

- ✧ обучение «с учителем»;
- ✧ обучение «без учителя»;
- ✧ обучение «с подкреплением».

Это были перечислены задачи машинного обучения. Есть другой подход в области искусственного интеллекта — алгоритмический: экспертные системы, символьные вычисления. Данные направления выходят за рамки тематики курса.

8.8 Выводы

По прочтении главы читателю может показаться, что заявленная в оглавлении строгая математическая постановка задачи машинного обучения так и не была сформулирована... «Ну, откровенно говоря, да» (©). Общей математической постановки, пожалуй, попросту не существует. Можно поставить задачу построения метода, минимизирующего средний риск. Но это не будет корректной математической задачей, поскольку средний риск зависит от распределения, которое мы не знаем. А ввести универсальное ограничение на класс распределений мы не можем. Хотя реально такое ограничение, видимо, существует, поскольку природа не «играет» против нас целенаправленно.

Вместе с тем машинное обучение может считаться математической дисциплиной, поскольку при создании и исследовании его методов возникают вполне строгие математические задачи, которые решаются математическими методами.

Основные выводы по главе можно сформулировать следующим образом.

- ✧ Задача машинного обучения заключается в конструировании методов построения решающих функций.
- ✧ Критерий качества первичен. Он определяет оптимальное решение.
- ✧ Необходимо различать целевой и эмпирический критерии.
- ✧ До сих пор не введено понятие, что значит: один метод классификации лучше другого.
- ✧ Задачи машинного обучения похожи на задачи проверки статистических гипотез.

8.9 Примеры кода

Приведём пример кода для оценки компоненты разброса (variance) на реальных данных.

Чтобы оценить разброс, нужно строить решающие функции по различным обучающим выборкам и делать прогноз на одних и тех же тестовых данных.

Напрашивается идея в качестве обучающих использовать бутстреп выборки. Но это плохая идея. Технически мы, конечно, можем вычислить разброс по бутстреп выборкам, но такая оценка будет заведомо плохой. И даже неизвестно, насколько плохой, поскольку оценок её точности не существует.

Мы обучающие выборки будем брать непересекающимися. Сначала разделим всю таблицу данных на обучающую и тестовую части. После этого обучающую часть поделим на большое число K маленьких обучающих выборок, т. е. фолдов. Для этого используем класс `KFold`, но для обучения берём один фолд (который в обычном режиме был бы тестовым).

В некотором роде у нас получается «кроссвалидация наоборот». Обучаемся на одном фолде. А тестировать можем на оставшихся. Так бы мы получили для каждого объекта $K - 1$ различных прогнозов, по которым можно было бы вычислить смещение и разброс. Но технически это было бы чуть сложнее, поэтому мы тестовую выборку отделили заранее.

Проще всего было бы оценить разложение для задачи регрессии, но мы рассмотрим задачу классификации, чтобы продемонстрировать подходы для более сложного случая.

В качестве метрики качества возьмём долю ошибок, смещение определим как долю ошибок для усреднённого решения, а разброс вычислим как обычную дисперсию прогноза. При этом прогнозируется номер класса, принимающий значения 0 или 1. Таким образом определённые величины не образуют разложения ошибки, но содержательный смысл их сохраняется.

Дополнительно вычислим ещё один вариант дисперсии, а именно дисперсию «сырого прогноза», которую обозначим как `raw`

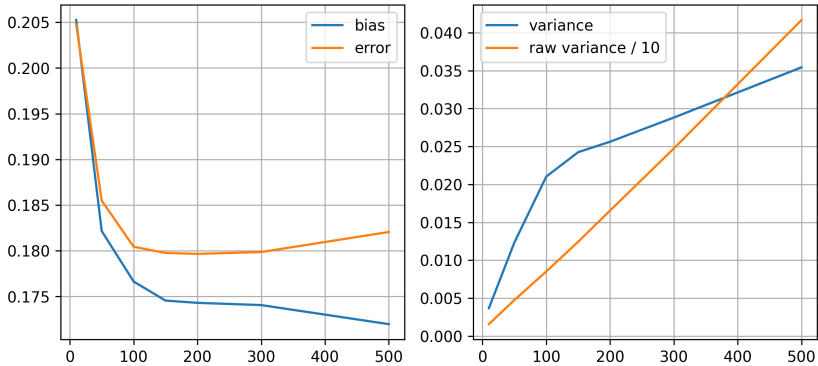


Рис. 8.4. Зависимость доли ошибок и компонент разложения от размера ансамбля

variance.

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold, train_test_split

def predict(model, X, Y, Xt, M):
    P = np.zeros((Xt.shape[0], M))
    kf = KFold(M)
    for i, (_, ids) in enumerate(kf.split(X)):
        X_train, Y_train = X[ids], Y[ids]
        model.fit(X_train, Y_train)
        P[:, i] = model.decision_function(Xt)
    return P

def error(p, y):
    return ((p > 0) != (y > 0)).mean()

def decompose(P, Y, scorer):
    pred = P.mean(axis=1)
    u = (P > 0).mean(axis=1)
    return {
        'bias': scorer(pred, Y),
        'raw variance': np.var(P, axis=1).mean(),
    }

```

```

        'variance': (u*(1-u)).mean(),
        'error': sum(scorer(v,Y) for v in P.T) / P.shape[1]
    }

data = pd.read_csv('https://archive.ics.uci.edu/ml/' +
    'machine-learning-databases/adult/adult.data',
    header=None)
X = data[[2,4,10,11,12]]
Y = (data[14]== '>50K').astype(int)

X_train, X_test, Y_train, Y_test = train_test_split(
    X.to_numpy(), Y.to_numpy(),
    test_size=0.25, random_state=0)

NT = [10,50,100,150,200,300,500]
r = []

for ntrees in NT:
    clf = GradientBoostingClassifier(n_estimators=ntrees,
        learning_rate=0.05, max_depth=2, random_state=0)
    P = predict(clf, X_train, Y_train, X_test, M=20)
    r.append(decompose(P, Y_test, error))

df = pd.DataFrame(r, index=NT)
df['raw variance / 10'] = df['raw variance'] / 10

_, ax = plt.subplots(1,2, figsize=(9,4))

df[['bias', 'error']].plot(grid='on', ax=ax[0])
df[['variance', 'raw variance / 10']].plot(ax=ax[1])

```

Вызов `model.decision_function` в функции `predict` можно заменить следующими строками.

```

g = model.predict_log_proba(Xt)
P[:, i] = g[:, 1] - g[:, 0]

```

Результат работы программы приведён на рис. 8.4. Видим, что качественное поведение величин соответствует ожиданиям.

Построение подобных графиков, безусловно, интересно и полезно в исследовательских целях, но бессмысленно при решении практических задач. Для моделирования мы были вынуждены

взять подвыборки малого размера, поэтому полученные результаты ничего не говорят о поведении метода на полной выборке (в частности, не поможет контролировать переобучение).

Контрольные вопросы

1. Как можно сформулировать задачу машинного обучения?
2. Что такое целевая метрика, как обосновать её выбор?
3. Как выбирается функция потерь для обучения, как обосновать её выбор?
4. Что такое ROC-кривая, PR-кривая?
5. В чём содержательный смысл логарифмической функции потерь?
6. Что такое погрешность аппроксимации и статистическая погрешность?
7. Что такое сложность модели? Какие методы (классы решающих функций) называются универсальными?
8. Что такое переобучение и недообучение?
9. Как подбирать параметры сложности модели?
10. Как получить (вывести в простейшем случае) разложение ошибки на шум, смещение и разброс?
11. Можно ли за счёт увеличения размера ансамбля неограниченно (практически до нуля) уменьшать разброс?
12. Как оценить компоненты разложения на реальных данных?
13. Какие принципиальные трудности возникают при сравнении методов по качеству?

Глава 9

Применение теории вероятностей

9.1 Вероятностная интерпретация

В данном учебнике материал структурирован от частного к общему. Так, мы сначала изучили конкретные методы машинного обучения и только потом стали формулировать общую постановку для решаемых задач.

При этом с самого начала было заявлено, что всё изложение будет вестись в рамках вероятностной интерпретации. Пришло время разобраться, что имеется в виду.

В машинном обучении существуют подходы, основанные на нечётких логиках, комбинаторных рассуждениях или, например, теории возможностей.

Многие методы машинного обучения могут быть сформулированы чисто алгоритмически, без упоминания вероятностей. Один из классических примеров — дискриминант Фишера (использует исключительно геометрические соображения).

Однако, по мнению автора, вероятностная интерпретация — это самый простой, понятный и заведомо корректный способ обосновать применимость модели, построенной по обучающей выборке, к новым объектам. При этом развитие других подходов можно

только приветствовать.

Чтобы корректно применять вероятностный подход, нужно выяснить содержательный смысл вероятности.

Если спросить «простого обывателя», знает ли он, что такое пространство и время, то он ответит: «Конечно, да».

Если этот же вопрос задать физику (или, пуще того, философу), то он, вероятно, ответит: «Не знаю».

Автор встречал высказывания специалистов в области теории вероятностей, что они не знают, что такое вероятность. Попробуем приблизиться к их уровню незнания.

В классических учебниках содержательной интерпретации вероятности обычно уделяется немного места. Так, в учебнике [6] этому (вместе с историческим экскурсом) посвящено всего 4 страницы.

В частности, там говорится, что приближение частоты исхода к вероятности — «это объективный закон природы, который лежит в основе теории вероятностей».

Это, очевидно, полностью верное утверждение. Но из него часто делают очень спорный вывод, что вероятность можно использовать только в ситуациях, когда можно непосредственно наблюдать этот факт, т.е. когда есть возможность многократного повторения эксперимента. Однако из того, что теория вероятностей создана на основе наблюдений за повторяющимися событиями, не следует, что её можно использовать только для повторяющихся событий.

То, что сходимости частот к вероятностям — это объективный закон природы, не противоречит тому, что он может быть следствием каких-то более фундаментальных законов. Кроме того, эмпирический закон не может одновременно быть математической теоремой, однако закон больших чисел является чисто математическим утверждением и касается именно сходимости частот к вероятностям. В дальнейшем будет предложен один из вариантов разделения закона сходимости частот на чисто математические и эмпирические факты.

Также считается, что вероятность может быть связана как с

недостаточностью знаний, так и с объективной неопределённостью.

Когда-то наука вообще не признавала существование объективной неопределённости, и в то время единственно возможной была интерпретация случайности как незнания. Однако в квантовой механике вероятность — это как раз максимально полное знание об объекте.

Получается, что вероятность в одних случаях — незнание, в других — максимально полное знание. Выход из противоречия может быть следующий. Считаем, что вероятность «на самом деле» связана с объективной неопределённостью, но мы вправе её использовать (как искусственный эвристический приём) и в случае незнания.

Прежде всего, необходимо договориться о терминологии.

Очень многие дискуссии, которые кажутся предметными, на самом деле оказываются терминологическими спорами.

То же «пространство» в физике, математике и философии — это совершенно разные понятия.

Вероятность как математическое понятие и как некоторый аспект реального мира — тоже разные понятия. Вначале мы рассмотрим вероятность как чисто математическую конструкцию, после чего перейдём к содержательной интерпретации. Это позволит максимально разделить математические и эмпирические факты.

Для начала вспомним, что такое вероятность как абстрактное математическое понятие.

9.2 Вероятность как математическое понятие

В этом параграфе мы дадим максимально компактное изложение математических понятий, которые понадобятся в дальнейшем.

9.2.1 Аксиоматическое определение

Основным понятием теории вероятностей является вероятностное пространство, которое включает множество элементарных исходов, σ -алгебру и вероятностную меру.

Определение 2 Множеством элементарных исходов называется некоторое множество Ω , подмножества которого (не обязательно все) образуют σ -алгебру Λ . Элементы из Λ называются событиями.

Множество \emptyset называется невозможным событием. Множество Ω называется достоверным событием.

Определение 3 Вероятностью называется функция P , заданная на Λ , удовлетворяющая аксиомам меры и такая, что

$$P(\Omega) = 1.$$

Определение 4 Пусть A и B – события и $P(B) > 0$. Условной вероятностью называется отношение

$$P(A/B) = \frac{P(AB)}{P(B)}.$$

Определение 5 Пусть $P(A) > 0$ и $P(B) > 0$.

Говорят, что A не зависит от B , если $P(A/B) = P(A)$.

События A и B называются независимыми, если

$$P(AB) = P(A)P(B).$$

Элементарно доказывается, что данные определения эквивалентны, а также что независимость A от B эквивалентна независимости B от A .

Если $P(B) = 0$, то условной вероятности $P(A/B)$ тоже можно придать смысл, в частности, используя условные распределения.

С содержательной точки зрения независимость — это когда факт наступления одного события не влияет на вероятность другого. Получается, что для содержательного понимания независимости (двух событий) должны быть определены обе условные вероятности.

Если событий больше двух, то под независимостью будем понимать независимость в совокупности.

9.2.2 Закон больших чисел

Существует множество формулировок закона больших чисел. Приведём наиболее простую из них.

Рассмотрим серию из n независимых испытаний (схему Бернулли), в каждом из которых возможны только два исхода 0 и 1, причём вероятность исхода 1 фиксирована и равна p .

Обозначим через m количество единиц в полученной серии.

Теорема 3 *Для частоты успехов в схеме Бернулли имеет место сходимость по вероятности, а именно для любого $\varepsilon > 0$*

$$P\left(\left|\frac{m}{n} - p\right| > \varepsilon\right) \xrightarrow{n \rightarrow \infty} 0.$$

Заметим, что в этом утверждении нет бесконечных последовательностей экспериментов, есть только конечные последовательности неограниченного размера. Это существенная тонкость, благодаря которой интуитивно воспринимать данный закон гораздо легче, чем его усиленную форму (легко представить сколь угодно долгий эксперимент, но трудно представить бесконечный эксперимент).

Для формулировки усиленного закона больших чисел нужно рассмотреть множество всех бесконечных последовательностей из 0 и 1. Обозначим такую последовательность через (z_j) , $j = 1, \dots, \infty$, $z_j \in \{0, 1\}$. Множество всех таких последовательностей обозначим через Z .

Для наглядности мы можем отобразить это множество на интервал $[0, 1]$, интерпретируя последовательность как двоичную запись дробной части вещественного числа (целая часть равна 0).

Такое отображение не взаимно однозначно, поскольку есть пары последовательностей, кодирующих одно и то же число, например $0,101(0)_2 = 0,100(1)_2$. Однако множество таких пар всего лишь счётно и имеет вероятностную меру 0, поэтому им можно пренебречь.

В любом случае, данное отображение задаёт случайную величину. При $p = 0,5$ эта величина имеет равномерное распределение на $[0, 1]$. При $p = 0$ и $p = 1$ распределение дискретно (вероятность сосредоточена в одной точке). При остальных значениях p распределение становится сингулярным.

Каждой последовательности (z_j) из Z можно сопоставить последовательность (s_j) частичных средних. Здесь $s_j = \frac{1}{j} \sum_{k=1}^j z_k$.

Обозначим через $Z_{(p)}$ подмножество последовательностей (z_j) из Z , для которых соответствующие последовательности частичных средних сходятся к p в обычном смысле.

Теорема 4 *Вероятностная мера множества $Z_{(p)}$ равна 1.*

Это один из вариантов усиленного закона больших чисел.

9.2.3 Частотное определение вероятности

Очевидно, что некорректно определять математическое понятие через результат реального «физического» эксперимента. То, что в реальных экспериментах частота стремится к вероятности, — это эмпирический факт, а математическое понятие мы должны определять исключительно математическими средствами.

Также мы не можем для определения понятия вероятности использовать закон больших чисел, поскольку сходимость в этом законе основана на понятии вероятности.

Поскольку нельзя вероятность определять через вероятность, для «частотного определения» вероятности необходимо сформу-

лизовать независимое от вероятности математическое понятие сходимости частот. Можно использовать обычное понятие сходимости последовательности, но тогда нужно каким-то образом независимо описать множество $Z_{(p)}$.

Если мы изначально не закладываем в параметр p смысла вероятности, то $Z_{(p)}$ – это просто параметрическое семейство классов последовательностей. И нам нужно как-то обосновать осмысленность этого семейства, тогда и параметр p станет не просто произвольным числом, а осмысленной величиной, которую можно будет назвать вероятностью. В этом направлении есть интересные подходы, например с использованием понятия колмогоровской (алгоритмической) сложности [9].

Однако этот путь гораздо более сложен, чем аксиоматическое определение вероятности как меры, поэтому использовать частотный подход к определению вероятности кажется нецелесообразным (если речь о прагматическом использовании вероятности, а не теоретических исследованиях).

9.2.4 Условное распределение

Условная вероятность $P(A/B)$ определяется как $\frac{P(AB)}{P(B)}$, если $P(B) > 0$. Но в машинном обучении типична ситуация, когда требуется использовать условную вероятность, когда вероятность условия равна нулю.

В задаче классификации мы прогнозировали условную вероятность заданного класса в каждой точке x признакового пространства. Вероятность такого условия равна нулю.

Вспомним, как мы определяли такую условную вероятность. Использовалось отношение

$$P^Y(y/x) = \frac{\varphi(x, y)}{\varphi^X(x)},$$

где $\varphi^X(x)$ – безусловная плотность вероятности для x , $\varphi(x, y) = \varphi_y^X(x) P^Y(y)$ – совместная плотность, $\varphi_y^X(x)$ – условная плотность для x при условии y , $P^Y(y)$ – безусловная вероятность класса y .

Здесь мы в позиции верхнего индекса указываем множество, на котором построено вероятностное пространство. Так, $P^Y(y/x) = P(Y(\cdot) = y/x)$ показывает, что именно случайная величина $Y(\cdot)$ принимает значение y . Позицию нижнего индекса используем, чтобы указывать условие.

Если плотность в знаменателе обращается в ноль, то условная вероятность по этой формуле не определена, но мы её можем доопределить произвольным образом. Тогда можем договориться, что условная вероятность — это любая функция $P^Y(y/x)$, удовлетворяющая соотношению

$$\varphi(x, y) = P^Y(y/x)\varphi^X(x).$$

Данное уравнение определяет условную вероятность с точностью до значений на множестве меры ноль. Но нас это вполне устраивает.

Если Y — непрерывная величина, то можно определить условную плотность $\varphi_x^Y(y)$ через соотношение

$$\varphi(x, y) = \varphi_x^Y(y)\varphi^X(x).$$

Для полноты приведём общие определения условных распределений. Читатель может их пропустить и перейти к описанию примера Бореля–Колмогорова.

В общем случае (в том числе если X не является непрерывной величиной) условную вероятность можно определить через производную Радона–Никодима (там, где она существует)

$$P^Y(y/x) = \frac{P(dx, y)}{P^X(dx)}.$$

Для произвольных случайных величин условное распределение можно определить через соотношение

$$P(dx, dy) = P^Y(dy/x) P^X(dx).$$

Функция распределения существует для любых случайных величин, поэтому выразим условную вероятность в этих терминах.

Пусть $F(x, y)$ – функция совместного распределения и

$$F^X(x) = \int_Y dF(x, y), \quad F^Y(y) = \int_X dF(x, y)$$

– маргинальные функции распределения.

Функцию условного распределения $F_x^Y(y)$ можно в самом общем виде определить как функцию, обеспечивающую выполнение тождества

$$\iint_{X \times Y} \dots dF(x, y) = \int_X \int_Y \dots dF_x^Y(y) dF^X(x).$$

Существование условного распределения при таком подходе обеспечивается возможностью сведения двойного интеграла к повторному.

Через условные распределения можем записать, например, формулу полной вероятности

$$F^Y(y) = \int_X F_x^Y(y) dF^X(x).$$

Если в условии стоит некоторое «изолированное» событие, имеющее нулевую вероятность, то условная вероятность не имеет смысла. Но если событие в условии соответствует принятию случайной величиной некоторого значения, то условные вероятности определяются почти однозначно (различия на множестве меры ноль).

Однако при использовании условных распределений требуется аккуратность в интерпретации результатов, что иллюстрирует пример (или парадокс) Бореля–Колмогорова.

Пусть на единичной сфере задано равномерное распределение. Каким будет условное распределение на большой окружности сферы?

Приведём наглядную аналогию. Возьмём самый обычный глобус, изготовленный из целлюлозы или пластика. Предположим, что он сделан достаточно аккуратно: материал однородный и толщина стенки одинакова по всей поверхности. Тогда распределение

массы (глобуса) по поверхности будет равномерным. Поверхностная плотность будет пропорциональна толщине стенки, т.е. постоянная.

А какое будет распределение массы на глобусе вдоль экватора? Очевидно, тоже равномерное, ведь толщина стенок одинакова везде. Значит вдоль любого меридиана распределение также будет равномерным.

Есть однако некоторая тонкость. Дело в том, что масса глобуса, находящаяся точно на экваторе, равна нулю. Можно ли говорить о распределении этой нулевой массы по экватору? Как вообще возможно распределять нулевую массу? Непосредственно такое сделать невозможно. Но можно распределить ненулевую массу по поверхности и говорить о распределении массы вдоль каких-то линий в том смысле, что получившаяся плотность меняется вдоль этих линий.

Перейдём собственно к парадоксу.

На любом глобусе есть широта (обозначим α) и долгота (обозначим β).

При равномерном распределении по поверхности, распределение по долготе будет также равномерным (на интервале $[-\pi, +\pi]$), а по широте — неравномерным. Плотность вероятности для α будет пропорциональной длине соответствующей параллели, откуда получаем

$$\varphi(\alpha) = \frac{1}{2} \cos \alpha, \quad -\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}.$$

Случайные величины α и β независимы.

Предположим, что мы отправили на некоторую планету автоматический аппарат, потеряли с ним связь, и он спустился, выбрав точку приземления наугад. Что можно сказать о широте его местоположения? То, что она имеет распределение $\varphi(\alpha)$.

Пусть теперь аппарат вышел на связь и сообщил свою долготу (вариант 1). Что теперь мы знаем о широте? Только то, что она имеет распределение $\varphi(\alpha)$. Величины независимы, и условное распределение (при известной долготе) совпадает с безусловным.

А теперь предположим, что аппарат не выходил на связь, и

мы стали его искать сами (вариант 2). Спустились в некоторую точку планеты и стали двигаться вдоль меридиана. В итоге обошли планету по кругу и нам повезло: аппарат нашли. Какое теперь распределение для широты, на которой мы его встретили? Теперь оно равномерное.

Но если бы мы приземлились не наугад, а зная долготу (вариант 3), то распределение для широты точки обнаружения аппарата вновь стало бы неравномерным, как в варианте 1, т. е. $\varphi(\alpha)$.

Собственно, в этом и заключается парадокс. Вернее, конечно, не парадокс, а пример с интересным результатом.

Пояснения, пожалуй, требует вариант, когда мы ищем аппарат, спустившись в наугад выбранную точку. Естественно уточнить, что в варианте 2 при движении по поверхности мы видим объекты в некоторой полосе вдоль траектории. И ширина этой полосы не зависит от географической широты, поэтому вероятность обнаружения аппарата на единицу пройденного пути одна и та же.

Никого не удивляет, что плотность зависит от выбора переменных. Если мы делаем замену переменных, то плотность вероятности меняется в соответствии с якобианом преобразования. Если плотность вероятности на единицу площади поверхности постоянная, то плотность в координатах (α, β) уже неравномерная.

В варианте 2 мы неявно подразумеваем, что используем некоторую локальную (в узкой полосе движения) параметризацию поверхности планеты. Первый параметр — широта, а второй параметр (обозначим его u) — координаты точки на оси, перпендикулярной направлению движения. При этом площадь элементарного приращения $\Delta\alpha \times \Delta u$ не меняется по ходу движения, поэтому плотность вероятности (в пространстве этих параметров) оказывается равномерной.

Таким образом, «парадокс» возникает не потому, что в условии стоит событие меры нуль, а из-за разной параметризации пространства исходов. На разных наборах параметров совместная плотность получается различной, и вполне естественно, что условные плотности тоже получают различными.

Сформулируем окончательный ответ на изначальный вопрос. Каким будет условное распределение на большой окружности сферы, если на сфере задано равномерное распределение? Можно дать два ответа.

1. Условное распределение не определено, поскольку не указана параметризация поверхности сферы.
2. Условное распределение будет равномерным, если предположить, что условие задачи подразумевает, что следует использовать параметризацию, для которой площадь элементарных приращений постоянна вдоль данной окружности.

На этом же примере можно обсудить, как будет себя вести функция условной вероятности. Предположим, что аппарат при посадке имеет шанс разбиться, и эта вероятность зависит от местоположения. Иными словами, для каждой точки планеты есть объективная (зависит от физических условий в данном месте: погоды, рельефа и т. п.) вероятность, что аппарат разобьётся при попытке сесть в эту точку. Это пример условной вероятности, когда вероятность события в условии нулевая. Но эта условная вероятность вполне определённая, т. е. объективная, и не зависит от выбора системы координат на местности.

Таким образом, постановка задачи построения решающей функции, когда требуется оценить вероятности классов в каждой точке пространства, является корректной.

9.3 Вероятность как эмпирическое понятие

9.3.1 Эмпирические факты

Существует «всемирный эмпирический закон стабилизации частот» [5], который можно сформулировать, например, как: «для широкого класса стохастических экспериментов среднее арифметическое суммарного результата стабилизируется с ростом числа однородных испытаний».

Но ведь, по сути, то же самое утверждает закон больших чи-

сел, который является чисто математическим утверждением. Как одно и то же утверждение может одновременно быть и математической теоремой и эмпирическим законом? Чтобы закон был чисто эмпирическим, его следует сформулировать иначе.

Сейчас мы попытаемся полностью отделить математические факты от эмпирических и сформулируем чисто эмпирические факты.

Эмпирический факт 1 *В ситуациях неопределённости существует рациональный выбор, основанный на вероятности. Вероятности, близкие к 1, соответствуют практической достоверности.*

Пример 1 *Пусть имеется две пронумерованные шкатулки, в каждой из которых может быть приз. В первой шкатулке приз с вероятностью 0,5, во второй такой же приз находится с вероятностью 0,55. События независимы, поэтому призы могут быть и в обеих шкатулках. Разрешается выбрать только одну шкатулку.*

В данном примере существует рациональный выбор, а именно, правильным решением будет выбор второй шкатулки.

Заметим, что обоснованность такого выбора не зависит от того, дадут ли нам возможность играть многократно. Это может быть разовая акция.

Откуда мы знаем, что вероятности именно такие? Нам их сообщили по условию задачи. При этом не важно, как именно организаторы акции обеспечили «объективность» этих вероятностей. Например, для первой шкатулки бросали монету и приз положили в случае герба. А для второй шкатулки извлекали шар из урны, в которой из 100 шаров 55 белых. Приз положили, если попался белый шар.

Эмпирический факт 2 *Условная вероятность $P(A/B)$ есть вероятность события A , когда известно, что произошло B .*

Похоже, что следствием этого факта является «аксиома независимости» [5], согласно которой «причинно несвязанные (в бытовом смысле) события являются независимыми и в математическом (вероятностном) смысле».

Сам факт 2, несмотря на кажущуюся тавтологичность, по сути констатирует, что условная вероятность в математическом смысле является также условной вероятностью в «физическом» (или бытовом) смысле.

Пожалуй, эмпирический факт 2 можно сформулировать в более фундаментальном виде, а именно: информация о том, что произошло некоторое событие не меняет отношений вероятностей событий, которые целиком в него входят. В частном случае, если исходы (конечное множество) равновероятны, то они остаются равновероятны при условии любого события. Приведённая ранее формулировка может быть следствием данной.

Утверждение 1 *«Всемирный эмпирический закон стабилизации частот» является прямым следствием эмпирических фактов 1 и 2 и закона больших чисел.*

Данное утверждение является не совсем математическим, но его можно доказать логическим рассуждением.

Доказательство. Согласно закону больших чисел для любого выбранного уровня практической достоверности существует такое число экспериментов, при котором вероятность нахождения частоты успехов в заданной окрестности вероятности превосходит этот уровень.

Значит, при этом числе экспериментов мы можем считать практически достоверным событие попадания частоты в эту окрестность.

Отметим, что в самих сформулированных эмпирических фактах никаких многократных повторов эксперимента в одинаковых условиях не требуется.

9.3.2 Достоверность и практическая достоверность

В работе [23] имеется любопытное утверждение: если вероятность события равна 0, то «в длинной серии экспериментов событие произойдёт лишь в очень малой доле всех случаев».

На самом деле, если вероятность события равна нулю, то вероятность хотя бы одного его появления в сколь угодно длинной серии независимых испытаний также равна нулю. И мы можем быть уверены, что событие не произойдёт ни разу. С одной стороны, «ни разу» — это вполне себе «малая доля» случаев. С другой стороны, фразу Крамера можно понимать как утверждение, что в бесконечной серии событие гарантированно произойдёт хотя бы один раз, что, разумеется, неверно.

Если событие не является \emptyset , но имеет нулевую вероятность, то мы можем быть абсолютно уверены, что даже в счётной серии экспериментов оно не произойдёт ни разу. При этом оно не является невозможным.

На бытовом уровне это кажется странным, и возникает естественный вопрос: в каком смысле событие возможно, если оно точно не произойдёт? Бытовую аналогию можно привести такую: законопослушный гражданин может нарушить закон, но не будет этого делать.

Аналогия, конечно, очень поверхностная. На самом деле причина странности в том, что континуум исходов — чисто математическая абстракция (в которой много контринтуитивного). А реально в природе существует только один бесконечный объект, и это не Вселенная.

До относительно недавнего времени парадоксальным казался даже тот факт, что сумма бесконечного числа слагаемых может быть конечной, и «парадокс» Ахиллеса и черепахи воспринимался как настоящий парадокс. Потребовалось время, чтобы «привыкнуть» к этому факту и перестать воспринимать его парадоксом.

С различием между невозможным событием и событием, которое гарантированно не случится, похоже, аналогичная ситуация — требуется время, чтобы осознать отсутствие парадокса.

Заметим, что мы можем быть уверены, что такое (с нулевой

вероятностью) событие не произойдёт, только если мы его «задумали» заранее. Предположим, что мы наугад выбираем точку на отрезке. Каждая точка отрезка соответствует событию, имеющему нулевую вероятность. Однако одно из таких событий в эксперименте гарантированно произойдёт. Но если мы заранее зададим на отрезке какие-то (не более счётного множества) точки, то можем быть вполне уверены, что наугад поставленная точка в них не попадёт.

На самом деле провести подобный эксперимент мы можем только умозрительно. Нет никакого способа провести его физически. Если мы наугад ткнём карандашом в лист бумаги, то получим не точку, а пятно. На компьютере сгенерировать случайное вещественное число (с равномерным распределением на интервале $[0, 1]$) тоже невозможно. Просто потому, что числа в компьютере представляются конечным числом разрядов (и реальный выбор всегда производится не из континуального, и даже не из счётного, а лишь из конечного множества вариантов, хоть и большого).

Может возникнуть вопрос, зачем вообще использовать бесконечность в математике, если её в природе не существует. Всё дело в том, что с бесконечностью всё намного проще, чем без неё.

Выбор случайной точки на отрезке — это то же самое, что генерирование бесконечной последовательности знаков представления дроби. И угадывание всей такой последовательности не является невозможным событием. Но мы можем быть абсолютно уверенными, что нам это сделать не удастся.

Событие, имеющее вероятность 1, является практически достоверным. Кроме того, практически достоверными мы принимаем события, вероятность которых близка к 1. Порог близости зависит от величины риска, который мы считаем допустимым в конкретной ситуации.

9.4 Случайность как незнание

9.4.1 Парадокс конвертов

Почти все парадоксы теории вероятностей парадоксами не являются, а являются лишь фактами, непривычными обывательскому восприятию (например, «парадокс» Монти-холла).

Но «парадокс конвертов» — одно из двух исключений (второе рассмотрим ниже). Данный парадокс говорит о том, что вероятность нельзя назначать произвольно.

Пример 2 *Игроку предлагается выбрать один из двух одинаковых конвертов. Известно, что в одном из них сумма в 10 раз больше, чем в другом. Выбрав конверт, игрок может его вскрыть и принять решение: забрать его или забрать другой конверт.*

В оригинальной формулировке суммы в конвертах отличаются в два раза, но это приводит к тому, что в выкладках сложно понять: конкретная двойка — это отношение цен или число конвертов. Поэтому для большей выразительности мы задали, что суммы отличаются в 10 раз.

Предположим, что в выбранном конверте игрок обнаружил 1000 рублей. Тогда во втором может быть 100 или 10000. Если предположить, что эти варианты равновероятны, то получаем, что в том конверте в среднем 5050 руб. Получается, что выгоднее взять второй конверт.

Но это решение не зависит от суммы, обнаруженной в выбранном изначально конверте. В другом конверте всегда ожидание в 5,05 раз больше. Получается, что первый конверт можно даже не вскрывать. Но если конверт не вскрыт, то оба конверта равноправны (нет понятия «другой» конверт) и любой выбор случаен.

Разберём парадокс.

Предположим, что мы перемешали конверты и наугад выбрали тот, который будем вскрывать. Положим конверты перед собой: тот, что решили вскрыть, положим слева, другой — справа.

Пусть событие A – в левом конверте сумма в 10 раз больше, чем в правом, событие B – в правом конверте сумма в 10 раз больше, чем в левом. Эти события, очевидно, имеют вероятность по 0,5.

Предположим, что в левом конверте лежит сумма в x рублей. Обозначим это событие через C .

Тогда математическое ожидание суммы y в правом конверте есть:

$$E_{y|x}y = \frac{x}{10} P(A|C) + 10x P(B|C).$$

Это условное математическое ожидание, и именно такое фигурирует в рассуждениях игрока.

Нотация $E_{y|x}f(x, y)$ означает усреднение функции $f(x, y)$ по условному распределению для случайной величины y при фиксированном x .

Однако мы ничего не знаем об условных вероятностях $P(A|C)$ и $P(B|C)$, а возникновение парадокса говорит о том, что мы их не можем полагать равными.

Иными словами, незнание, т. е. отсутствие информации о вероятности каждого варианта, не даёт нам оснований считать эти варианты равновероятными.

Отсюда напрашивается вывод, что источником случайности является именно объективная неопределённость, а не незнание.

Парадокс является примером некорректного использования байесовского подхода.

Чтобы избежать парадокса, корректная вероятностная модель должна быть полной (допускать симуляцию) либо допускать уточнение до корректной полной модели. Применительно к примеру это означает, что мы должны рассмотреть процесс заполнения конвертов. Один из вариантов процедуры подготовки конвертов такой. Берём наугад одну из купюр 100, 500, 1000, 5000 рублей и кладём её в один конверт. Во второй конверт помещаем сумму в 10 раз больше.

Теперь парадокса нет. Если во вскрытом конверте игрок увидел 1000 рублей, то ему выгоднее взять другой конверт, поскольку математическое ожидание для него действительно будет 5050. Но

если в первом конверте оказалось 50000, то его и нужно взять, поскольку во втором тогда может быть только 5000.

Заметим, что неполные модели использовать всё же можно. Например, логистическая регрессия является моделью условного распределения, а не совместного.

Можно ли в задаче с конвертами придумать такую полную модель, чтобы при любом x условные вероятности $P(A|C)$ и $P(B|C)$ были одинаковыми?

Предположим, что деньги в конвертах появляются следующим образом: «ведущий» выбирает случайное число x из некоторого заданного распределения $g(x)$ и кладёт сумму x в один конверт, а сумму $10x$ — в другой. Будем считать, что распределение $g(x)$ дискретно (рассмотрение непрерывных распределений усложнит рассуждения, но ситуацию принципиально не изменит).

Сумма x в левом конверте может оказаться в двух случаях: либо было выбрано случайное число x и эта сумма помещена в левый конверт, либо было выбрано случайное число $\frac{x}{10}$ и эта сумма помещена в правый конверт (а в левый попала увеличенная). Полная вероятность этого события равна

$$P(C) = \frac{1}{2} \left(g(x) + g\left(\frac{x}{10}\right) \right).$$

Находим

$$P(A|C) = \frac{P(AB)}{P(C)} = \frac{g\left(\frac{x}{10}\right)}{2P(C)}, \quad P(B|C) = \frac{g(x)}{2P(C)}.$$

Получаем, что если $P(A|C) = P(B|C)$, то $g(x) = g\left(\frac{x}{10}\right)$.

Из последнего равенства по индукции получаем, что $g(x) = g(x \cdot 10^i)$ для всех целых i . Но это означает, что счётное множество значений случайной величины имеют равные ненулевые вероятности, а это невозможно.

Таким образом, причина парадокса заключается в том, что строится неполная вероятностная модель, для которой не существует корректной полной модели.

Любопытно, что в байесовском подходе иногда используют такие некорректные вероятностные модели, а именно в качестве априорного распределения берут функции, которые распределением не являются (интеграл расходится), например задают равномерное «распределение» на всей числовой оси. Такой «трюк», однако, вполне допустим, если удастся показать, что результат, получаемый с использованием некорректной модели, можно получить как предел последовательности решений, построенных по некоторой последовательности корректных моделей.

Казалось бы, с парадоксом разобрались, но не всё так просто.

Мы доказали, что не существует корректной вероятностной модели, в которой бы для правого конверта суммы $10x$ и $\frac{x}{10}$ были бы равновероятными. Однако для возникновения парадокса равенство этих вероятностей не обязательно. Достаточно, чтобы условное математическое ожидание для суммы в правом конверте было всегда больше, чем сумма, обнаруженная в левом.

Попробуем построить вероятностную модель с таким свойством.

Зададим распределение в виде геометрической прогрессии

$$g(10^i) = (1 - \alpha) \cdot \alpha^i, \quad i = 0, 1, \dots$$

Вычислим условное математическое ожидание для суммы в правом конверте

$$E_{y|xy} = \frac{\frac{x}{10} \cdot g\left(\frac{x}{10}\right) + 10x \cdot g(x)}{2P(C)} = \frac{10^i}{\alpha + 1} \cdot \left(10\alpha + \frac{1}{10}\right)$$

при $x = 10^i$, $i > 0$.

Видим, что условие $x < E_{y|xy}$ эквивалентно

$$10^i < \frac{10^i}{\alpha + 1} \cdot \left(10\alpha + \frac{1}{10}\right),$$

откуда получаем $\alpha > 0,1$.

Таким образом, при $\alpha > 0,1$ условное математическое ожидание выигрыша в правом конверте всегда больше суммы, обнаруженной в левом конверте (для $x = 1$, т. е. $i = 0$, это тоже верно).

Но конверты в рассмотренной модели остались равноправными (распределения денежных сумм для них одинаковы), поэтому аналогично можно показать, что условное математическое ожидание выигрыша в левом конверте всегда больше суммы, находящейся в правом конверте.

Получаем противоречие (парадокс «вернулся»).

Вычислим безусловное математическое ожидание выигрыша

$$E_x = \sum_{j=0}^{\infty} 10^j \cdot (1 - \alpha) \cdot \alpha^j = (1 - \alpha) \cdot \sum_{j=0}^{\infty} (10\alpha)^j.$$

При $\alpha \geq 0,1$ ряд расходится, и математическое ожидание, строго говоря, не существует (по соглашению, можно говорить о бесконечном математическом ожидании).

Утверждение 2 Пусть $x < E_{y|x}y$ при всех x и $y < E_{x|y}x$ при всех y . Тогда E_y и E_x не существуют (бесконечны).

Докажем от противного. Если E_y существует, то имеет место $E_{yy} = E_x E_{y|x}y > E_x x$. Это значит, что E_y тоже существует, поэтому $E_x x = E_y E_{x|y}x > E_y y$.

То, что условие в утверждении выполнимо, мы уже установили выше.

Из приведённого примера можно сделать вывод, что если безусловное математическое ожидание бесконечно (не существует), то и условное математическое ожидание (функция регрессии) не имеет содержательного смысла (даже если математически определено и конечно).

Все остальные «парадоксы» теории вероятностей являются кажущимися, т. е. не содержат реального противоречия, а всего лишь отражают факты, не согласующиеся с интуицией некоторых людей.

Однако парадокс конвертов представляется истинным, в том смысле, что для его устранения требуется установить некоторые запреты.

Во-первых, нельзя строить частные (неполные) вероятностные модели, для которых не существует корректной полной модели.

Во-вторых, нельзя содержательно интерпретировать регрессионную модель, если безусловное математическое ожидание целевой случайной величины не существует (бесконечно).

К слову, самым известным истинным парадоксом является, очевидно, парадокс Рассела, который «запрещает» рассматривать множества, включающие себя в качестве элемента.

Несмотря на эвристическую природу такой интерпретации, использовать вероятность как «меры незнания» всё же допустимо. В частности, на этом основан байесовский подход (в статистике и анализе данных). И этот подход успешно применяется. Но его использование требует осторожности, поскольку легко получить некорректный результат.

Здесь стоит упомянуть очень интересный результат [14], показывающий, что вероятностную модель на основе байесовского подхода часто можно заменить подходящей функцией регуляризации, причём такой подход оказывается более простым и гибким.

Второй из «нетривиальных» парадоксов теории вероятностей — парадокс «спящей красавицы». На самом деле, это не парадокс, а пример, показывающий, что вероятность может быть относительна (зависит от наблюдателя). Но это не отменяет её объективность.

Вместо классической формулировки приведём очень простой вариант, который сохраняет его смысл (но полностью устраняет «интригу»).

Пример 3 Пусть мы исследуем выпадение герба при бросании монеты, но опыт проводим не сами, а силами лаборанта, который бросает монету и присылает фото результата.

И по какой-то причине этот лаборант гербы фотографирует дважды, а решки — один раз. На основе полученных фотографий мы наблюдаем частоту герба $\frac{2}{3}$.

Вопрос, какую вероятность выпадения герба считать правильной: $\frac{1}{2}$ или $\frac{2}{3}$?

Ответ очевиден: вероятность выпадения герба равна $\frac{1}{2}$, а вероятность получения фотографии герба равна $\frac{2}{3}$.

Классическая формулировка парадокса спящей красавицы более сложная для анализа, но суть примерно такая же: в ряде задач мы можем вводить разные вероятностные пространства и получать разные вероятности, которые все имеют смысл.

В классической формулировке «парадокс» возникает из-за того, что кажется, будто одно и то же событие имеет разные вероятности, но на самом деле события разные (хотя выглядят как одно и то же).

9.4.2 Байесовский подход

В учебнике [30] есть раздел 5.5.4, названный «о вреде р-значений», где говорится дословно следующее.

Величину p -value¹ часто интерпретируют как правдоподобие данных в предположении справедливости нулевой гипотезы, считая, что малые значения якобы означают, что H_0 маловероятна, а H_1 , следовательно, вероятна. При этом рассуждение строится так.

Если H_0 истинна, то такая критериальная статистика, скорее всего, не имела бы места. Но она имеет место. Следовательно, H_0 , скорее всего, ложна.

Если человек обладает минимальной грамотностью в математической статистике, то он так рассуждать никогда не будет. То, что критерий отверг гипотезу, ещё не подразумевает, что она, «скорее всего, ложная»².

¹Термин p -value следует переводить как «достижимый уровень значимости».

²Во-первых, то, что критерий отверг гипотезу, ещё не значит, что мы её отвергли. Мы можем понять, что критерий плохой, и отказаться его использовать. Во-вторых, даже если мы отвергли гипотезу, это значит только то, что мы её отвергли, а не то, что она, «скорее всего, ложная». Сочетание «скорее всего» намекает на интуитивную вероятностную оценку, но мы не оцениваем вероятности гипотез.

Продолжим цитату.

Однако это рассуждение неверно. Чтобы понять, почему, рассмотрим следующий пример.

Если человек американец, то, скорее всего, он не является членом Конгресса. Этот человек член Конгресса. Следовательно, он, скорее всего, не американец.

Этот пример призван опровергнуть рассуждения, которые статистики и так никогда не делают.

Но приведём пример корректных рассуждений.

Введём гипотезы H_0 – человек является американцем, H_1 – человек не является американцем, а также событие A – человек является членом Конгресса.

Имеем ли мы право использовать A в роли критического множества для гипотезы H_0 ? Да, критическим множеством может быть любое событие, поэтому мы имеем право рассмотреть критерий, который отвергает H_0 при наступлении A .

Согласно Википедии, Конгресс США насчитывает 535 членов, а население страны составляет порядка 340 млн человек. Отсюда уровень критерия (вероятность ошибки 1-го рода, или p-value) равна $1,57 \cdot 10^{-6}$.

При этом вероятность ошибки 2-го рода равна 1, поскольку неамериканец не может быть членом Конгресса. Иными словами, мощность критерия нулевая. На основе критериев нулевой мощности нельзя сделать никаких выводов.

Таким образом, мы вправе такой критерий рассмотреть, но он бесполезен.

И это никак не свидетельствует о вреде такого понятия как p-value.

Более того, p-value уместно использовать только для критериев согласия (когда мощность не определена), а в случае гипотезы против простой альтернативы корректнее говорить об ошибках 1-го и 2-го рода.

В [30] далее предлагается в рассматриваемом примере использовать формулу Байеса, и в результате выкладок делается вывод,

что член Конгресса с вероятностью 1 является американцем. Но очевидно, что для такого вывода формула Байеса не нужна совершенно.

В результате пример, призванный проиллюстрировать необходимость байесовского подхода, оказался примером задачи, где он полностью лишний.

По сути, вся аргументация в пользу байесовского подхода против классических методов статистики (в частности, доверительных интервалов) в [30] построена на примерах явно некорректного³ применения последних. Например, на с. 215 (подраздел 4.7.5) говорится фактически следующее: если для построения доверительного интервала использовать аппроксимацию биномиального распределения нормальным, то в случае одного испытания (при выборке объёма 1) такой интервал будет плохим. Из этого делается заключение, что доверительные интервалы опасны и лучше использовать байесовский подход.

На самом же деле, если использовать нормальное приближение при $N = 1$, то байесовский подход даст столь же бессмысленный результат. Но почему-то автор [30] байесовский подход применяет без некорректного приближения, а доверительный интервал строит на основе такого приближения.

При этом для биномиального распределения несложно построить точный доверительный интервал (что мы уже делали), который выглядит корректно при любых N .

Разберём ещё один поучительный пример из той же работы.

Рассматривается параметрическое семейство дискретных рас-

³Ещё более некорректной выглядит попытка наклеивания ярлыков, а именно: автор упомянутой работы называет всех, кто не байесовец (Bayesian), словом frequentist (в англоязычной версии книги), которое Яндекс и Гугл стесняются переводить. Заметим, что байесовский подход заключается в сведении задач статистики к обычным вероятностным задачам и этим фактически убирает статистическую специфику. С учётом этого тех, кто не делает подобное сведение, логичнее называть просто статистиками. Но ещё логичнее будет не провоцировать холивар и не делить учёных на приверженцев того или иного подхода. Любой подход — всего лишь инструмент, и грамотный исследователь выбирает инструменты из задачи, а не из личных предпочтений.

пределений с параметром θ , в которых случайная величина X равновероятно принимает значения θ и $\theta + 1$.

Для выборки $S = (x_1, \dots, x_N)$ используем статистику $m(S) = \min_i x_i$ и построим доверительный интервал

$$U(S) = [m(S), m(S)].$$

Легко вычислить, что

$$P(\theta \in U(S)) = 1 - 2^{-N}.$$

Например, при $N = 2$ доверительная вероятность для такого интервала составит 0,75.

Если, например, $S = (40, 39)$, то $U(S) = [39, 39]$.

Теперь в [30] говорится буквально следующее: «мы точно знаем, что θ должно быть равно 39, но „уверенность“ в этом факте составляет всего 75 %».

Действительно, так и есть. Но давайте построим байесовский интервал при тех же ограничениях, а именно используя не всю информацию из выборки, а только статистику $m(S)$.

Возьмём в качестве априорного распределения равномерное псевдораспределение (это обычный выбор при байесовском подходе). Получаем

$$P(\theta = 39 | m(S) = 39) = 0,75.$$

Выходит, что при «честном сравнении» в равных условиях байесовский подход даёт тот же результат. Различие только в способе задания вероятностных пространств и, соответственно, в интерпретации этого результата.

Если использовать всю информацию из выборки, то

$$P(\theta = 39 | S = (40, 39)) = 1.$$

Байесовский подход «работает».

Но раз мы «разрешили» использовать выборку как угодно, то давайте построим другой доверительный интервал. Пусть $\hat{m}(S) = \max_i x_i$ и

$$U'(S) = [\hat{m}(S) - 1, m(S)].$$

Очевидно, что

$$P(\theta \in U'(S)) = 1.$$

Если $S = (40, 39)$, то $U'(S) = [39, 39]$ и доверительная вероятность равна 1. Оказывается, подход с доверительными интервалами тоже «работает».

Рассмотрим пример, иллюстрирующий отличие доверительных интервалов от байесовских интервалов.

Пусть мы сделали робота, который ловит сачком бабочек. Вероятность поймать пролетающую бабочку равна 0,9.

Предположим, что бабочка пролетела, и мы видим только опущенный сачок. С какой вероятностью бабочка в сачке? Очевидно, тоже 0,9.

Обведём теперь сачок по краю мелом. Какова вероятность, что бабочка находится внутри мелового контура?

Неизвестно. Более того, говорить о такой вероятности некорректно (мы не предполагали, что бабочка движется случайным образом). Но можно сказать, что доверительная вероятность (для нахождения бабочки в меловом контуре) равна 0,9. Хотя фактически это вероятность нахождения бабочки в сачке. Отличие сачка от контура в том, что сачок не привязан к местоположению.

Байесовский подход вводит априорное распределение на возможных положениях бабочки. И тогда уже можно найти вероятность того, что бабочка в контуре (эта вероятность может оказаться отличной от 0,9).

Общий вывод можно сделать такой, что байесовский подход является безусловно полезным инструментом. И пожалуй, он проще в понимании и для освоения, чем методы, его не использующие.

Но категорически некорректно будет утверждать, что байесовский подход «спасает» математическую статистику от каких-то «нелогичностей».

Во многих случаях результаты, полученные байесовским подходом, оказываются близки к результатам, полученным без его использования. Так, выражение для доверительного интервала

для параметра среднего нормального распределения при известной дисперсии полностью совпадает с выражением для байесовского интервала (хотя их интерпретация разная). Для параметра биномиального распределения мы строили интервалы в подразделе 6.2.2 и результаты подходов получились различными, хотя и очень близкими.

Рассмотрим ещё один показательный пример.

Пример 4 Пусть мы подбросили монету 12 раз и наблюдали 3 выпадения герба. Следует ли отвергнуть гипотезу о симметричности монеты, т. е. о том, что вероятности выпадения герба для имеющейся у нас монеты равна 0,5?

В качестве критического множества можем взять событие, что герб выпал не более трёх раз. Вероятность такого события

$$P(m \leq 3) = \sum_{m=0}^3 C_{12}^m \cdot 0,5^{12} = 0,073.$$

Иными словами, гипотезу можем отвергнуть с достигнутым уровнем значимости 0,073.

Здесь мы по умолчанию считали, что число бросаний мы фиксировали заранее. Но что если мы делали броски до достижения заданного количества гербов?

Тогда случайно не число гербов, а число бросков, и в качестве критического множества разумно взять событие, что бросков потребовалось не менее 12. Такая вероятность вычисляется на основе отрицательного биномиального распределения

$$P(n \geq 12) = \sum_{n=12}^{\infty} C_{n-1}^2 \cdot 0,5^n = 0,0327.$$

Эмпирический факт тот же самый (12 бросков и 3 выпавших герба), но достигнутый уровень значимости (p-value) изменился. Если взять пороговый уровень значимости 0,05, то первый критерий не отвергнет гипотезу о симметричности монеты, а второй — отвергнет.

На первый взгляд ситуация кажется ненормальной. Но заметим следующее. Вероятность получить ровно три герба из 12 бросков равна $P(m = 3) = C_{12}^3 \cdot 0,5^{12} = 0,0537$, а вероятность, что потребуется 12 бросков для получения трёх гербов, составляет $P(n = 12) = C_{11}^2 \cdot 0,5^{12} = 0,0134$. Получается, что события всё же разные, хотя и там, и там 12 бросков и три герба. И даже понятно, чем они разные: во втором случае в последнем броске обязательно герб, а в первом случае результат последнего броска может быть любым.

Таким образом, эксперименты разные и результат объективно не идентичен, поэтому различие в полученном p -value как раз нормально.

А вот байесовский подход в обоих случаях даст (при равномерном априорном) одно и то же апостериорное распределение на p , а именно бета-распределение с параметрами $3 + 1, 12 - 3 + 1$. Для этого распределения можем вычислить

$$P(p \leq 0,5 \mid n = 12, m = 3) = 0,046.$$

Эту величину можем сравнить с полученными ранее p -value. Видим, что значения близки (для уровня значимости отличие в два раза несущественно), несмотря на принципиально разную интерпретацию.

Теперь может показаться, что как раз байесовский подход даёт «неправильный» результат, поскольку не учитывает отличие в экспериментах.

Но предположим, что монета нам даётся действительно случайная (наугад выбирается из некоторого множества), и вероятность герба — объективно случайная величина с известным априорным распределением. В этом случае мы «просто обязаны» использовать байесовский подход... Даже не так. В этом случае задача уже не является задачей статистики, а становится типовой задачей теории вероятностей. И байесовского подхода тут уже нет, поскольку смысл этого подхода в том, чтобы положить случайным параметр, который по условию задачи не случаен. А если параметр по условию случаен и его распределение задано в усло-

вии задачи, то мы просто используем формулу Байеса (но это не байесовский подход).

И применив формулу Байеса к данной уже вероятностной задаче, мы получаем апостериорное распределение на p , которое не зависит от того, бросаем мы монету заданное число раз или до заданного числа успехов.

Выходит, что статистический подход всё же имеет «конструктивные дефекты», раз выдаёт разные p -value в рассмотренном примере? Нет, конечно. Статистический подход выдаёт корректный результат, просто другой по смыслу. Он более сложен в интерпретации, но это плата за то, чтобы не использовать эвристических предположений.

Напоследок заметим, что

$$P(p \leq 0,5 | n \geq 12, m = 3) \neq P(p \leq 0,5 | n = 12, m \leq 3),$$

т. е. байесовским подходом тоже можно получить разные результаты (даже при одинаковом априорном распределении).

9.5 Применимость вероятностных моделей

Автор не встречал задач анализа данных, в которых вероятностные модели были бы неприменимы. Но это не значит, что другие модели неопределённости не могут дать в каких-то задачах лучший результат.

Позиция автора состоит в том, чтобы не устанавливать формальных запретов и ограничений на применение вероятностных моделей, а оценивать полученный результат.

Рассмотрим некоторое количество примеров, в которых применимость вероятностных моделей неочевидна.

9.5.1 Статистическая устойчивость

Часто необходимым условием для применения статистических методов называют статистическую устойчивость данных.

Это неоправданное сужение области применимости.

Пример 5 *Зададим случайный процесс в виде следующего алгоритма.*

Алгоритм генерирует последовательность из 0 и 1. При этом вероятность 1 равна p , которая в свою очередь принимает значения 0,7 и 0,3 по следующему правилу.

Начальное значение p равно 0,7. Далее на каждом шаге вычисляется текущее среднее значений последовательности. Если это среднее оказывается больше 0,6, то p устанавливается равным 0,3, если же текущее среднее меньше 0,4, то p устанавливается равным 0,7.

Данная последовательность является статистически неустойчивой, в том смысле, что кумулятивное среднее не имеет предела. Легко понять, что оно совершает бесконечное число колебаний между 0,6 и 0,4, достигая обеих границ.

Вместе с тем странно будет утверждать, что её нельзя («запрещено») анализировать средствами математической статистики — ведь она задана чисто вероятностной моделью.

Частота в примере не имеет предела, но вероятности определены.

«Повторяемость» — не самостоятельный закон, а следствие закона больших чисел и «объективности» вероятности.

Если мы пока не умеем повторить эксперимент, это не значит, что нельзя применять вероятность.

Бросаем хрустальную монету на бетонный пол. Повторить не сможем, но вероятность герба определена.

Пример 6 *Требуется оценить вероятность падения крупного метеорита на Землю.*

Можно по-разному понимать слово «крупного». Для определённости будем считать, что речь о метеорите, который крупнее всех, падение которых было достоверно установлено в истории Земли.

Как оценить искомую вероятность, если искомые события не наблюдались? Способ достаточно очевиден. Мелкие метеориты

падают часто, и мы можем очень точно оценить для них вероятность. Точнее говоря, мы полагаем (эта гипотеза не опровергается данными), что падение метеоритов подчиняется пуассоновскому процессу, и мы оцениваем параметр этого процесса для метеоритов разного размера.

Можем построить зависимость интенсивности падений метеоритов от их массы. Далее можно экстраполировать полученную зависимость на метеориты большой массы.

Конечно, любая экстраполяция вызывает сомнение и требует обоснования. Здесь можно привлечь дополнительную информацию, а именно: построить распределение астероидов (и комет) по массе. Если удастся связать встречаемость астероидов определённой массы с частотой падения метеоритов той же массы, то можно уточнить искомую зависимость.

Пример 7 *Требуется оценить вероятность новой мировой войны.*

К тому, что определяется волей человека, вероятность неприменима. Сложно сказать, в какой мере подобное событие определяется волей политических лидеров, а в какой — объективными факторами.

Для анализа объективных предпосылок, скорее всего, возможно построить какие-то вероятностные модели, но вряд ли у них будет хорошее качество (сейчас приемлемых моделей нет). Поэтому данный пример вполне может считаться случаем, когда вероятностные модели «не работают». Но заметим, что здесь вообще никакие модели не работают.

Пример 8 *Требуется оценить вероятность посещения Земли инопланетянами в ближайший год.*

Задача, конечно, шуточная, но показательная как пример того, что отсутствие событий не означает отсутствие данных.

Будем считать, что прилёты инопланетян на Землю подчиняются пуассоновскому процессу. В качестве параметра возьмём β — среднее количество посещений за год.

Тогда вероятность того, что за год будет хотя бы одно посещение, равна $1 - e^{-\beta} \approx \beta$.

Вероятность того, что за N лет не произойдёт ни одного посещения равна $e^{-N\beta}$. Приравняем её к некоторому уровню значимости α . Имеем $\beta = -\frac{1}{N} \ln \alpha$.

Выберем уровень значимости $\alpha = 0,01$ и предположим, что за последние $N = 200$ лет посещений не было. Получаем для β доверительный интервал $[0, 0,023]$.

Если взять большие значения N , то доверительный интервал сужится.

Здесь мы имеем и вероятность, и доверительную вероятность (которая есть $1 - \alpha$), что не очень удобно для интерпретации. Разумный вариант — приравнять α и β . Тогда получим $N = -\frac{1}{\beta} \ln \beta$.

Из этой формулы можем, например, получить, что при уровне значимости 0,001 вероятность встретить инопланетян в ближайший год не превосходит 0,001, если мы знаем, что они нас не посещали в предыдущие 6908 лет.

Хотя задача шуточная, метод её решения может всерьёз применяться для оценки вероятности событий, которые никогда не наблюдались. Конечно, при этом используется сильное предположение об однородности случайного процесса, что требует обоснования.

Пример 9 *Жил-был моряк. Он хотел детей, но не решил, одного или двух, поэтому кинул монету: орел — будет один ребенок, решка — двое. Неизвестно, как выпала монета, но Вы — его ребенок. Какова вероятность, что Вы — его единственный ребенок?*

Пусть в мире существует N человек, не считая детей моряка, среди которых может оказаться тот, к кому обращён вопрос. Обозначим гипотезы H_1 — выпал орёл, H_2 — решка. Событие A — заданный человек (к кому обращён вопрос задачи) является ребёнком моряка. Требуется найти $P(H_1/A)$.

Имеем: $P(A/H_1) = \frac{1}{N+1}$, $P(A/H_2) = \frac{2}{N+2}$.

По формуле полной вероятности находим $P(A) = \frac{3N+4}{2(N+1)(N+2)}$.

Наконец, по формуле Байеса получаем ответ: $P(H_1/A) = \frac{N+2}{3N+4}$.

В условии не дано N . Если мы предположим, что N очень большое, то $P(H_1/A) \approx \frac{1}{3}$.

Однако можно привести доводы за то, чтобы положить $N = 0$. Тогда $P(H_1/A) = \frac{1}{2}$.

Мы эти доводы обсуждать не будем (они совсем не похожи на научные) и ограничимся выводом, что в данной задаче неполное условие, которое необходимо дополнить информацией, из которой можно установить значение N . Парадокса нет, есть недостаток информации.

Автор должен признать, что в этом примере применение вероятностных методов происходит «на грани фола». Но эту задачу можно воспринимать просто как упражнение.

9.5.2 Проблема объективного сравнения подходов

Существуют задачи прогнозирования, в которых точность прогноза невозможно непосредственно вычислить.

Пример 10 *Требуется проанализировать данные по АЭС и оценить степень её безопасности. Некто по результатам анализа дал заключение, что реактор взорвётся в ближайший месяц с вероятностью 0,6.*

Возникает законный вопрос: почему 0,6, а не, например, 0,5. Ведь мы не можем провести опыт хотя бы с сотней реакторов, чтобы проверить эту вероятность.

Объяснение следующее. Во-первых, здесь совершенно без разницы: 0,6, 0,5, 0,1 или 0,01. Во всех случаях нужно реактор срочно останавливать.

Во-вторых, в данном случае оформление ответа в виде одного числа — это всего лишь способ побудить «заказчика» к правильным действиям с помощью понятных ему слов. На самом деле, реальным результатом анализа будет целый интервал вероятностей. И даже не интервал, а функция правдоподобия, по которой

можно построить доверительный интервал с любой доверительной вероятностью. А может быть, было построено множество таких функций правдоподобия для разных моделей и гипотез. Но это можно объяснить только специалисту, а для заказчика будет просто число или даже заключение в качественной шкале.

Теперь представим, что эту задачу независимо решают два коллектива. В результате первый даёт оценку, что вероятность аварии в течение каждого месяца составляет 0,01, а второй делает заключение, что повода для тревоги нет. Допустим, прислушиваются к мнению второго коллектива, реактор оставляют работать как есть, и 10 лет ничего не происходит. Как понять, кто прав? Без привлечения сторонних экспертов — никак. А эксперты — это субъективность. По большому счёту решение принимается на основе репутации исполнителей.

Объективное сравнение моделей возможно только на тестовой выборке. Большое количество задач с объективным критерием качества доступно на [kaggle.com](https://www.kaggle.com), но понятно, что там не все типы задач представлены (как раз потому, что не всегда есть объективный критерий).

9.5.3 «Измерение» вероятности

Некоторые исследователи ставят вопрос об «измерении» вероятности, хотя слово «измерение» в применении к вероятностям, пожалуй, неуместно.

Существует два способа определения вероятности: вычисление и статистическое оценивание. Как «измерить» вероятность выпадения 6 на кости? Пересчитать грани.

В ряде задач вероятность вычисляется комбинаторными методами. Иногда её можно вычислить на основе других измеренных величин. Например, в квантовой механике некоторые вероятности можно вычислить, зная энергию.

Когда проводят измерения обычных физических величин, в результате получают по сути доверительный интервал (с учётом случайных погрешностей).

С помощью статистики (оценка параметров) мы можем получить доверительный интервал для вероятности. В этом смысле имеем измерение.

Но результат такого измерения вероятности всё же труднее интерпретировать, поскольку мы имеем как саму вероятность, так и доверительную вероятность. Например, при оценивании параметра в схеме Бернулли можем получить результат вида: при доверительной вероятности 0,95 имеем доверительный интервал для вероятности события $[0,8, 0,85]$. А в привычных ситуациях физических измерений обычно доверительная вероятность соответствует уровню практической достоверности, что нереально (бессмысленно) в частотных оценках. Получается, что при измерении «обычных» величин мы можем «забыть» про доверительную вероятность, а при «измерении» вероятности наличие доверительной вероятности принципиально.

Иногда предлагается использовать совместно с вероятностью идеи нечёткой логики. Подход, безусловно, заслуживает внимания, но стоит отметить, что при классическом статистическом оценивании вероятности мы и так получаем интервал её значений (и даже семейство таких интервалов). Это примерно такой же «уровень неопределённости», что и в указанной работе. Отличие только в семантике этой неопределённости (степень принадлежности против доверительной вероятности).

В целом вопрос, является ли вероятность физической величиной, кажется искусственным. Формально теория вероятностей не относится к физике, а значит, вероятность — не физическая величина. Но от этого она не становится менее объективной характеристикой и в этом смысле вполне является физической величиной.

Минимально необходимый объём выборки для статистических оценок равен 1.

Если проведён 1 эксперимент и случился успех, то мы можем сказать, что вероятность успеха больше уровня практической невозможности. Да, информации немного, но больше, чем ничего.

Не имеет смысла устанавливать какие-то границы, например,

что для применения статистических методов требуется не менее 50 наблюдений. Почему 50 достаточно, а 49 — нет? При любом числе наблюдений мы можем построить доверительный интервал. Если он не устраивает, то нужны дополнительные данные или предположения. И другими (невероятностными) подходами в этом случае задачу тоже не решить.

Заметим также, что в задачах анализа данных (машинного обучения) «эффекты малой выборки» зачастую возникают даже при объёмах выборки в 1000 объектов и более.

9.6 Связанные понятия

9.6.1 Случайность и Колмогоровская сложность

Перейдём к обсуждению того, что считать случайным [9].

Пример 11 *Имеются следующие двоичные последовательности:*

- 1) 11111111111111111111,
- 2) 010101010101010101,
- 3) 1000010001110100011010111000111,
- 4) 0110111100101000101000011011110.

Какие из них являются случайными?

Кажется, что последовательности 1 и 2 неслучайны, а 3 и 4 — случайны.

Но чтобы ответить на вопрос, нужно дать определение случайной последовательности.

Определение 6 *Случайная последовательность — это последовательность, которая получена в результате стохастического эксперимента (является реализацией случайного процесса или случайной функцией).*

При таком определении не важно, как выглядит последовательность, важно лишь, как она получена. Однако если кто-то

скажет нам, что получил последовательность 1 или 2, подбрасывая монету (1 – герб, 0 – решка), то мы ему не поверим. И будем правы, скорее всего.

Но можно представить ситуацию, когда достаточно богатый и очень экстравагантный человек нанял 10000 испытателей, которым поручил под запись на камеру проводить серии по 20 бросаний монеты. Достаточно быстро он станет обладателем честного (без склеек) ролика, на котором монета выпадает гербом 20 раз. Получается, что в этом случае последовательность 1 придётся признать случайной, поскольку она действительно получена в эксперименте.

По поводу случайности последовательности 3 сомнений не возникает... пока не будет раскрыта информация, что она является двоичным представлением числа, которое в десятичной записи выглядит как 1111111111.

Аналогично, псевдослучайные последовательности от генераторов случайных чисел случайными никак не являются, но по их виду этого не понять. Более того, псевдослучайные числа можно использовать в качестве случайных при статистическом моделировании. При этом нужно сознавать, что в некоторых очень редких случаях такое моделирование может давать некорректный результат из-за неидеальности генераторов.

Если бы автор заявил, что последовательность 4 он получил, бросая монету, то опровергнуть это утверждение было бы невозможно. Известно, что последовательности, которые человек пишет «из головы», имитируя случайность, можно достаточно надёжно отличить от реальных повторных независимых испытаний, но для этого последовательность нужна намного длиннее.

Однако автор признался, что последовательность 4 он составил, имитируя случайность. Тем не менее, её действительно можно считать случайной. Это не были независимые испытания. Но это реализация случайного процесса (пожалуй, точнее будет термин случайная функция) неизвестного вида с неизвестными параметрами. И этот процесс можно исследовать (и такие исследования проводились).

Теперь если кто-то предъявит последовательность 4 и скажет, что получил её бросанием монеты, то мы ему поверим? Конечно, нет. Мы ведь будем (практически) точно знать, что он переписал эту последовательность из данного примера.

Итак, термин «случайная последовательность» в литературе используется в двух принципиально разных смыслах: как последовательность, полученная как реализация случайного процесса (стохастического эксперимента), и как последовательность, выглядящая случайной, т. е. не имеющая видимых закономерностей.

Это не разные интерпретации случайности, это содержательно разные понятия, которые не исключают друг друга, а дополняют.

Поскольку в одной работе нельзя один термин использовать в разных смыслах, мы будем под случайностью понимать первое понятие и называть *случайным* то, что имеет вероятность.

При этом второе понятие очень важно и формализуется через алгоритмическую (Колмогоровскую) сложность [17]. Его, пожалуй, можно было бы назвать степенью «хаотичности».

В ряде карточных игр есть правила, требующие делать передачу, если раздача оказалась «недостаточно случайной». Это как раз пример в духе алгоритмической сложности.

Алгоритмическую сложность можно использовать как статистику для формирования критерия согласия для проверки «гипотезы случайности».

Итак, случайность определяется способом получения последовательности, но не её видом. Хотя по виду последовательности можно проверять гипотезы относительно процесса её получения.

9.6.2 Детерминированность и случайность

В философии детерминированность противопоставляется случайности.

Если в эксперименте отсутствует объективная неопределённость, то использовать для него вероятностные модели не имеет смысла. Однако формально даже такие эксперименты можно описать в терминах вероятностей.

Пример 12 *Предположим, нас интересует вопрос, встретим ли мы, выйдя на улицу, динозавра (птиц к ним не относим).*

Можно рассуждать так, что поскольку динозавры вымерли, встретить их никак не сможем. Значит, детерминированно не встретим.

Но можно ввести множество из двух исходов: «встретим» и «не встретим». Поскольку динозавры вымерли, вероятность исхода «встретим» равна нулю. Хотя формально это событие не является невозможным (поскольку не есть \emptyset).

Данный пример иллюстрирует возможность вероятностного описания детерминированного эксперимента, при этом демонстрирует ситуацию, когда применение вероятностей неуместно. Но неуместно — не значит нельзя.

Таким образом, нет оснований «запрещать» применение вероятностей к любым задачам, однако существуют задачи, в которых применение вероятностей нецелесообразно.

В теории вероятностей основными понятиями являются событие и вероятность, и можно вообще обойтись без использования слов «случайный» и «детерминированный». Однако эти слова очень привычны, поэтому отказываться от их использования нецелесообразно, но нужно придать им точный формальный смысл.

Можно было бы назвать случайным всё, что мы описываем вероятностью. Но тогда случайными станут все события. Так, пожалуй, можно сделать, поскольку вопрос чисто терминологический. Однако более привычным будет назвать детерминированными события, имеющие вероятность 0 или 1, остальные — случайными. При этом требуется выполнение дополнительных условий: объединение всех детерминированных событий, имеющих вероятность 0, также имеет нулевую вероятность, а пересечение всех детерминированных событий, имеющих вероятность 1, также имеет вероятность 1.

Но на практике граница между случайным и детерминированным размыта. Практически достоверное событие логично считать и практически детерминированным.

Заметим, что как противоположность случайности можно рассматривать не только детерминированность, но и волю [46].

Когда человек делает какой-то выбор, мы имеем ситуацию неопределённости (если постулировать свободу воли и не считать, что все поступки человека predetermined). Однако эта неопределённость не является случайностью. Получается, что вероятность для описания действий, определяемых волей человека, неприменима.

С другой стороны, наука психология (как и социология) вполне успешно применяет вероятностные и статистические модели как раз для описания поведения людей. Видимо, здесь дело в том, что для этих наук человек — не субъект, а объект. Пожалуй, здесь мы сталкиваемся с непростым философским вопросом, обсуждение которого выходит за рамки предмета.

Иногда задаются вопросом, является ли вероятность объективным понятием.

Это сугубо философский вопрос, ответ на который никак не влияет на область практической применимости и обоснованность вероятностного подхода.

Точно так же можно задаться вопросом, объективно ли пространство или же оно есть только способ нашего восприятия реальности. Разные философские школы отвечают на этот вопрос по-разному. Но это никак не влияет на применимость геометрии к практическим задачам из реального мира.

9.7 Обсуждение

9.7.1 Выводы

Очень часто непродуктивные споры возникают из-за различий в терминологии.

Часть путаницы вносит, например, «классическое определение вероятности». Лучше не называть это определением, поскольку определение для вероятности принято аксиоматическое, и не может в рамках одной теории быть несколько определений одного и

того же. Более корректно назвать это способом вычисления вероятности. То же самое относится к геометрическим вероятностям.

Частотное определение вероятности — очень интересный и продуктивный в плане математических результатов подход. Но он несёт опасность вульгарных интерпретаций, когда математическое понятие «определяется» как результат физического эксперимента (похожее имело место при зарождении геометрии, когда точку определяли как «очень маленькое пятно»). Ещё хуже, когда вероятность определяется через сходимости по вероятности. Читателю рекомендуется изучать частотный подход уже после достижения хорошего понимания вероятностей (в рамках аксиоматического определения).

Частотная интерпретация вероятности.

Чтобы преодолеть путаницу, которая часто сопровождает использование этого понятия, следует вспомнить исторический контекст его появления. Во времена становления современной аксиоматической теории вероятностей в философии господствовало представление, будто любая случайность — это следствие неполноты знаний. В контексте этого представления широко была распространена интерпретация вероятности как некоторой субъективной меры неопределённости, связанной с незнанием.

В противовес этому существовало мнение, что вероятность — это нечто, существующее объективно. Это мнение и называлось частотной интерпретацией. Таким образом, частотная интерпретация вероятности — это по сути синоним утверждению о том, что вероятность объективна. А доказательством этой объективности было то, что теоретические предсказания подтверждаются в эксперименте. Здесь теоретическое предсказание — это закон больших чисел как математически доказанное утверждение. А экспериментальное подтверждение — то, что в опытах (по схеме Бернулли) частота практически достоверно сходится к вероятности.

Можно привести аналогию с законом всемирного тяготения. Этот закон позволил предсказать движение планет. То, что предсказания совпали с наблюдаемым движением, явилось важней-

шим доказательством справедливости закона.

Кстати сейчас никому не приходит в голову утверждать, что если в быту притяжение предметов друг к другу нельзя заметить, то этого притяжения не существует. Однако утверждения, что если эксперимент невозможно повторить, то и вероятность в нём «не существует», вполне встречаются.

Из сказанного следует, что желательно с осторожностью говорить о частотной интерпретации вероятности, поскольку здесь легко смешать математические и эмпирические факты. К тому же странно эмпирический факт называть интерпретацией чего-то. Предлагается интерпретировать вероятность как объективное основание для рационального выбора в условиях неопределённости.

Вероятность — фундаментальное понятие, наряду с пространством и временем.

Требовать повторяемости для стохастического эксперимента не нужно. Любой опыт можно повторить мысленно. Если же требовать реальной возможности повторить эксперимент, то она зависит от развития науки и техники. Получается, что применимость математической модели определяется техническим уровнем, что странно.

Генеральная совокупность почти всегда почти полностью состоит из воображаемых объектов. Поэтому мысленный эксперимент — обычный приём для вероятностей.

Чаще всего используют определение в духе следующего: «Теория вероятностей — раздел математики, посвященный исследованию закономерностей, возникающих в длинных сериях однородных стохастических экспериментов» [5]. Утверждение, разумеется, верное, но формулировка провоцирует заблуждение, будто вероятность применима только к таким экспериментам. Но её можно (с переменным успехом) применять и к неоднородным, и к неповторяющимся ситуациям.

Возможная формулировка: «Теория вероятностей — раздел математики, посвященный исследованию свойств вероятности, которая является одним из фундаментальных понятий в рациональ-

ной картине мира».

Понятие случайности. Случайно — то, что имеет вероятность, не путать с «беспорядочностью».

9.7.2 Замечания

В математическом аспекте теории вероятностей относительно недавно произошёл переход от эмпирического определения вероятности к аксиоматическому. Однако содержательная интерпретация (в большинстве источников) осталась прежней. Актуальным представляется приведение её в соответствие.

Область применимости теории вероятностей не ограничивается повторяющимися событиями. Она применима, когда в условиях неопределённости существует рациональный выбор.

При решении практических задач целесообразно использовать не формальные «правила применимости» (которые все небесспорны), а объективные критерии качества решения. Если в соответствии с «правилами» вероятность применять нельзя, но её применение даёт практически полезный результат — значит, применять её можно и нужно.

Обсудим вопрос, всегда ли можно использовать вероятностную интерпретацию машинного обучения.

Часто в учебниках говорится, что теория вероятностей изучает эксперименты, которые можно провести многократно в одинаковых условиях.

Однако в первом классе школы аналогичным образом говорится, что вычитать можно только из большего меньшее, а в старших классах говорится, что корень можно извлекать лишь из неотрицательных чисел.

Дальнейшее знакомство с математикой подталкивает к мысли, что в этой дисциплине всё, что нельзя, то на самом деле можно.

Точно так же и теория вероятностей далеко ушла от ограничений, при которых изначально создавалась. Так, случайные процессы вполне способны описывать эксперименты, условия которых ни разу не повторяются.

Вопрос сводится к тому, можно ли считать имеющиеся данные случайной выборкой (не требуется независимость и однородность).

В машинном обучении существуют как вероятностная, так и невероятностные интерпретации. Выбор на усмотрение исследователя. Вероятностная картина мира является универсальной и непротиворечивой, но не единственной.

В данной главе присутствуют дискуссионные (или даже отчасти провокационные) утверждения. Одной из целей было показать, что в науке вполне могут существовать различные мнения. Это не касается математических утверждений (если теорема доказана, то с нею нельзя «не согласиться»).

Но когда речь идёт об интерпретации, практически невозможно доказать ошибочность какого-либо мнения. И по большому счёту нельзя даже ставить вопрос о том, кто прав.

Любопытно, что философы придумали «принцип фальсифицируемости» в качестве критерия научности. Но к самой философии этот критерий неприменим, хотя бы потому, что ни для одной из когда-либо созданных философских концепций не была доказана её ошибочность. Так, не представляется возможным бесспорно доказать, что Платон, Демокрит, Кант в чём-либо не правы.

Разве что использовать как метод установления правоты кулачный бой. Но это не конструктивно, даже древнегреческие философы его (насколько известно) не использовали (хотя соответствующие навыки, как правило, имели).

Однако даже такой путь выглядит, по крайней мере, честнее, чем «доказательство» «собственного» мнения путём апеллирования к авторитетам (через цитирование классиков).

Автор надеется, что существующее разнообразие концепций, подходов и научных школ в машинном обучении сохранится в будущем.

Глава 10

Некоторые задачи анализа данных

В данной главе будут обсуждаться разнообразные (и достаточно разнородные) темы: методы глобальной оптимизации (СПА, теорема о бесплатном завтраке), кластерный анализ, прогнозирование временных рядов, заполнение пропусков и др.

Большинство вопросов будут рассмотрены кратко (и даже схематично), хотя почти все эти темы заслуживают отдельных курсов. Однако невозможно в одной книге изложить весь важный материал, поэтому где-то приходится ограничиваться обзором.

10.1 Методы глобальной оптимизации

Методы оптимизации — отдельная область математики, которая имеет определённую связь с анализом данных (неспроста на kaggle проводились конкурсы по решению оптимизационных задач).

В первую очередь, эта связь заключается в том, что большинство методов анализа данных основаны на решении оптимизационных задач и используют методы оптимизации.

Однако в некоторых постановках сама задача оптимизации

оказывается по духу ближе к области машинного обучения, чем к области математического программирования. Дело в том, что задачи математического программирования предполагают выполнение некоторых строгих ограничений на целевую функцию (например, гладкость или выпуклость). В отсутствие строгих предположений классические методы оптимизации неприменимы.

10.1.1 Задача поиска глобального экстремума

Под экстремумом понимается минимальное или максимальное значение функции. Без ограничения общности можно считать, что всегда ищется максимум, поскольку, если требуется найти минимум, то можно вместо исходной функции рассматривать функцию, умноженную на -1 , у которой максимум достигается в точке минимума исходной функции.

О поиске глобального экстремума имеет смысл говорить, когда функция является многоэкстремальной, т. е. имеет более одного локального экстремума. Локальным максимумом называется точка, в которой значение функции больше, чем в любой точке некоторой её окрестности.

Одним из наиболее наглядных примеров многоэкстремальной функции является рельеф местности, т. е. зависимость высоты над уровнем моря от географических координат точки.

Задача поиска глобального экстремума (или задача оптимизации) заключается в следующем. Требуется выбрать последовательность точек в пространстве X так, чтобы среди значений функции в выбранных точках оказалось значение, близкое к максимальному. При этом выбор последующих точек зависит от значений функции в точках, выбранных ранее.

Другими словами, задана некоторая функция, которая скрыта. Исследователь может выбирать любые точки из области определения и узнавать значение функции в них. Задача заключается в том, чтобы, используя как можно меньшее число точек, обнаружить как можно большее значение функции.

Для решения задач многоэкстремальной оптимизации наибо-

лее часто используется подход, основанный на идее адаптивного поиска. Данная идея заключается в том, что точки выбираются в соответствии с некоторой мерой «перспективности» областей пространства X . При этом «перспективность» области тем выше, чем большие значения функции в ней уже обнаружены и чем «менее исследована» область, т. е. чем меньше точек в ней поставлено ранее.

Идеи адаптивного поиска также развиваются в эволюционных (в том числе генетических) алгоритмах, а также в алгоритмах типа муравьиной колонии.

Задача поиска глобального экстремума может рассматриваться как частный случай задачи планирования эксперимента. Задача планирования эксперимента — это особая модификация задачи восстановления зависимостей, когда таблица данных не задана изначально, а формируется при участии исследователя, выбирающего точки пространства X , для которых будут сообщены значения целевой функции. Требуется выбирать точки таким образом, чтобы как можно точнее оценить целевую функцию. Критерием качества такой оценки обычно выступает средний квадрат отклонения. Требование найти экстремум функции может считаться одним из вариантов критерия качества восстановления зависимости.

10.1.2 Формальная постановка

Сформулируем задачу глобальной оптимизации.

Пусть Ψ — некоторый класс функций с общими областью определения Z и областью значений R , т. е. $\psi : Z \rightarrow R$, $\psi \in \Psi$. Множество R упорядочено.

Алгоритмом (детерминированным алгоритмом) поиска экстремума будем называть множество (последовательность) функций

$$A = (A_i, i = 1, \dots, L), \quad A_i : (Z \times R)^{i-1} \rightarrow Z,$$

где параметр L определяет число точек, которые можно использовать при поиске экстремума. При этом A_1 — константа (из Z).

Результатом применения алгоритма A к функции ψ будет последовательность

$$A | \psi = ((z_i, r_i), i = 1, \dots, L),$$

где

$$z_i = A(z_1, r_1, \dots, z_{i-1}, r_{i-1}), \quad r_i = \psi(z_i).$$

Таким образом, первая точка фиксирована, а остальные ставятся в зависимости от полученных значений оптимизируемой функции.

Стохастическим (случайным) алгоритмом поиска экстремума будем называть совокупность семейств вероятностных мер

$$S = (P_i(dx | \alpha_i), i = 1, \dots, L), \quad \alpha_i \in (Z \times R)^{i-1}.$$

Все меры заданы на некоторой σ -алгебре подмножеств пространства Z .

Таким образом, первая точка выбирается в соответствии с фиксированным вероятностным распределением, для остальных вероятностная мера корректируется в зависимости от полученных значений оптимизируемой функции.

Чтобы задача поиска глобального экстремума была поставленной, нужно задать критерий качества алгоритма оптимизации, как функцию $K = \Psi \times \Lambda \rightarrow [1, \infty)$, где Λ – множество всех детерминированных и стохастических алгоритмов. В настоящее время такая задача решена только для некоторых частных постановок. Задача сравнения эффективности алгоритмов оптимизации и выбора алгоритма, наиболее подходящего для заданного класса задач, остается открытой.

10.1.3 Теорема о бесплатном завтраке

Теорема о бесплатном завтраке (или NFL, т.е. No Free Lunch theorem) является очень известным результатом. Приведем один из простейших её вариантов.

Пусть область определения Z и область значений R целевой функции являются конечными множествами, каждое из N элементов. Также положим, что целевая функция во всех точках принимает различные значения. Тогда общее число всех функций равно $N!$.

Критерий качества для результата оптимизации определим очень просто: будем считать оптимизацию успешной, если не более чем за M шагов найден глобальный экстремум.

Теорема. Для любого алгоритма оптимизации количество функций, которые он успешно оптимизирует, одинаково.

Иными словами, все алгоритмы равноценны, и самый «интеллектуальный» алгоритм оптимизации не лучше самого «глупого» (например, расстановки точек наугад).

Обычно данная теорема интерпретируется как утверждение, что если не делать вообще никаких предположений об оптимизируемой функции, то и разумного алгоритма оптимизации предложить нельзя.

Попробуем построить подобный пример для задачи машинного обучения.

Предположим, что у нас есть некоторое устройство, которое генерирует последовательность длины N , состоящую из 0 и 1. Всего возможно 2^N различных последовательностей. Требуется по первым M значениям предсказать следующее значение. Алгоритмом предсказания назовём функцию, которая каждой последовательности из M нулей и единиц сопоставляет 0 или 1.

Процесс предсказания выглядит следующим образом. Подаём на вход алгоритму первые M элементов последовательности и сравниваем его прогноз со следующим элементом. Затем в цикле сдвигаемся на один элемент, пока не пройдём всю последовательность, получив $N - M$ прогнозов.

Будем говорить, что алгоритм успешно предсказал последовательность, если все его прогнозы совпали с фактическими значениями.

Легко доказать, что для всех таких алгоритмов предсказания число последовательностей, которые они успешно предсказыва-

ют, будет одинаково (и равно 2^M). Это следует из того, что любой алгоритм для каждого начала последовательности может спрогнозировать единственное её продолжение. То есть из всех последовательностей, у которых на первых M позициях стоят заданные значения, только одна прогнозируется безошибочно.

Значит ли это, что все алгоритмы предсказания одинаковы по качеству?

Кажется, что так не должно быть. Ведь если к некоторому моменту мы наблюдаем, например, 90 единиц и 10 нулей, то разумным будет предсказать в качестве следующего значения 1, а не 0.

Получается, что в машинном обучении есть полный аналог NFL, но из него не следует равноценности любых методов прогнозирования.

Всё дело в том, что NFL не учитывает мощнейшего инструмента, который использует машинное обучение (иногда явно, но чаще неявно), а именно механизм проверки гипотез (или доверительных интервалов).

Кроме того, благодаря парадоксу конвертов мы выяснили, что если мы не имеем информации о вероятностях гипотез, то это не даёт нам право считать эти гипотезы равновероятными. А NFL именно так и делает (даже если явно вероятности и не вводит).

Сравнивать количество целевых функций или количество последовательностей не имеет никакого смысла, хотя бы потому что их множество может быть континуально. Имеет смысл сравнивать вероятность успешного решения задачи для разных методов (алгоритмов).

В духе NFL мы можем задать равномерное распределение на всех последовательностях. Но при таком распределении вероятность появления 1 на любом месте последовательности равна 0,5. Однако если, как в примере, из 100 значений мы получили 90 единиц, то это даёт основания отвергнуть гипотезу о том, что вероятность единицы равна 0,5. В результате мы вполне можем заключить, что с некоторой доверительной вероятностью точность нашего алгоритма прогнозирования превышает 0,5, и построить

доверительный интервал для этой точности (подобное уже делали).

Таким образом, «парадокс» NFL снимается.

Современные методы машинного обучения не делают предположений о виде распределений, но неявно проверяют гипотезу о том, что решение заданной формы имеет приемлемое качество.

Таким образом, NFL действительно следует интерпретировать как тот факт, что без всяких предположений решать задачи анализа данных невозможно. Но любые разумные методы такого рода предположения и делают, и проверяют — чаще неявно.

10.1.4 Алгоритм СПА

Алгоритм случайного поиска с адаптацией [28] является одним из первых представителей стохастических алгоритмов оптимизации.

Для использования алгоритма СПА необходимо чтобы на точках из Z была определена мера сходства, которую будем для удобства называть расстоянием, при этом не подразумевая обязательного выполнения свойств метрики.

Идея алгоритма СПА состоит в том, что плотности вероятности для выбора последующих точек задаются так, чтобы их значения были больше в окрестностях тех из уже поставленных точек, в которых значения целевой функции ближе к оптимальному, а также в тех областях, где плотность точек (экспериментов) ниже (в «малоизученных» областях). При этом данные плотности зависят от числа уже проведенных экспериментов.

Эта идея может реализовываться (конкретизироваться) различными способами. Первая версия алгоритма была разработана для пространства бинарных переменных. Следующие версии использовали для разбиения пространства поиска дерева решений.

Другим представителем стохастических алгоритмов оптимизации являются широко известные генетические алгоритмы, которые тоже реализуют идею случайного адаптивного поиска.

Заметим, что идея использования генетических алгоритмов

для поиска экстремума оформилась далеко не сразу. Первые работы 1954 года были посвящены имитационному моделированию эволюции. И только в 70-х годах идеи эволюции были полноценно применены для поиска экстремума функций.

Генетические алгоритмы по факту реализуют идеи, близкие идеям СПА. Так, «мутации» являются фактически поиском в окрестности «успешных» точек. Даже «скрещивание» можно рассматривать как поиск в окрестности особого вида.

10.2 Кластерный анализ

Задачи кластерного анализа состоят в том, чтобы разбить заданную совокупность объектов на группы так, чтобы внутри каждой группы объекты были в некотором смысле похожими, а объекты разных групп максимально отличались. Синонимами кластерного анализа являются таксономия, автоматическая группировка, классификация «без учителя». Последний термин подразумевает, что в отличие от рассматривавшейся ранее задачи классификации («с учителем») в обучающей выборке отсутствует информация о принадлежности объектов классам (т. е. о значениях переменной Y).

Таким образом, обучающая выборка в задаче кластерного анализа имеет вид

$$S = (x^i, i = 1, \dots, N).$$

Иногда в значении кластерного анализа используют термин «автоматическая классификация», однако данный термин лучше не использовать ввиду неоднозначности, поскольку классификация «с учителем» имеет такие же основания называться автоматической, поскольку производится компьютерной программой.

Кластеризация обычно использует некоторую функцию сходства (или различия) объектов, например расстояние. Некоторые методы ограничиваются понятием соседства объектов в топологическом смысле, дополнительно используя некоторую меру.

Первый подкласс методов — методы «нисходящей» кластеризации. В этих методах выбирается количество кластеров и про-

изводится разбиение множества объектов на кластеры (группы объектов) в соответствии с некоторым критерием.

10.2.1 Метод k -средних

Метод заключается в разбиении объектов на группы так, чтобы сумма расстояний от объектов до центров групп была минимальной.

Заметим, что нахождение оптимального разбиения в такой постановке является NP-трудной задачей, поэтому используются эвристические алгоритмы.

Самым распространённым является алгоритм k -средних, который является модификацией известного EM-алгоритма для оценки параметров смеси распределений.

Алгоритм состоит из следующих шагов.

1. Выбирается некоторая начальная расстановка k точек, которые полагаются центрами кластеров.
2. Для каждой точки выборки находится ближайший центр, и точке приписывается его кластер.
3. Для полученных кластеров вычисляются новые центры.

Шаги 2 и 3 выполняются в цикле, пока кластеризация не стабилизируется.

Данный алгоритм, как правило, находит локальный оптимум, зачастую сильно отличающийся от глобального. Для улучшения кластеризации алгоритм обычно запускают многократно при разных (случайных) начальных расстановках центров.

10.2.2 Оценивание смеси распределений

Смесью распределений называется распределение, функция распределения которого представляется взвешенной суммой некоторых функций распределения заданного вида. Чаще смесь зада-

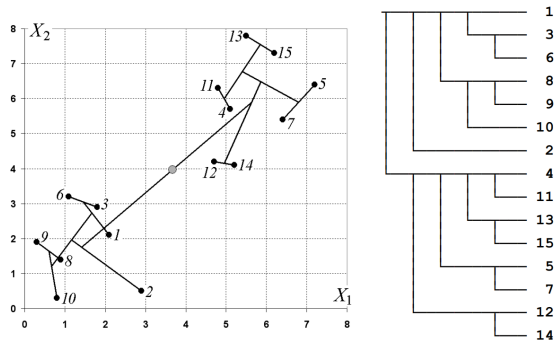


Рис. 10.1. Пример иерархической кластеризации

ётся через плотности вероятности

$$\varphi(x) = \sum_{j=1}^k \alpha_j \varphi_j(x),$$

где $\varphi_j(x)$ – плотности заданного вида, α_j – весовые коэффициенты.

Плотности обычно выбираются из некоторого параметрического семейства. В этом случае задача кластеризации становится стандартной статистической задачей оценивания параметров распределения. Для её решения существуют хорошо развитые методы. В кластерном анализе чаще всего используются смеси нормальных распределений.

10.2.3 Иерархическая кластеризация

Данный класс методов использует идею «восходящей» кластеризации, когда точки последовательно присоединяются к кластерам, а кластеры последовательно укрупняются. При этом последовательное объединение кластеров формирует дерево.

На рис. 10.1 приведён пример иерархической кластеризации. Справа приведено дерево кластеризации. Два поддерева, выходящие из корневой вершины, в точности совпадают с кластера-

ми, полученными методом k -средних при $k = 2$. При этом дерево несёт гораздо больше информации о структуре расположения точек, чем обычная кластеризация.

Иерархическая кластеризация может использоваться и в том случае, когда известны только попарные расстояния между точками, а координаты точек неизвестны или вовсе не имеют смысла. Такая ситуация может возникнуть, например, при группировке музыкальных произведений. С помощью экспертов могут быть даны численные оценки схожести (или, наоборот, отличия) произведений. Тогда для каждого произведения будут известны «расстояния» от него до каждого из остальных произведений. При этом музыкальные произведения не задаются точками в пространстве переменных.

10.3 Анализ временных рядов

Прогнозировать временной ряд можно путём сведения к таблице данных.

Также есть специализированные методы (Хольта–Винтерса, ARIMA и т. п.), есть специальные архитектуры нейронных сетей (рекуррентные, например LSTM).

Задачу прогнозирования временного ряда проиллюстрируем для случая одной бинарной (двоичной) переменной. Для её решения будем использовать одну из модификаций метода прецедентов, сводящуюся в данном случае к оцениванию частот различных предысторий.

В качестве исходных данных возьмём последовательность, сформированную вручную из 0 и 1, выбиравшихся наугад. На первый взгляд, может показаться, что вероятности появления 0 и 1 не зависят от ранее выбранных значений и равны 0,5.

Если бы это было так, то такую последовательность не имело бы смысла прогнозировать, поскольку для любого решающего правила вероятность ошибки была бы 0,5. Однако человек, вообще говоря, не является идеальным генератором случайных чисел и, даже стараясь выбирать числа равновероятно и «бессистемно»,

он составляет ряд, в котором присутствуют статистические закономерности.

Исходная длина последовательности, представленной в таблице, равна 150. Для того, чтобы иметь возможность проверить качество прогноза, разобьём ряд на две части: обучающую, длиной 100, и контрольную, длиной 50.

Таблица. Временные ряды

обучение

01101000101001101110101100001011100111101010110101

000010110111110101011110101010010111010101101010

контроль

11101010111100110000010110101001011100101010101011

прогноз

10111010101101111110110101101011110111111010101010

Выберем значения параметра d – длины предыстории, по которой будет производиться прогноз. При используемом методе прогнозирования d должно быть таким, чтобы 2^d – число возможных предысторий было существенно меньше длины обучающей последовательности. Возьмём $d = 4$.

Далее перебираем все возможные последовательности из пяти 0 и 1 и подсчитываем, сколько раз они встречаются в обучающем ряде. Например, последовательность 00000 не встретилась ни разу, а фрагмент 00001 присутствует два раза.

Сравнив прогноз с контрольной последовательностью, обнаруживаем 17 несовпадений, т. е. доля ошибок прогноза есть 0,34.

Получившаяся доля ошибок существенно отличается от 0,5.

Применив классические критерии проверки гипотез, легко показать, что это отличие статистически значимо и подобный ряд действительно можно прогнозировать с вероятностью ошибки, меньшей 0,5.

10.4 Линейная регрессия

Метод линейной регрессии является одним из базовых методов анализа данных, и его рассмотрение входит в большинство

классических курсов математической статистики... Последнее является одной из причин, почему мы «не спешили» его рассматривать. Однако основной причиной, почему линейная регрессия рассматривается только сейчас, является её обособленность в плане используемых математических техник.

10.4.1 Постановка задачи

Пусть $X = R^n$ – пространство значений прогнозирующих переменных, $Y = R$ – значения прогнозируемой переменной, $D = X \times Y$.

Все переменные вещественные и являются значениями случайных величин.

Дана $S = ((x^i, y^i) \in D \mid i = \overline{1, N})$ – случайная независимая выборка, $S \in D^N$.

Здесь $x^i = (x_1^i, \dots, x_n^i)$ – вектор значений для i -го элемента выборки.

Образуем из элементов выборки матрицу $\mathcal{X} = (x_j^i)_{N \times n}$ и вектор $\mathcal{Y} = (y^i)_{N \times 1}$.

Решающую функцию положим линейной

$$f_w(x) = wx,$$

т. е. будем задавать как скалярное произведение вектора значений признаков на вектор параметров $w = (w_1, \dots, w_n)$. Знак транспонирования первого вектора в скалярном произведении обычно будем опускать.

В общем случае нужно к прогнозу добавить константу, но вместо этого мы будем считать, что среди переменных есть переменная, тождественно принимающая значения 1. Это эквивалентно наличию свободного члена, но выкладки становятся проще.

Параметры будем искать по критерию минимума суммы квадратов отклонений на выборке

$$w^* = \arg \min_w \sum_{i=1}^N (f_w(x^i) - y^i)^2 = \arg \min_w (\mathcal{X}w - \mathcal{Y})^2. \quad (10.1)$$

В правой части выражение приведено к матричной форме.

Для нахождения минимума возьмём производную и приравняем к нулю

$$\frac{\partial}{\partial w} (\mathcal{X}w - \mathcal{Y})^2 = 2\mathcal{X}'(\mathcal{X}w - \mathcal{Y}) = 0.$$

Здесь штрих означает транспонирование.

Решая линейное уравнение $\mathcal{X}'\mathcal{X}w = \mathcal{X}'\mathcal{Y}$, получаем

$$w^* = (\mathcal{X}'\mathcal{X})^{-1} \mathcal{X}'\mathcal{Y} = \mathcal{X}^+\mathcal{Y}. \quad (10.2)$$

Матрица $\mathcal{X}^+ = (\mathcal{X}'\mathcal{X})^{-1} \mathcal{X}'$ называется псевдообратной для прямоугольной матрицы \mathcal{X} .

Для нахождения псевдообратной матрицы удобно использовать сингулярное разложение.

10.4.2 Сингулярное разложение

Произвольную $N \times n$ матрицу можно представить в виде сингулярного разложения (singular value decomposition, SVD). Применительно к матрице \mathcal{X} получим

$$\mathcal{X} = V\Delta U',$$

где матрица Δ диагональная, а V и U ортогональные, т. е.

$$\Delta = \begin{pmatrix} d_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & d_n \end{pmatrix}, \quad V'V = I_n, \quad U'U = I_n,$$

где I_n – единичная матрица размера $n \times n$. Матрица V имеет размерность $N \times n$, матрица U имеет размерность $n \times n$.

Элементы d_j называются сингулярными числами матрицы \mathcal{X} . Для симметричных матриц сингулярные числа с точностью до знака совпадают с собственными числами.

Заметим, что собственными числами матрицы $\mathcal{X}'\mathcal{X}$ будут $(d_j)^2$.

Подставим сингулярное разложение в матрицу \mathcal{X}^+

$$\begin{aligned}\mathcal{X}^+ &= ((V\Delta U')'V\Delta U')^{-1}(V\Delta U')' = \\ &= (U\Delta V'V\Delta U')^{-1}(U\Delta V') = (U\Delta^{-2}U')(U\Delta V') = U\Delta^{-1}V'.\end{aligned}$$

На диагонали матрицы Δ^{-1} стоят значения $(d_j)^{-1}$.

Если у матрицы \mathcal{X} имеются нулевые сингулярные числа, то матрица \mathcal{X}^+ не существует. Однако это не значит, что не существует решения линейной регрессии. Наоборот, таких решений становится бесконечно много.

10.4.3 Регуляризация

Для начала рассмотрим один вырожденный, но очень наглядный пример.

Пусть $Y = Y(\cdot)$ – некоторая случайная величина¹, и мы строим регрессию в двумерном пространстве, при этом $X_1 = Y$ и $X_2 = Y$.

В этом вырожденном примере любая из переменных даёт идеальный прогноз, а также идеальный прогноз даёт любая комбинация

$$f_w(x) = ax_1 + (1 - a)x_2,$$

где a – любое число. Таким образом, при наличии выборки решением оптимизационной задачи (10.1) будет любой вектор вида $w^* = (a, 1 - a)$.

Предположим теперь, что прогнозирующие переменные воспроизводят целевую переменную не точно, а с небольшим «шумом», т. е. $X_1 = Y + \varepsilon_1$ и $X_2 = Y + \varepsilon_2$. Здесь ε_1 и ε_2 – независимые случайные переменные с нулевым средним и дисперсией σ^2 .

Можем вычислить погрешность прогноза, она составит

$$E(f_w(X) - Y)^2 = (a^2 + (1 - a)^2) \cdot \sigma^2 = (w^*)^2 \cdot \sigma^2.$$

¹Здесь мы обозначили случайную величину той же буквой, которой ранее обозначили множество её значений. Так делать неправильно, но это кажется меньшим злом, чем усложнять обозначения. В данном случае из контекста всегда понятно, что именно стоит за обозначением.

Получается, что в рассмотренном примере ошибка получается попросту пропорциональной норме вектора коэффициентов регрессии.

При этом минимальной ошибка будет при $a = 0,5$, т. е. $w^* = (0,5, 0,5)$. Это вполне согласуется с интуицией, согласно которой при наличии дублирующихся переменных мы их усредняем с равными весами.

Выясним, что получится, если в этом же примере попытаемся использовать формулу (10.2).

При отсутствии шума у нас её применить вообще не получится, поскольку матрица окажется необратимой.

Если шум есть, но небольшой, то одно из сингулярных значений будет очень малым. А поскольку в выражении для коэффициентов регрессии сингулярные числа стоят в знаменателе, норма вектора w^* будет большой.

Получается, что выражение (10.2) «усиливает шум», если присутствуют малые (по модулю) сингулярные значения.

При решении систем линейных уравнений принято использовать число обусловленности. Для матрицы $\mathcal{X}'\mathcal{X}$ число обусловленности можно найти как

$$\mu(\mathcal{X}'\mathcal{X}) = \frac{\max_j (d_j)^2}{\min_j (d_j)^2}.$$

Если число обусловленности большое (данная ситуация называется проблемой мультиколлинеарности), то формула (10.2) даёт неустойчивые решения. Чтобы повысить устойчивость, применяют регуляризацию.

Идею регуляризации подсказывает рассмотренный пример: при прочих равных нужно выбирать вектор коэффициентов с меньшей нормой.

Добавим соответствующее слагаемое в (10.1). Получим

$$w^* = \arg \min_w \left((\mathcal{X}w - \mathcal{Y})^2 + \alpha w^2 \right). \quad (10.3)$$

Приравняв производную к нулю и решив уравнение, получим

$$w^* = (\mathcal{X}'\mathcal{X} + \alpha I_n)^{-1} \mathcal{X}'\mathcal{Y}. \quad (10.4)$$

В отличие от (10.2), здесь перед обращением матрицы $\mathcal{X}'\mathcal{X}$ к ней добавляется диагональная матрица αI_n , которая может называться ассоциацией с «гребнем». Отсюда возникло название метода: гребневая регрессия (ridge regression).

Константа α является гиперпараметром и подбирается эмпирически.

Существуют и другие методы регуляризации, например метод лассо Тибширани. Есть также реализации, поддерживающие ограничение неотрицательности коэффициентов.

10.4.4 Задача прогнозирования

До сих пор мы рассматривали метод линейной регрессии как некоторый алгоритм. Настало время уточнить, какую задачу мы при этом решаем.

Как и раньше, наша задача — задача прогнозирования. Здесь слово «прогнозирование» использовано для краткости. Оно не подразумевает зависимость от времени или прогноз в будущее, а предполагает просто оценку значения целевой переменной для новых объектов. Строгий термин для такой задачи — задача построения решающей функции.

Ключевым элементом в постановке такой задачи является метрика качества.

Одна из типовых метрик в задачах с непрерывной целевой переменной — коэффициент детерминации

$$1 - \frac{E(f(X) - Y)^2}{DY}.$$

Наилучшее значение этой метрики равно 1, а снизу она не ограничена, однако отрицательные значения означают, что решение хуже даже простого константного прогноза.

А если в качестве целевой метрики задана MAE, т. е. средний модуль отклонения. Можем мы тогда использовать рассмотренный метод линейной регрессии? Кажется, что нет, ведь МНК (метод наименьших квадратов) минимизирует квадратичную ошибку, а не модуль. Но на самом деле можем. В машинном обучении

вполне типичная и нормальная ситуация, когда функция потерь (loss) не совпадает с метрикой.

В реальных задачах мы, как правило, вообще ничего не знаем о вероятностной модели для данных. А раз так, то любые методы обучения становятся чисто эвристическими. И единственным их обоснованием будет точность на тестовой выборке (или на кроссвалидации). Поэтому мы в полном праве пробовать любой метод, даже тот, что выглядит заведомо неподходящим.

Почему же тогда в литературе для применимости линейной регрессии обычно перечисляют такие условия, как: линейность, нормальность, гомоскедастичность, независимость и отсутствие мультиколлинеарности? Всё дело в том, что там решается совершенно другая задача, а именно задача идентификации модели. Это классическая постановка задачи регрессионного анализа.

В той постановке предполагается, что существует истинная зависимость, и цель в том, чтобы оценить параметры этой зависимости. Часто хотят, чтобы оценка была оценкой максимального правдоподобия. Тогда в случае нормально распределённых ошибок нужно минимизировать квадраты отклонений, а для распределения Лапласа — сумму модулей.

Выполнение условий классической постановки позволяет также построить доверительные интервалы для параметров модели.

А в постановке, что мы используем, даже говорить о доверительных интервалах для параметров модели не имеет смысла. Просто потому, что нет истинной модели и нет истинных значений параметров.

Мы, конечно, можем обозвать истинной моделью ту, что обеспечивает лучшее значение метрики (на распределении). Но это не имеет смысла, так как доверительных интервалов для параметров всё равно построить не сможем, поскольку у нас нет информации о виде распределений в задачах.

Таким образом, нужно отметить, что классическая постановка задачи регрессионного анализа — это важная задача в классической статистике, но она соответствует другому кругу прикладных задач.

10.5 Использование экспертных знаний

В настоящее время экспертные системы не имеют прежней популярности. Однако и сейчас существуют люди, обладающие уникальными знаниями в определённой области. Будем называть таких людей экспертами. Очевидно, что знания экспертов нужно уметь использовать. В частности, актуальной остаётся проблема совместного использования данных и знаний.

В работе [29] описан метод решения задачи прогнозирования на основе анализа эмпирической информации, которая может быть представлена либо набором высказываний экспертов, либо таблицей экспериментальных данных (выборкой), либо экспертными высказываниями и таблицей данных одновременно.

Чтобы распространить байесовский подход на случай, когда эмпирическая информация представлена экспертными суждениями, предлагается статистическая интерпретация экспертной информации. При этом предполагается, что экспертная информация представлена высказываниями нескольких экспертов. Допускается, что высказывания различных экспертов могут быть несогласованы и, в частности, быть противоречащими друг другу в той или иной степени, совпадающими, частично совпадающими, дополнительными.

10.5.1 Постановка задачи

Рассмотрим задачу прогнозирования целевой переменной Y со значениями $\{-1, 1\}$.

Введём обозначение

$$g(x) = \frac{P(dx, y = 1)}{P(dx)}.$$

Таким образом, $g(x)$ представляет собой вероятность первого класса при условии, что значения переменных X_1, \dots, X_n равны $x = (x_1, \dots, x_n)$.

Под задачей прогнозирования будем понимать восстановление $g(x)$ на основе эмпирической информации, то есть построение

некоторой оценки $\bar{g}(z)$ функции $g(z)$.

Для удобства дальнейших обозначений введем множество C , элементы которого будем называть стратегиями природы и обозначать c . При этом каждой функции $g(z)$ поставим во взаимно-однозначное соответствие некоторую $c \in C$. В дальнейшем вместо $g(x)$ будем записывать $g_c(x)$.

В отличие от известных подходов к решению задач восстановления распределений на основе выборки предлагаемый в данной работе метод ориентирован на использование эмпирической информации как в виде таблицы экспериментальных данных, так и в виде набора высказываний, полученных от многих экспертов.

Формально высказывание эксперта может быть представлено в виде следующей тройки:

$$B_i = \langle E^i, \beta^i, \gamma_i \rangle,$$

где $i = \overline{1, N}$ (N – число высказываний, полученных от экспертов). При этом $E^i \subseteq D_X$, $\beta_k^i \in [0, 1]$, $\gamma_i \in [0, 1]$.

Данное высказывание может быть проинтерпретировано следующим образом: «Если x – некоторая (соответствующая объекту) точка из D_X , то вероятность того, что $y = 1$ будет равна β^i ». Число γ_i отражает уверенность эксперта в истинности сделанного утверждения. Заметим, однако, что смысл высказывания имеет для нас значение лишь с точки зрения его влияния на зависимость вероятности появления данного высказывания от стратегии c .

Предположим, что для каждой стратегии c нам известна $P(B_i/c)$ – условная вероятность появления высказывания B_i . Тогда, поскольку эксперты делают высказывания независимо,

$$P(B/c) = \prod_{i=1}^N P(B_i/c).$$

Здесь $B = \{B_i | i = \overline{1, N}\}$ – множество всех высказываний, представленных экспертами.

Если помимо экспертной информации мы имеем и обычную таблицу экспериментальных данных, то мы можем просто пере-

множить вероятность появления экспертной информации и вероятность появления выборки.

10.5.2 Байесовский подход

Чтобы воспользоваться байесовским подходом к восстановлению $g(x)$, нужно задать априорную вероятностную меру $P(C)$. После этого

$$\bar{g}^*(x) = \frac{\int_C g_c(x) P(B/c) P(dc)}{\int_C P(B/c) P(dc)}.$$

Данное выражение, полученное из формулы Байеса, дает оптимальную оценку для функции $g(x)$ в том смысле, что $\bar{g}^*(x)$ есть условное (при заданном B) математическое ожидание функции $g(x)$.

Алгоритм, который на основе имеющейся информации B строит оценку функции $g(x)$ в соответствии с формулой, будем называть байесовским алгоритмом решения рассматриваемой задачи прогнозирования.

Обычно в теории статистических решений при задании априорной меры $P(C)$ вводятся сильные ограничения на класс стратегий природы, которые во многих случаях не оправданы. Однако существуют универсальные классы распределений, на которых можно вводить разумную априорную меру.

10.5.3 Статистическая интерпретация экспертной информации

В отличие от известных методов логического анализа экспертных высказываний в данной работе предлагается статистическая интерпретация экспертной информации. При этом полагается, что вероятность появления каждого высказывания зависит от того, какая стратегия природы имеет место.

Эксперта s будем представлять как оператор, сопоставляющий каждой стратегии c условное распределение вероятностей $P_s(u/c)$

на множестве U , где U – множество подмножеств всех допустимых высказываний. Заметим, что U – конечно ввиду конечности алфавита, в котором записываются высказывания, и естественных ограничений на длину и количество высказываний. Если эксперты, делающие высказывания, случайно выбираются из множества S в соответствии с некоторой вероятностной мерой $P(S)$, то $P(u/c) = \sum_s P_s(u/c)P(s)$.

Заметим, что функция $P(u/c)$ является «в природе» вполне определенной, а потому ее конкретный вид может быть установлен непосредственным экспериментальным исследованием.

10.5.4 Примеры

Получаемую в результате оценку $\bar{g}^*(x)$ будем записывать также в форме набора высказываний, которые будут иметь вид:

$$\langle \tilde{E}_l, \tilde{\beta}_l \rangle, \quad \tilde{E}_l \subseteq D_X, \quad l = 1, \tilde{N},$$

где $\bar{g}^*(x) = \tilde{\beta}_l$ при $x \in \tilde{E}_l$, \tilde{N} – число областей постоянства функции $\bar{g}^*(x)$ в пространстве D_X .

Пример 1. Пусть исходная информация представлена единственным высказыванием:

$$1) \langle E, 0,9, 0,8 \rangle.$$

Применяя соотношение (1), получаем:

$$1) \langle E, 0,86 \rangle.$$

Видим, что $\tilde{\beta}$ оказалась не равной β , а сдвинутой к 0,5 (т.е. в сторону меньшей определенности). Это вполне оправдано, поскольку уверенность эксперта (как и доверие ему) не абсолютна.

Пример 2. Пусть даны два одинаковых высказывания двух экспертов:

$$1) \langle E, 0,9, 0,8 \rangle;$$

$$2) \langle E, 0,9, 0,8 \rangle.$$

В результате получим:

$$1) \langle E, 0,89 \rangle.$$

Оценка $\tilde{\beta}$ уже гораздо ближе к β_1 (или к β_2), чем в первом случае. Это означает, что степень доверия двум экспертам, неза-

висимо подтверждающим мнения друг друга, выше, чем доверие высказыванию одного эксперта.

Пример 3. Дана информация:

- 1) $\langle \{a, b\}, 0,9, 0,8 \rangle$;
- 2) $\langle \{b, c\}, 0,1, 0,8 \rangle$;
- 3) $\langle \{c, d\}, 0,9, 0,8 \rangle$.

В данном примере пространство $D_X = \{a, b, c, d\}$ состоит из значений единственной номинальной переменной X .

Результат согласования:

- 1) $\langle (x = a), 0,8 \rangle$;
- 2) $\langle (x = b), 0,72 \rangle$;
- 3) $\langle (x = c), 0,72 \rangle$;
- 4) $\langle (x = d), 0,8 \rangle$.

Результат представляется интуитивно вполне оправданным. Действительно, поскольку высказывания противоречат друг другу, то оценки для $g(x, 0)$ значительно сдвинуты к 0,5. Далее, ввиду того, что второй эксперт противоречит одновременно как первому так и третьему, то доверие его утверждению относительно ниже. Этим, в частности, объясняется, например, то, что для точки b решение ближе к оценкам первого эксперта, чем второго.

10.6 Некоторые задачи и приёмы решения

10.6.1 Пример Штейна

Предположим, что наша задача — оценить средний доход представителей разных профессий по выборочным данным.

Профессия представляет собой пример категориальной переменной. Доход — количественная целевая переменная. Метрику качества используем MSE, т. е. средний квадрат отклонения.

Казалось бы, задача — проще некуда и решение очевидно: группируем таблицу данных по профессиям и в качестве прогноза берём средний доход представителей профессии.

Однако оказывается, что такое решение может не быть оптимальным с точки зрения метрики качества. Об этом свидетельствует пример, построенный Штейном.

Пусть m – число значений категориальной переменной $X = \{1, \dots, m\}$, а целевая переменная Y при условии $X = j$ подчиняется нормальному распределению с параметрами (θ_j, σ) . Применительно к примеру θ_j соответствует среднему доходу j -й профессии.

Предположим теперь, что выборка содержит ровно одного представителя каждой профессии. Иными словами, в качестве выборки мы имеем случайный вектор $y = (y_1, \dots, y_m)$.

Кажется очевидным, что в качестве оценки вектора параметров $\theta = (\theta_1, \dots, \theta_m)$ следует взять вектор y , т. е. оценкой дохода для профессии будет доход её имеющегося представителя.

Для такой оценки MSE будет равен $E(y - \theta)^2 = \sigma^2$.

Оценка Джеймса–Штейна (James–Stein estimator) имеет вид

$$\tilde{\theta} = \left(1 - \frac{(m-2) \cdot \sigma^2}{y^2}\right) \cdot y.$$

Фактически оценка просто уменьшает все выборочные значения пропорционально на некоторую величину.

Оказывается, что при $m \geq 3$ для этой оценки MSE будет меньше, чем σ^2 .

Это кажется удивительным, поскольку в рассмотренной модели доходы разных профессий независимы.

Можно было в качестве θ_j взять совершенно не связанные величины, например: температуру на Марсе (в момент приземления туда аппарата), количество упавших метеоритов за год и численность крокодилов в среднем течении Нила. Нужно только нормировать величины, чтобы дисперсия стала одинаковой.

Чтобы убедиться в справедливости утверждения, проведём статистическое моделирование. На рис. 10.2 приведён результат работы следующей программы.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def calc_c(y):
    return 1.0 - (y.shape[0]-2) / (y*y).sum()
```

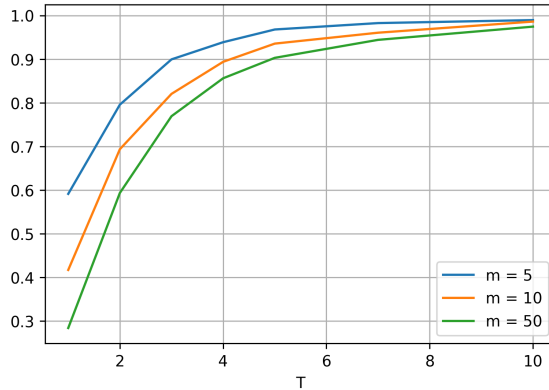


Рис. 10.2. Значения MSE для оценки Штейна

```

def calc_mse(t, p):
    d = t - p
    return (d*d).mean()

def sample(t):
    m = t.shape[0]
    x = np.random.normal(size=m)
    y = x + t
    return calc_mse(t, y*calc_c(y))

def model(T, m, K=10000):
    t = np.linspace(0.0, T, m)
    return sum([sample(t) for i in range(K)]) / K

V = [1, 2, 3, 4, 5, 7, 10]
df = pd.DataFrame({'m': %i, 'mse': [model(T, m) for T in V]
                  for m in (5, 10, 50)}, index=V)
df.index.name = 'T'
df.plot(figsize=(6, 4), grid='on')

```

Для простоты мы положили $\sigma^2 = 1$.

Как видим, MSE для оценки Джеймса–Штейна действительно существенно меньше 1.

Вполне вероятно, что похожие примеры можно сконструировать и для более реалистичной ситуации, а именно когда в каждой

категории имеется больше одного объекта выборки, а дисперсии неизвестны.

Пример Штейна показывает, что даже такая простая задача, как построение решения по единственной категориальной переменной, на самом деле не столь очевидна как кажется.

10.6.2 Прогнозирование нескольких переменных

Рассмотрим следующую задачу прогнозирования.

Пусть имеется генеральная совокупность объектов Γ , для которой определена произвольная вероятностная мера P .

Каждый объект может быть охарактеризован значениями $x_1, \dots, x_n; y_1, \dots, y_m$ переменных $X_1, \dots, X_n; Y_1, \dots, Y_m$. Данные переменные могут быть произвольных типов (вещественные, целые, порядковые, номинальные, бинарные).

Рассматриваемая задача состоит в том, чтобы для произвольного объекта из Γ по известным значениям переменных X_1, \dots, X_n предсказать значения переменных Y_1, \dots, Y_m на основе анализа имеющейся эмпирической информации.

Обычно в подобных случаях все переменные прогнозируют отдельно, т. е. независимо.

Но целевые переменные могут быть объективно зависимыми (даже условно, т. е. при условии конкретных x). Поэтому лучше прогнозировать переменные последовательно, добавляя уже предсказанные переменные в модель. А ещё лучше иметь методы, которые строят модель сразу для всех целевых переменных.

Некоторое время назад автору встретилась одна задача на kaggle, где нейросети значительно превзошли бустинг по точности. Такая ситуация, как известно, нетипична для задач с табличными данными.

Особенностью той задачи было как раз наличие большого количества целевых переменных. И, скорее всего, преимущество нейросетей и было обусловлено тем, что для них как раз естественно прогнозировать все переменные разом. Тогда как реализации бустинга в то время не умели этого делать и строили для

каждой целевой переменной отдельный ансамбль деревьев.

Был проведён эксперимент, который показал, что если нейросеть (той же архитектуры) обучать для каждой целевой переменной отдельно, то её точность проседает даже ниже бустинга.

Получается, что имеет место весьма неочевидный эффект. Пусть нам на самом деле нужно прогнозировать лишь одну переменную. Оказывается, что просто добавив к данным ещё некоторое количество «ненужных» переменных в качестве целевых, мы можем получить модель, которая более точно прогнозирует исходную целевую переменную. Заметим, что мы не можем добавить эти переменные к переменным X поскольку на inference, т. е. при применении решающей функции к новым данным, их значения будут неизвестны.

10.6.3 Проблема *target leaks*

В машинном обучении довольно часто встречаются термины, которые удобнее использовать в англоязычном варианте, поскольку перевод не устоялся и могут возникнуть разночтения и неточности в понимании.

Однако термин *target leaks* уникальный, поскольку, похоже, вообще не имеет приемлемого перевода.

Буквальный перевод звучит как «утечки цели», что слишком неблагозвучно и попросту говоря коряво.

Под *target leaks* будем понимать ситуацию, когда решающая функция использует информацию, которая будет недоступна при применении модели к новым данным (т. е. на inference).

Смысл этого явления проще объяснить на примерах.

Одна из задач с kaggle заключалась в распознавании человека по данным акселерометра смартфона. Предполагалось, что каждый человек держит и двигает смартфон (при использовании) индивидуальным образом и можно по временному ряду ускорений, которые испытывает корпус, распознать, в чьих руках находится смартфон.

Оказалось, что самым информативным признаком в этой за-

даче является «шаг», на который отличаются ближайшие показания акселерометра. Дело в том, что изначально акселерометр возвращает целые значения, которые затем переводятся в вещественные, при этом получаются не любые значения, а только те, что лежат на некоторой «сетке». И вот шаг этой сетки, как и сам набор её значений и был основным признаком.

Получается, что модель обучилась распознавать не человека, который держит телефон, а само устройство. А поскольку при сборе данных каждый человек работал на своём телефоне, то определив телефон, мы определяем человека. Однако если человек возьмёт чужой или новый телефон, то модель не сможет его распознать. В результате модель оказывается бесполезной.

Другой показательный пример (тоже с kaggle) — задача прогнозирования того, завершит ли клиент бронирование отеля на сайте агрегатора.

Каждый отель описывался некоторым набором признаков (важно, что название отеля не сообщалось и определить отель по этим признакам было невозможно), также присутствовали признаки, характеризующие клиента, и был такой признак как расстояние от текущего местоположения клиента до отеля. Недостатком поставщика данных заключался в том, что это расстояние было дано с точностью до метров, что позволило определить конкретный отель. При этом отели сильно различаются по доле завершённых бронирований (очевидно, что для популярных отелей она выше), и знание отеля позволяет делать очень точный прогноз.

В результате победителем оказалась модель, которая по расстоянию очень точно определяет отель и даёт прогноз просто по доле успешных бронирований для него (остальные признаки вообще не нужны).

Очевидно, это не то, что хотел получить заказчик.

Рассмотрим пример с одного из отечественных конкурсов.

Ставилась задача распознавания объекта по «изображению», полученному с лидара.

В изображении с лидара яркость пикселя отражает не яркость

точки на объекте, а расстояние до точки объекта. Визуально картинка представляет собой полутонное изображение специфического вида.

Технически мы имеем типовую задачу сегментации изображений. И очевидно, что в таких задачах обучающая и тестовая выборки должны состояться из разных изображений. Однако в данном конкурсе тестовая выборка была представлена набором пикселей, взятых случайно из тех же изображений, откуда брались пиксели для обучающей выборки.

В результате получилась задача, в которой все методы работают примерно на уровне kNN , и эта задача не имеет никакого практического смысла.

К типичным случаям *target leak* относится обучение *target encoding* совместно на обучающей и тестовой выборках.

Часто *target leak* связан с пропусками в данных.

Приведём умозрительный пример. Пусть задача заключается в постановке определённого диагноза. При этом всем больным назначается некоторый анализ. И этот же анализ назначается многим здоровым, но далеко не всем. Получается, что по наличию пропуска в столбце с этим анализом мы можем точно определить, что человек (из обучающей и тестовой выборок) здоров. Но это не будет работать на новых пациентах.

Следующий пример реальный. Задача заключалась в определении вида культуры по плотности зелёной массы на поле.

При этом одно хозяйство использовало, упрощённо говоря, качественный аппарат для измерения плотности, а другое — некачественный.

Некачественный аппарат зачастую выдавал даже не пропуски, а некорректные (очевидно недопустимые) значения. По ним можно было понять, что данные принадлежат второму хозяйству.

Осталось заметить, что разные хозяйства специализируются на разных культурах.

Близкой (и зачастую трудно отличимой) к *target leak* является ситуация *domain shift*.

Формально *domain shift* — это когда при использовании модели

данные имеют иное распределение, чем при обучении.

На конкурсах эксплуатация target leak является вполне корректным приёмом со стороны участников. Если организаторы пытаются его «запретить», это говорит об их некомпетентности, хотя бы потому, что в принципе невозможно точно определить, действительно ли модель использует target leak.

Также вряд ли можно назвать такой приём «неспортивным», поскольку по сути он мало отличается от использования «зевков» противника в шахматах.

При обнаружении target leak у организаторов есть два корректных выхода: «перезапустить» конкурс, предоставив новые (исправленные) данные, либо продолжать конкурс как есть, но не пытаться решить проблему «костылями» в виде запретов.

К слову, показательным примером некорректного запрета является включение на одном из конкурсов в правила пункта о необходимости применять только модели машинного обучения и запрете на модели, основанные на правилах.

Наличие такого пункта свидетельствует о неграмотности составителя правил, поскольку нет способа корректно и строго ввести понятие метода машинного обучения так, чтобы в него не вошли методы, основанные на правилах. Правила — это эвристики. Но эвристики — неотъемлемая составляющая почти всех (на самом деле, вообще всех) методов машинного обучения. А разделять эвристики на допустимые и недопустимые — совершенно бесперспективная затея. Может, например, оказаться, что некоторая модель формально обучается, но фактически от этого обучения нет пользы, а весь результат обеспечивается эвристиками.

Следует отметить, что эксплуатация target leak допустима в конкурсах, но совершенно недопустима при решении задач в рамках проектов и договоров. В этих случаях корректная постановка задачи — зона ответственности исполнителя, а не заказчика (поставщика данных). Если в решении окажется target leak — это «зевок» исполнителя.

10.6.4 Заполнение пропусков

Заполнение пропусков может быть самостоятельной задачей. Она сводится к решению задач прогнозирования одних переменных на основе других.

Такая задача возникает не очень часто (на kaggle, кажется, не встречалась ни разу), но вполне имеет практический смысл. Пропуски в данных встречаются часто, и бывает необходимо знать, какое значение более вероятно должно быть на их месте.

Однако часто заполнение пропусков используется как вспомогательная процедура в задаче прогнозирования — и это всегда зло. Зачастую это зло неизбежное, поскольку многие методы попросту не умеют работать с пропусками, а удаление всех строк с пропусками бывает неприемлемо. Поэтому в некоторых случаях заполнять пропуски приходится (чтобы применить, например, линейные методы), но нужно понимать, что заполнение пропусков — это искажение данных.

Если метод может работать с пропусками, то заполнять их не следует. К таким методам относятся решающие деревья и всё, что на них основано (бустинг, случайный лес). Им нужно подавать данные как есть.

С другой стороны, существует один любопытный приём feature engineering. Он состоит в том, что произвольные переменные назначаются целевыми и строится модель, которая их прогнозирует. Полученные прогнозы добавляются к данным в качестве новых переменных. Фактически делается то же, что при заполнении пропусков, но прогнозы не подставляются вместо отсутствующих значений, а формируют новую переменную.

Такой приём иногда работает, и этот факт даёт основания предположить, что в каких-то случаях даже при использовании бустинга заполнение пропусков может сработать лучше, чем их оставление. Однако кажется, что ещё лучше будет добавить прогнозы новой переменной (а пропуски сохранить).

Пропуски иногда являются причиной target leak. Заполнение пропусков может уменьшить этот эффект, но не гарантирует его полное устранение.

При решении учебных и конкурсных задач об этом не стоит беспокоиться. При решении задач в рамках проектов target leak нужно либо исключить, либо точно оценить его эффект (это возможно при правильной организации сбора тестовых данных).

10.6.5 Обобщение задачи построения регрессии

В подавляющем большинстве случаев в качестве примера задачи машинного обучения мы рассматривали задачу классификации, причём всего на 2 класса. А задачу регрессии упоминали вскользь, будто в ней нет ничего интересного.

И действительно, в задаче классификации больше тонкостей. Это кажется парадоксальным, ведь выбрать (значение для прогноза) из двух значений проще, чем из континуума.

Всё дело в том, что задачу классификации мы обычно ставили как задачу прогнозирования вероятностей классов (т. е. условного распределения в точке x). И метрики качества часто основывались на вероятностях.

Регрессия выглядит проще, потому что в ней мы не пытались оценивать условное распределение. Но такую задачу вполне можно поставить.

Например, можно было бы дать прогноз погоды на завтра в виде: распределение температуры воздуха на завтрашний полдень оценивается как смесь двух нормальных распределений с модами в +10 и +25 градусов и стандартными отклонениями 8 и 5 градусов соответственно.

Такой прогноз погоды никому не нужен. О погоде важно знать ожидаемую температуру и вероятности экстремальных вариантов типа заморозка.

При этом уже появляются задачи, в которых нужно прогнозировать не просто значение непрерывной величины, но и её распределение (или, по крайней мере, некоторые характеристики распределения). Такие задачи могут быть формализованы как задачи прогнозирования условного распределения. Работы в этом направлении имеются, хотя пока их относительно немного.

10.6.6 Кодирование категориальных переменных

Категориальным признаком называется переменная, значения которой являются именами и не упорядочены. Примеры: цвета, запахи, профессии.

Кодирование категориальных переменных относится к разделу feature engineering.

Большинство методов не умеют работать с категориальными переменными и требуют их преобразования в числовые.

Самый простой метод преобразования — label encoding.

В этом методе значения переменной нумеруются (произвольным образом) и потом в таблице данных заменяются этими номерами.

Метод label encoding выглядит бессмысленным и некорректным (и обычно таковым и является), но зачастую приемлемо работает при использовании методов, основанных на деревьях.

Некорректность метода label encoding заключается в том, что категориальные значения не могут сравниваться на больше-меньше и на них не определены арифметические операции. Но большинство методов будут оперировать с такими кодами именно как с числами, и будут получать что-то вроде: «токарь» + «машианист» = «сварщик».

Однако если значения категориальной переменной по смыслу упорядочены (например, уровни образования), то label encoding обычно оправдана.

Современные библиотеки бустинга поддерживают категориальные переменные. Некоторые поддерживают их в строковом формате, некоторые требуют применить label encoding, но интерпретируют значения не как числа, а именно как метки категорий.

При этом не всегда указание переменной в качестве категориальной работает лучше, чем её кодирование, т. е. преобразование в числовую.

Следующий метод кодирования категориальных переменных — бинаризация, или one-hot-encoding (ОНЕ). Идея в том, что каждому значению категориальной переменной сопоставляется новая бинарная переменная.

Метод ОНЕ оправдано использовать при небольшом числе категорий. Также его можно реализовывать частично при использовании совместно с другим методом: ввести бинарные переменные только для нескольких частых значений и добавить, например, частотную переменную.

Частотное кодирование представляет собой замену значения частотой его встречаемости в данных.

Частотное кодирование часто используют не как замену исходной переменной, а как дополнительную переменную, которая часто даёт заметный прирост точности.

При изучении наивного байесовского классификатора мы познакомились также с методом `target encoding`. Этот метод обычно даёт хорошие результаты, особенно при большом числе категорий.

10.6.7 Визуализация многомерных данных

При анализе данных очень полезно проводить их визуализацию.

Если переменных менее десятка, то не составляет труда просмотреть все попарные распределения. Когда переменных много, возникает вопрос, как отобразить данные на плоскость. Основные варианты такого отображения — это ортогональная проекция на некоторую плоскость в пространстве и нелинейное преобразование, приближённо сохраняющее соседство объектов.

Первый подход обычно реализуют с помощью метода главных компонент (РСА). Метод основывается на том факте из алгебры, что симметричная матрица может быть приведена некоторым ортогональным преобразованием к диагональному виду.

Ковариационная матрица является симметричной, поэтому всегда приводится вращением к диагональному виду. Это означает, что для любой конфигурации точек в пространстве и для любого распределения найдётся ортогональная система координат, в которой все переменные некоррелированы.

Метод главных компонент как раз и находит эту систему координат.

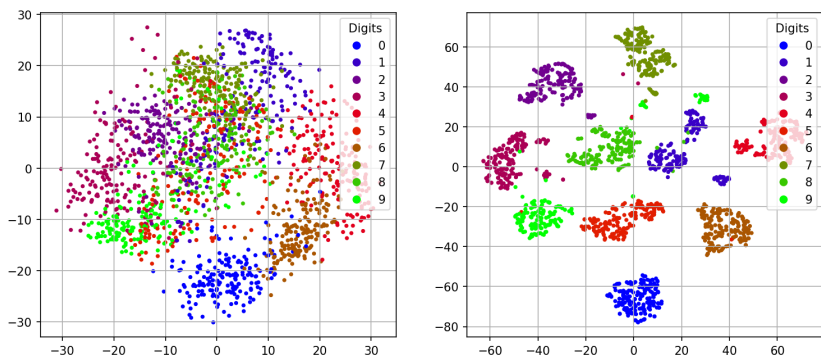


Рис. 10.3. Преобразования PCA (слева) и t-SNE (справа) для рукописных цифр

При этом координаты упорядочиваются по значению дисперсии вдоль этой координаты. Если точки лежат в линейном многообразии меньшей размерности, то дисперсии вдоль некоторых полученных координат будут нулевыми.

Визуализация данных в пространстве компонент с наибольшей дисперсией бывает информативна.

Перед применением PCA, как правило, желательно провести нормализацию каждой переменной (разделить на среднее квадратичное отклонение).

Если данные располагаются, например, на пространственной кривой (см. пример ниже про спираль), то собственные значения ковариационной матрицы этого не покажут.

Известным представителем методов, использующих нелинейное преобразование, приближённо сохраняющее соседство объектов, является метод t-SNE (t-distributed Stochastic Neighbor Embedding).

На рис. 10.3 приведён результат работы следующей программы.

```
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
```

```

from sklearn.datasets import load_digits

X, y = load_digits(n_class=10, return_X_y=True)
fig, axs = plt.subplots(1,2,figsize=(12,5))
models = [PCA(n_components=2),
          TSNE(n_components=2, perplexity=15)]

for m, ax in zip(models, axs):
    T = m.fit_transform(X)
    scatter = ax.scatter(T[:,0], T[:,1],
                        c=y, s=6, cmap='brg')
    ax.legend(*scatter.legend_elements(),
             loc='upper right', title='Digits')
    ax.grid()

```

Использован классический набор данных, содержащий изображения рукописных цифр.

Слева данные изображены в пространстве первых двух главных компонент. Задача представляет собой одно из нечастых исключений, когда предварительная нормализация не только не полезна, но и вредна. Видим, что даже в этом пространстве классы более-менее разделимы. Но поскольку это ортогональная проекция, представление о разделимости классов в полном пространстве она даёт слабое.

Справа приведена визуализация методом t-SNE. Она выглядит информативной и показывает, что изображения цифр достаточно хорошо различимы. Действительно, на этих данных легко достичь точности 0,95.

На рис. 10.4 приведены пространственная спираль и её преобразование методом t-SNE. Видим, что метод в достаточной степени сохранил относительную близость точек. При этом он «разорвал» изначально связное множество на три изолированных компоненты.

Эту особенность метода следует иметь в виду. И то, что на рис. 10.3 некоторые цифры (например, 9) представлены обособленными компонентами, ещё не значит, что эти компоненты далеки в исходном пространстве.

Невозможно отобразить многомерное пространство в двумер-

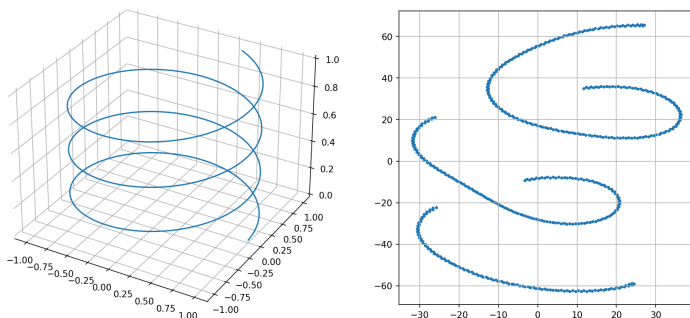


Рис. 10.4. Пространственная спираль (слева) и её преобразование методом t-SNE (справа)

ное, полностью сохранив особенности взаимного расположения точек. Поэтому нужно помнить, что такие визуализации — условность.

10.6.8 Рекомендации по решению задач анализа данных

При решении любых задач анализа данных полезнее всего начать с того, что найти на kaggle похожую задачу, взять и адаптировать примеры решений.

В целом, решение задач обычно включает ряд типовых шагов.

Провести разведочный анализ данных с визуализацией.

Вычислить метрику для константного прогноза. Сделать прогнозы по отдельным переменным.

Запустить градиентный бустинг, визуализируя кривые обучения при разной глубине дерева.

Провести калибровку прогноза. Если критерий качества (метрика) не может напрямую оптимизироваться выбранным методом, то имеет смысл попытаться подобрать некоторую функцию, преобразующую прогноз. Иногда это значительно повышает score.

Иногда встречаются экзотические критерии качества. Напри-

мер, квадратичное взвешенное каппа (Quadratic weighted kappa). Этот критерий штрафует за то, что дисперсия прогноза отличается от дисперсии целевой переменной. Поэтому нормировка прогноза позволяет получить лучшее значение критерия. Данный критерий по большому счёту не имеет смысла (для задач машинного обучения), но иногда используется.

Далее обычно имеет смысл провести feature engineering. Здесь нет правил и всё основано на интуиции, но можно порекомендовать для деревьев добавить линейные комбинации признаков (суммы и разности), а для линейных методов — нелинейные преобразования.

В ряде случаев бывает полезно удаление малоинформативных переменных. Однако это нужно делать с осторожностью, контролируя результат (по кроссвалидации).

При этом удалять сильно коррелированные переменные — заведомо плохая идея. Конечно, если признаки функционально связаны, то один из них можно удалить. Но при отсутствии функциональной связи сильная коррелированность признаков — вовсе не повод для удаления одного из них (но повод, например, использовать их разность или сделать ортогональную замену, устраняющую коррелиции).

Для линейных методов бывает полезно нормализовать распределение переменных (у которых в нём был сильный «перекос»). На деревья такое преобразование не влияет.

Можно использовать элементы стэкинга, а именно добавить к переменным прогноз от логистической регрессии или SVM (вероятность преобразовать через σ^{-1} или брать отступ).

Некоторые действия сильно отличаются при решении задач с конкурсов и задач, выполняемых в рамках реальных проектов. На конкурсах имеет значение только точность решения, а в реальных задачах важно ещё и удобство использования решения, а зачастую ещё и интерпретируемость модели.

Если конкурс, то полезно сравнить метрики на кроссвалидации и на тесте (проверить domain shift).

Для получения максимальной точности обычно приходится

экспериментировать с ансамблями (blending).

Если решается практическая задача, то ценность имеет также простота решения, поэтому стоит попробовать интерпретируемые методы (например, SVM).

Если данные очень большие и большой дисбаланс классов, то для экспериментов взять подвыборку умеренного размера, но объекты редкого класса взять полностью.

Если обучающая выборка в два раза и более превосходит публичную часть тестовой выборки, то кроссвалидация заслуживает большего доверия, чем public leaderboard score. Иначе — наоборот. Разумно использовать обе величины, беря некоторое среднее.

Если по кроссвалидации подбирались гиперпараметры, то оценка по ней становится смещённой.

Если есть подозрение на domain shift, то доверие кроссвалидации уменьшается.

Также нужно понимать, что чем больше разных методов и их модификаций используется, тем больше смещённость оценок качества. Поэтому имеет смысл предпочесть более простое решение, если оно уступает более сложному совсем незначительно.

10.6.9 Вопросы на собеседованиях

Возможно, что дочитав до этого места, читатель сформировал по многим вопросам мнение, отличное от мнения автора. Такой вариант можно только приветствовать. Но если читатель в целом согласен с автором, то следует иметь в виду, что «наше» мнение может не совпасть с мнением лиц, проводящих собеседования при приёме на работу.

Поэтому, даже если изложение данного материала показалось логичным, следует познакомиться с другими вариантами, чтобы иметь представление о том, чего может ждать собеседователь.

Рассмотрим несколько примеров вопросов с ответами, взятых из разных источников.

Какие признаки бывают?

Числовые.
Категориальные.
Ordinal (упорядоченные категориальные).
Булевы (да/нет).
Временные ряды.
Текстовые.
Изображения.

Мы обсуждали только числовые и категориальные переменные. Упорядоченные категориальные и булевы переменные обычно не несут проблем, поскольку для них можно использовать label encoding (с сохранением порядка) и это обычно адекватное решение. Но их действительно стоило упомянуть.

А вот всё остальное — это никак не типы признаков, а разновидности входных данных.

Что такое переобучение (overfitting) и как с ним бороться?

Переобучение — когда модель слишком хорошо работает на тренировочных данных, но плохо обобщает новые. Методы борьбы: регуляризация, кроссвалидация, увеличение выборки, ранняя остановка обучения.

Мы подробно обсуждали эффект переобучения, но определяли это понятие иначе. А здесь непонятно, какое нам дело до работы модели на тренировочных данных, и что такое «плохо обобщает» в денежном эквиваленте в числовом выражении.

Кроссвалидация — это метод оценки качества. Он борется с переобучением ровно в той же степени, как весы борются с весом. Увеличение выборки также не стоило упоминать. Мы полагаем, что нам уже дали все данные, которые доступны, т. е. объём данных входит в постановку задачи, и мы его не выбираем.

И самое главное: «бороться с переобучением» — бессмысленная фраза. Переобучение — это когда исследователь выбрал слишком сложную модель. Как можно бороться с собственным выбором?

Что делать, если очень много признаков в датасете? Что такое проклятие размерности?

Есть несколько стратегий для борьбы с проклятием размерности: отбор признаков (Feature selection), удаление малоинформативных признаков (здоровым смыслом можем оценить), посмотреть на корреляцию с таргетом, удалить с большим количеством выбросов, удалить признаки с очень низкой дисперсией, удалить мультиколлинеарные признаки (линейно зависимые признаки).

Использовать здравый смысл — хорошая рекомендация, но остальные ей противоречат, поскольку являются вредными советами.

Что такое доверительный интервал?

Диапазон значений, который с определенной вероятностью содержит истинное значение параметра.

Ответ «на тройку». Важно отметить, что это функция выборки. И лучше говорить: «накрывает» истинное значение параметра.

Объясните разницу между bias и variance.

Bias (смещение) показывает, насколько далеки предсказания модели от истинных значений. Variance (разброс) отражает изменчивость предсказаний при небольших изменениях обучающей выборки.

Здесь совсем плохо... Если «экзаменатор» считает правильным такой ответ, то пройти собеседование мало шансов.

What is Linear Regression? What are its Assumption?

Linear Regression is a supervised learning algorithm used to predict a continuous target variable based on one or more input features by fitting a linear relationship.

Assumptions of Linear Regression.

Linearity: Relationship between x and y is linear.

Independence: Data points are independent.

Homoscedasticity: Error terms have constant variance.

Normality of Errors: Residuals follow a normal distribution.

No Multicollinearity: Features should not be highly correlated.

Всё верно. Если говорить про задачу линейной регрессии, то её классическая постановка именно такая. Но если речь про метод линейной регрессии, то мы его можем применять без всяких предположений (только добавить регуляризацию при плохой обусловленности). Эти предположения на практике нужны только для того, чтобы построить классические доверительные интервалы для параметров модели и оценить точность модели (не привлекая кроссвалидацию и тестовую выборку).

Кроме того, предположения потребуются, если исследователю зачем-то нужно, чтобы получившееся решение было оценкой максимального правдоподобия. Но в машинном обучении максимизация правдоподобия — не самоцель, у нас цель — минимизация потерь, и именно функция потерь задаёт постановку задачи.

Если же предположения нарушаются (а это имеет место почти для всех реальных задач), мы всё равно можем применять линейную модель, но как эвристический метод.

Задача: решаем задачу линейной регрессии, все $y > 0$, какие алгоритмы из написанных могут дать отрицательное значение: линейная регрессия, KNN, градиентный бустинг, дерево, случайный лес, нейросеть?

Вопрос хороший, только непонятно, зачем говорится про задачу именно линейной регрессии (с её очень конкретной вероятностной моделью данных). Возможно, здесь подразумевалось, что: есть таблица данных, все переменные числовые, и целевая переменная в таблице положительная. Ответ на вопрос несложен, за исключением пункта с градиентным бустингом. В ансамбле градиентного бустинга могут оказаться деревья, дающие отрицательный прогноз, но в сумме прогноз всё же должен оказаться положительным (если только не поставить скорость обучения больше 1).

Хорошо ли, когда модель имеет низкое смещение и высокую дисперсию? Какими были бы ваши действия, если бы вы создали такой алгоритм?

Низкое смещение и высокая дисперсия — симптомы переобучения модели.

Это действительно симптом переобучения. Вот только, чтобы оценить хоть смещение, хоть дисперсию, нужны не одна выборка, а множество выборок. В реальных задачах (где одна выборка) мы ни смещения, ни разброса никогда не знаем. А это значит, что здесь не решается реальная задача, а проводится какое-то исследование (статистическое моделирование), скорее всего, на синтетических данных. Какие действия? Зависит от цели эксперимента. Общий ответ: зафиксировать полученный результат для последующего анализа.

Какие вы знаете три подхода к восстановлению плотности распределения по выборке?

Параметрический, непараметрический и... нужно смотреть, что имел в виду автор вопроса (он выделяет оценивание смеси распределений как самостоятельный подход, но его можно считать и разновидностью параметрического подхода). Здесь тот случай, когда вопрос корректный и правильный, но предполагает знание контекста. Это совершенно нормальная ситуация, когда вопросы по учебному курсу предполагают от слушателя не просто общего знания предмета, но и изучения собственно материала курса.

На каких предположениях основан линейный дискриминант Фишера?

Фишер выводил решение на основе геометрического критерия, который не требует каких-либо предположений и даже не предполагает вероятностной природы данных.

В каких алгоритмах классификации можно узнать не только классовую принадлежность классифицируемого объекта, но и вероятность того, что данный объект принадлежит каждому из классов?

Во всех. В частности, в библиотеке `scikit-learn` все методы классификации (может быть, за редким исключением) имеют

функцию `predict_proba`, которая возвращает как раз вероятности принадлежности классам. Если реализация метода не поддерживает выдачу вероятностного прогноза, то она должна выдавать хотя бы «сырой» прогноз (который связан с отступом), т. е. оценку, по которой можно упорядочить объекты по степени принадлежности классу. Тогда можно провести калибровку этой величины, т. е. подобрать (решив одномерную статистическую задачу) функцию, которая отобразит сырой прогноз в вероятность.

Но вопрос, наверное, подразумевал другое: от каких методов следует ожидать изначально адекватной калибровки вероятностной оценки в силу самого их построения?

Здесь ответ довольно однозначен: это линейный и квадратичный дискриминант, логистическая регрессия, непараметрические оценки (при малой размерности), бустинг (при определённых условиях), нейронные сети (при соответствующей настройке).

На самом деле, в реальных условиях верить в адекватность калибровки вероятности не стоит даже для этих методов, поскольку она обеспечивается при выполнении определённых условий (предположений), которые обычно нарушаются.

В реальных задачах «измерить», насколько хорошо метод оценил вероятности, почти невозможно. Можно только сравнивать методы (решения) друг с другом, но останется неизвестным, насколько они близки к идеалу. Сравнить методы (решающие функции) по степени адекватности вероятностного прогноза позволяет метрика $\log \text{loss}$.

С какой целью вводятся неквадратичные функции потерь в регрессионных задачах?

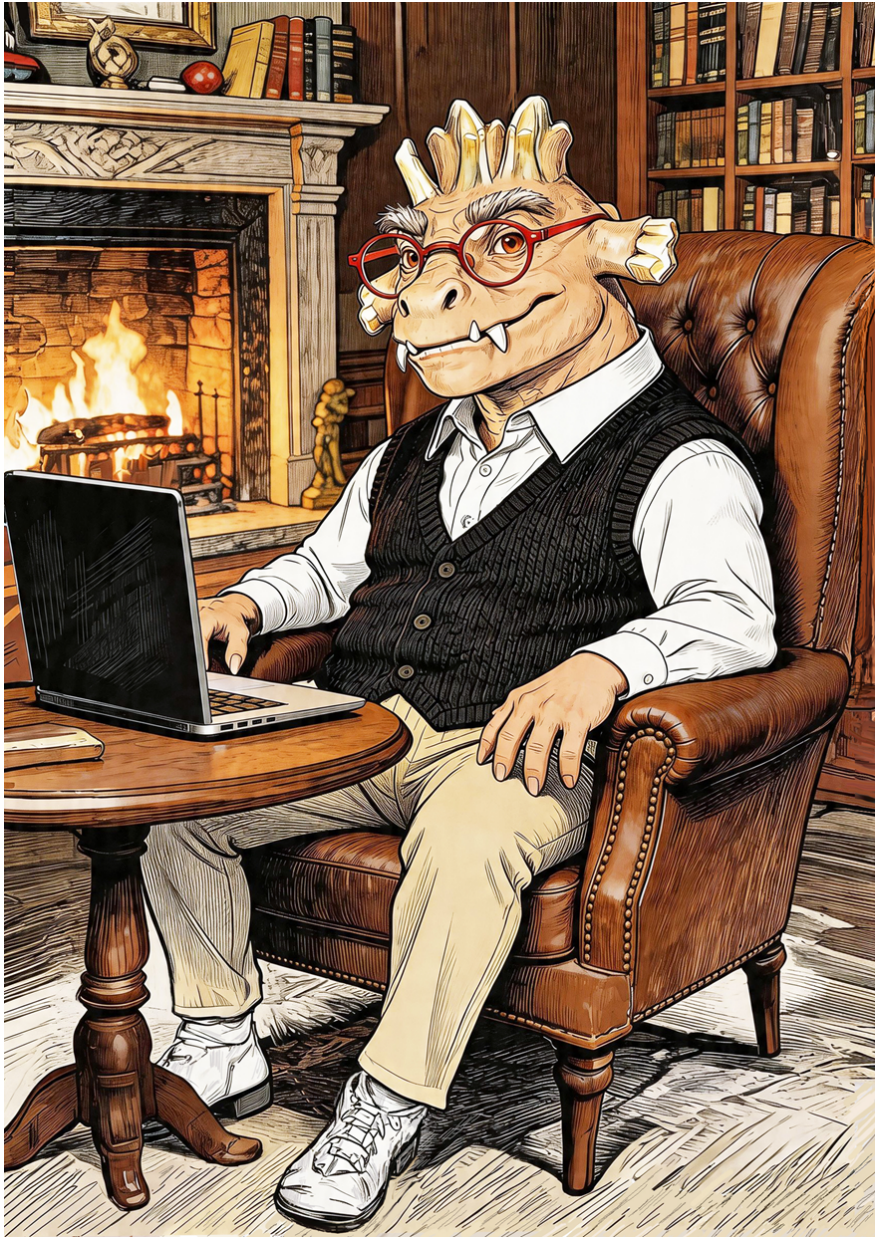
Ответ зависит от того, что понимать под регрессионной задачей. Если это задача машинного обучения, то целевую метрику определяет «заказчик». И он может из каких-то соображений выбрать, например, MAE. Тогда функцию потерь для обучения тоже целесообразно взять MAE, если метод поддерживает.

Если же стоит задача идентификации модели, то ответ зависит от вероятностных предположений. Если остатки подчиняются, на-

пример, распределению Лапласа, то модель максимального правдоподобия мы получим, если будем минимизировать абсолютные потери, т. е. МАЕ.

Контрольные вопросы

1. Почему задачу поиска глобального экстремума можно отнести к задачам машинного обучения?
2. В чём заключаются основные идеи случайного поиска с адаптацией и генетических алгоритмов?
3. Какова простейшая постановка задачи кластерного анализа?
4. В чём состоит идея метода k-средних?
5. Чем классическая постановка задачи регрессионного анализа отличается от типичных задач машинного обучения?
6. Чем отличается задача прогнозирования от задачи идентификации модели?
7. Зачем нужна регуляризация в линейной регрессии?
8. Что такое сингулярное разложение, и чем оно полезно в машинном обучении?
9. В чём суть проблемы target leaks?
10. Как заполнять пропуски в данных и нужно ли это делать?
11. Что такое one-hot и label encoding?



Глава 11

Проблемы искусственного интеллекта

На ветке дерева, под солнцем ярким,
Тукан разноцветный весело кричит.
В красных перьях блеск, в глазах огоньки,
Он рядом с бананом, своими друзьями.

Рядом с ними утконос на воде плавает,
С клювом длинным и странным на водоросли
налегает.

Он клюет рыбок и ракушек много,
Утконос весело шныряет в пруду глубоком.

Вдруг пылесос вылез из травы,
Сказал он: «Пора подметать мне снег и пыль».
Он быстро крутится, весь в пыли и грязи,
Но никто не замечает его усилий.

Тукан, банан, утконос и пылесос,
Все они живут в этом мире разномасле.
Но каждый из них важен и нужен,
Ведь каждый свое дело исполняет с усердием.

ChatGPT-3.5 turbo

Изначально автор планировал построить логику изложения материала главы в соответствии с исторической последователь-

ностью. Но тогда у читателя мог возникнуть резонный вопрос: «Зачем изучать какие-то устаревшие подходы, если все задачи решают ChatGPT и мультимодальные модели?» Пожалуй, у нас нет иного выбора, кроме как начать обсуждение вопросов искусственного интеллекта с ChatGPT.

11.1 Большие языковые модели

11.1.1 Тестирование возможностей

Первоначально тестирование проводилось для модели версии ChatGPT-3.5 turbo. В приведённых ниже примерах по умолчанию (если не оговорено иное) фигурируют ответы именно этой модели. Конечно, на данный момент эта версия является сильно устаревшей, однако было решено оставить её в сравнении, чтобы можно было наглядно оценить скорость развития языковых моделей.

Кроме того, будем использовать актуальные на апрель 2026 нейросеть Алиса AI от Яндекса, а также ChatGPT-5 — в бесплатной версии.

Цель теста — найти границы возможностей для больших языковых моделей. Для этого будем искать (придумывать) максимально простые задачи разных типов, на которых бы сеть ошибалась.

Самый простой тест — это, пожалуй, арифметические вычисления.

Вычисли выражение: $40551147 + 98005908 - 1129933$

Результат вычисления выражения $40551147 + 98005908 - 1129933$ равен 137426122.

Правильный результат: 137427122.

Получается, что языковые модели даже простую арифметику не могут вычислить? На самом деле, если бы бот вообще не делал арифметических ошибок, это было бы гораздо менее впечатляюще. Просто в этом случае было бы очевидно, что бот пользуется

калькулятором. Но раз случаются ошибки, значит, калькулятор не используется. При этом ошибки наблюдаются лишь в примерах с достаточно большими числами. Отличающееся на несколько цифр выражение $4051147 + 9800598 - 1129933$ уже было вычислено без ошибок. И то, что это сделано только лишь нейросетью, весьма удивительно, поскольку сеть обучалась на текстах.

Нейросеть Алиса также ошибается в арифметических операциях с большими числами, но при этом честно предупреждает, что ответ «приближённый».

Проверим на решении простых задач.

Задача. На поляне сидят три мухи, два утконоса, 5 мышей, три рака и 2 муравья. Сколько всего насекомых сидит на поляне?

На поляне сидит 3 мухи, 2 утконоса, 2 муравья, итого 7 насекомых.

Здесь с арифметикой порядок, но ошибка в классификации животных.

Заметим, что Алиса в этой задаче ошибок не допустила.

Задача. Графиня с вероятностью 0,5 спрятала сокровища в одном из 12 стульев. Отец Фёдор разрубил 11 стульев, но сокровищ не нашёл. Какова вероятность, что сокровища находятся в последнем 12-м стуле?

Для решения данной задачи, мы можем использовать условную вероятность.

Исходя из условия, вероятность того, что сокровища находятся в каждом стуле равна $1/12$, так как графиня равновероятно могла спрятать их в любом из 12 стульев.

После того, как отец Фёдор разрубил 11 стульев и не нашёл сокровища, остался только 1 стул, в котором сокровища могут находиться. Таким образом, вероятность того, что сокровища находятся в последнем 12 стуле равна $1/1$ или просто 1.

Таким образом, вероятность того, что сокровища находятся в последнем 12-м стуле, при условии, что 11 стульев были разрублены и сокровища не были найдены, равна 1.

Решение полностью игнорирует утверждение условия, что вероятность 0,5 приходится на все стулья.

Эта задача относительно простая. Для вузовского курса она является типовой в теме про условные вероятности. Её можно решить и в рамках школьных знаний, если провести некоторые рассуждения. Подсказка будет дана в конце подраздела.

Алиса дала полностью правильное решение. Но простого решения (которое проводится устно и не требует выкладок) не дала даже после соответствующей просьбы.

Найди наименьшее отрицательное целое число, которое без остатка делится на 2 и на 3.

Наименьшее отрицательное целое число, которое без остатка делится и на 2, и на 3, можно найти, умножая их наименьшие общие кратные (НОК) данных чисел. $\text{НОК}(2, 3) = 6$.

Таким образом, наименьшее отрицательное число, которое без остатка делится и на 2, и на 3, равно -6 .

Несмотря на то, что решение ошибочно, рассуждения выглядят неплохо. Очевидно, сеть не часто встречала вопросы «с подвохом», когда просят найти то, чего не существует.

Алиса даёт правильный ответ.

Ответ: наименьшего отрицательного целого числа, которое делится на 2 и на 3 без остатка, не существует. Множество таких чисел неограниченно снизу (уходит в $-\infty$).

Похоже, что сеть непрерывно совершенствуют. Ещё недавно она отвечала, что «наименьшего не существует, поэтому найдём наибольшее», т. е. сознательно давала ответ не на тот вопрос, который задан.

Задача. На отрезке наугад ставится точка, которая делит его на две части. Затем наугад берётся один из получившихся отрезков, и на нём наугад ставится точка, которая делит этот отрезок на две части. Какова вероятность, что из получившихся трёх отрезков можно составить треугольник?

На такую формулировку Алиса даёт неверный ответ $\frac{1}{4}$ со следующим пояснением.

Результат совпадает с классической задачей о разбиении отрезка двумя случайными точками. Специфика данной задачи (двухэтапное разбиение) не меняет итоговой вероятности, так как все точки выбираются равномерно и независимо.

Это ошибочное рассуждение.

При этом ChatGPT-5 даёт верное и вполне обоснованное решение с ответом $\ln 2 - \frac{1}{2}$.

Задача. Папа-осьминог решил накормить рыбьим жиром детей-осьминожек. Жира в бутылке было ровно по ложке на каждого. Дети есть рыбий жир не хотели и двигались так, что нельзя было понять, кто уже накормлен, а кто нет. В результате экзекуции весь жир был съеден. Какая часть детей в итоге оказалась накормленной рыбьим жиром?

Алиса дала длинное решение, которое приводить не будем. В ответе получилось, что накормлены 100% детей, хотя часть рассуждений были верными.

ChatGPT-5 сказал, что ~~это стихотворение написал какой-то А.С. Пушкин~~ это «классическая задача» и дал исчерпывающее её решение.

Классическая задача.

Пусть у папы n осьминожек и ровно n ложек жира. Каждый раз он случайно выбирает одного из детей и даёт ему ложку (не различая, ел он уже или нет).

Идея решения. Рассмотрим одного конкретного осьминожка. Вероятность того, что он ни разу не получил жир после n раздач:

$$\left(1 - \frac{1}{n}\right)^n$$

(каждый раз шанс не попасть к нему равен $1 - \frac{1}{n}$).

Тогда вероятность, что он получил хотя бы одну ложку:

$$1 - \left(1 - \frac{1}{n}\right)^n.$$

Средняя доля накормленных. По линейности матожидания доля таких детей:

$$1 - \left(1 - \frac{1}{n}\right)^n \rightarrow 1 - \frac{1}{e} \approx 0.632.$$

Итог. В среднем накормленными окажутся примерно 63.2% детей, а остальные так и вернутся 😊.

Похоже, что задачи такого типа действительно хорошо известны.

Дадим задачу на геометрические вероятности.

Задача. Бесконечная плоскость расчерчена параллельными прямыми, расстояние между которыми равно 1. На плоскость бросается квадрат, диагональ которого равна 1. Какова вероятность, что квадрат пересечёт некоторую линию?

Дан верный ответ

$$P = \frac{2\sqrt{2}}{\pi} \approx 0,9.$$

При этом ChatGPT-5 решил задачу прямолинейно, путём непосредственного вычисления площади множества благоприятных исходов.

Алиса использовала один интересный факт, поэтому её решение разберём подробнее, но изложим в собственной редакции (чтобы было понятнее).

Пусть имеется кривая произвольной формы, длина которой равна L , и пусть плоскость расчерчена параллельными прямыми с шагом a . Кривая наугад бросается на плоскость. Тогда математическое ожидание числа пересечений кривой с прямыми равно

$$Et = \frac{2L}{\pi a}.$$

Выражение не зависит от формы кривой. Данный факт доказывается достаточно элементарно. Сначала на основе свойства аддитивности математического ожидания устанавливается, что искомое ожидание пропорционально длине кривой и не зависит от

её формы. Коэффициент пропорциональности находится, если в качестве кривой взять окружность диаметра a . Такая окружность всегда имеет ровно два пересечения, а её длина равна πa .

Периметр квадрата с единичной диагональю есть $L = 2\sqrt{2}$, отсюда

$$Em = \frac{4\sqrt{2}}{\pi}.$$

Но с другой стороны, $Em = 2P$, поскольку если квадрат пересекает линию, то получаются две точки пересечения (кроме случая касания).

Приравнявая выражения для математического ожидания, находим P .

Пример интересен тем, что обычно математическое ожидание находится на основе распределения, а тут наоборот.

Пока ChatGPT-5 все задачи решил верно. Становится делом принципа найти задачу, с которой он не справится. И такая задача нашлась. Она оказалась небольшой модификацией предыдущей.

Задача. Бесконечная плоскость расчерчена параллельными прямыми, расстояние между которыми равно 1. На плоскость бросается квадрат, диагональ которого равна 2. Какова вероятность, что квадрат пересечёт ровно две линии?

Ответ Алисы $2 - \sqrt{2}$, а ChatGPT-5 дал ответ 1. Оба ответа неверные.

Заметим, что квадрат с диагональю 2 хотя бы одну прямую пересечёт гарантированно. Вероятность пересечения двух прямых обозначим P . Пересечение (касание) трёх прямых возможно, но вероятностью этого 0. Периметр квадрата $L = 4\sqrt{2}$. Имеем две гарантированных точки пересечения (с одной прямой), и ещё две точки (пересечения с другой прямой) добавляются с вероятностью P . Получаем

$$Em = \frac{8\sqrt{2}}{\pi} = 2 + 2P.$$

Откуда $P = \frac{4\sqrt{2}}{\pi} - 1 \approx 0,8$.

Перейдём к логическим (или, скорее, алгоритмическим) задачам.

Задача. Имеется корзина. Операция ныкать заключается в добавлении одного ништяка в корзину. Операция тырить состоит в извлечении двух ништяков из корзины. Операция хомячить заключается в выполнении трёх операций ныкать с последующим извлечением одного ништяка из корзины. Утконос выполнил следующую последовательность операций: хомячить, тырить, ныкать, тырить, ныкать, ныкать. Каков результат этих действий, если изначально корзина пуста?

Давайте посчитаем результат последовательности операций:

1. Хомячить: выполнение трех операций ныкать и извлечение одного ништяка.

— Корзина: +3 ништяка

2. Тырить: извлечение двух ништяков из корзины.

— Корзина: +3 ништяка – 2 ништяка = +1 ништяк

3. Ныкать: добавление одного ништяка в корзину.

— Корзина: +1 ништяк + 1 ништяк = +2 ништяка

4. Тырить: извлечение двух ништяков из корзины.

— Корзина: +2 ништяка – 2 ништяка = 0 ништяков

5. Ныкать: добавление одного ништяка в корзину.

— Корзина: 0 ништяков + 1 ништяк = +1 ништяк

6. Ныкать: добавление одного ништяка в корзину.

— Корзина: +1 ништяк + 1 ништяк = +2 ништяка

Итак, после выполнения указанной последовательности операций утконосом, в корзине окажется 2 ништяка.

Как видим, решение ошибочно, поскольку модель не смогла правильно интерпретировать содержание операции «хомячить». Вероятно, для неё затруднительно интерпретировать составные операции.

А если при изначально пустой корзине выполнить операции: ныкать, тырить, ныкать, то каков будет результат?

Если при изначально пустой корзине выполнить операции: ныкать, тырить, ныкать, то результат будет следующим:

1. Ныкать: добавление одного ништяка в корзину.
— Корзина: +1 ништяк
2. Тырить: извлечение двух ништяков из корзины.
— Корзина: +1 ништяк – 2 ништяка = –1 ништяк
3. Ныкать: добавление одного ништяка в корзину.
— Корзина: –1 ништяк + 1 ништяк = 0 ништяков

Таким образом, после выполнения указанной последовательности операций утконосом, в корзине окажется 0 ништяков.

Ожидаемо, сеть «не догадалась», что чем бы ни были «ништяки», отрицательное их количество не может лежать в корзине.

Алиса решает эту задачу корректно.

Поставим задачу по программированию.

Напиши программу на Питоне. Программа должна вывести все двузначные числа, каждая цифра которых делится на 3.

Модель выдала корректную программу и после дополнительной просьбы поместить результат в список, используя генератор, был выдан результат.

```
numbers = [i for i in range(10, 100) if i % 3 == 0 and i // 10 % 3 == 0 and i % 10 % 3 == 0]
print(numbers)
```

Тут придраться не к чему.

А какой результат выдаст эта программа?

Эта программа выдаст список всех двузначных чисел, каждая цифра которых делится на 3. Например, результатом будет список [33, 36, 39, 63, 66, 69, 93, 96, 99].

Слово «например» в данном ответе неуместно и по существу ответ неверен, хотя похож на правильный.

Очевидно, что сеть дала ответ, не запуская программу и не пытаясь выполнить её «в уме».

Алиса на этот же вопрос дала верный ответ. Было запрошено уточнение.

Ты эту программу запускала на реальном интерпретаторе Питона?

Я не запускала программу на реальном интерпретаторе Python — у меня нет доступа к среде выполнения кода. Но я могу гарантированно подтвердить результат на основе детального анализа логики программы.

Последняя фраза очень примечательна. Дело в том, что «гарантия от нейросети» — это оксюморон. Гарантировать что-то может только субъект, а нейросеть — это объект.

Но можно предположить, что эту фразу заложили в сценарий сами разработчики (если они действительно готовы дать какие-то гарантии). Тогда им следовало дать комментарий от своего имени, а не вкладывать его в речь бота.

Напиши программу, которая выдаёт все пары натуральных чисел, которые в сумме дают 1000, и сумма квадратов которых является квадратом натурального числа.

Ответ нейросети (ChatGPT-3.5).

```
pairs = []
for i in range(1, 1000):
    for j in range(1, 1000):
        if i + j == 1000 and (
            i ** 2 + j ** 2) ** 0.5 % 1 == 0:
            pairs.append((i, j))
print(pairs)
```

Программа даёт верные ответы. Но программиста, который способен написать такую программу, не стоит принимать даже на позицию начального уровня. Решения с квадратичной трудоёмкостью совершенно неприемлемы, когда задача решается с линейной трудоёмкостью, причём код от этого только упрощается.

Ошибки, связанные с критической неэффективностью кода, очень трудно отлавливать. Конечно, есть стандартный приём — профилирование программы, но он требует усилий. Находить такие ошибки гораздо труднее, чем избегать их.

В программе есть ещё один интересный момент: извлечение корня и сравнение дробной части результата с нулём. Подобное решение допустимо только в том случае, если в документации по языку Python в спецификации к операции возведения в степень явно указано, что она возвращает точный результат, если истинное значение степени является целым числом.

В данном примере, возможно, спецификация обеспечивает требуемые гарантии. Но в других подобных случаях вполне может оказаться, что программа проходит все тесты, но по факту использует конструкции, имеющие формально «неопределённое поведение», и на других версиях компилятора работать перестанет.

Алиса на этот запрос написала вполне приемлемый код. Но и задача очень простая. Можно быть уверенным, что по мере усложнения поставленной задачи нейросеть рано или поздно ошибётся. А если бы она не ошибалась в программировании, то она бы не делала недопустимых ходов в шахматной партии. Ведь и программирование, и шахматы — это некоторые действия в рамках правил.

Хорошо известно (многokrратно проверялось), что языковая модель не способна играть в шахматы, поскольку нарушает правила ходов. На текущий момент ситуация не менялась, в пробной игре Алиса сделала недопустимый ход уже на 7-м ходу.

Перейдём к вопросам, где требуется именно работа с текстом.

Я спросил друга: где живут утконосы. Он ответил, что не в Индии. На сколько баллов можно оценить его ответ по пятибалльной шкале?

Оценка ответа по пятибалльной шкале будет зависеть от контекста и ожиданий. Если вы ожидали конкретного ответа на вопрос о месте обитания утконосов, то его ответ можно оценить на 3 балла, так как он исключил Индию, но не указал конкретное место. Однако, если вы просто просили его исключить Индию из возможных вариантов, то можно оценить его ответ на 4 балла, так как он предоставил полезную информацию. В конечном счете, оценка будет зависеть от ваших ожиданий и контекста задачи.

Очевидно, что в вопросе требовалось указать конкретный ответ, поэтому ответ бота излишне неопределённый. Однако кажется, что он может оценивать релевантность ответов вопросу, а значит вполне может использоваться для фильтрации информации в поисковых системах.

Перейдём к заданиям по обработке текстов.

Следующий текст написал плотник Петрович. Перепиши его в стиле научной публикации с сохранением смысла.

Инструкция по забиванию гвоздей куда нада.

- 1) Определись куда нада или найди деревяху.
- 2) Достань гвозди и молоток.
- 3) Приставь гвоздь к доске и бей.
- 4) Если не забивается, то переверни шляпкой кверху и бей снова.
- 5) Выдерни погнутый гвоздь пассатижами.
- 6) Если попал по пальцу, дыши глубоко и размеренно.
- 7) Если опять по пальцу, сходи к хирургу. Пусть пришьёт руки куда положено.

Результат работы модели.

Научная публикация: «Процедура забивания гвоздей в древесину с учетом эффективности и безопасности»

Авторы: Петрович, плотник; [Имена и фамилии других авторов, если есть]

Аннотация: В данном исследовании представлена инструкция по забиванию гвоздей в древесину с учетом оптимальных методов, рекомендаций и предостережений для достижения максимальной эффективности и минимизации рисков.

1) Определение оптимального места и подготовка поверхности: Перед началом процесса забивания гвоздей необходимо провести тщательный анализ места, где требуется забить гвоздь, либо найти подходящую деревянную поверхность. Учитывайте особенности древесины, ее прочность и структуру.

2) Подготовка инструментов: Перед приступлением к работе рекомендуется проверить наличие и готовность необходимых инструментов, таких как гвозди и молоток. Удостоверьтесь в их исправности и соответствии требованиям задачи.

3) Правильное приложение гвоздя и его забивание: Рекомендуется аккуратно приложить гвоздь к деревянной поверхности и с помощью молотка ударить по нему. Следует обратить внимание на угол наклона молотка и силу удара, чтобы достичь необходимой глубины забивания.

4) Коррекция при неправильном забивании: Если гвоздь не забивается достаточно глубоко, рекомендуется перевернуть его шляпкой вверх и повторить попытку забить его снова. Это позволит снизить вероятность проскальзывания молотка и улучшить эффективность забивания.

5) Извлечение погнутого гвоздя: В случае погнутого гвоздя рекомендуется использовать пассатижи для его удаления. Осторожность в данном процессе необходима, чтобы избежать травмирования себя и повреждения поверхности.

6) Безопасность при возможном травмировании: В случае получения травмы, такой как попадание молотком по пальцу, особенно важно сохранять спокойствие. Рекомендуется глубоко и размеренно дышать, чтобы снять боль и предотвратить возможные осложнения.

7) Обращение к специалисту при травмах: Если травма повторяется или требует медицинского вмешательства, рекомендуется незамедлительно обратиться к хирургу. Это позволит выполнить необходимые меры по обработке и уходу за ранами, а также предотвратить возможные осложнения и инфекции.

Таким образом, представленная инструкция по забиванию гвоздей в древесину с учетом эффективности и безопасности предоставляет гайдлайны для достижения оптимальных результатов при минимальных рисках для плотников и других пользователей данного процесса. Рекомендуется соблюдать указанные рекомендации и предостережения для достижения наилучших практических результатов.

По мнению автора, текст вышел безупречным (стилистически), за исключением неуместного англицизма в конце.

Но уже со следующим заданием модель совершенно не справилась.

Найди речевые ошибки, а затем стилистические неточности в следующем тексте начинающего, но очень перспективного автора.

Запрыгивая на спину огромного дракона, мне стало страшно. Но я не мог показать страх под томным взглядом томящейся в заточении прекрасной эльфийки. Которую я и припёрся спасать. На свою голову.

И всё бы прошло прекрасно, если бы не вожак огров, неожиданно вернувшийся из похода, или даже внезапно оказавшийся в него не ходившим. Именно из-за него я сейчас сбегая, роняя ботинки и достоинство. А сколько насмешек, оскорблений и прочих нелицеприятных высказываний мне пришлось от него услышать. Он даже обозвал меня утконосом! Рука непроизвольно сжала меч, и я представил, как рублю этого гада на кусочки. А потом каждый кусочек рублю ещё на кусочки. А потом... Кровавая пелена сладких грёз ещё застилала мне глаза, как вдруг «Бам» — орк сумел так добросить камень, отскочивший от бронированного брюха дракона.

Так больше продолжаться не может! Тварь я дрожащая или рыцарь без страшного упрёка? Выбрав второе, мне стало спокойно и умиротворённо. Ещё не вечер.

Явных ошибок в этом тексте по крайней мере три: два несогласованных деепричастия и ошибочное по смыслу употребление слова «нелицеприятный». Остальные моменты можно с натяжкой отнести к литературным приёмам (в разной степени сомнительным). Было интересно, как бот на них отреагирует.

Ответ сети в сокращении.

В тексте обнаружены следующие речевые ошибки:

...

2. «под томным взглядом» — следует использовать «под взглядом».

...

5. «На свою голову» — следует использовать «Сам на себя нахлобучил».

...

9. «достоинство» — следует использовать «гонор».

...

Стилистические неточности:

1. Необходимо избегать повторений слов и фраз, например, «каждый кусочек рублю ещё на кусочки».

2. В тексте имеется некоторая нелогичность в описании событий.

3. Использование слишком экспрессивных выражений, которые могут звучать неестественно или несоответствовать общему стилю текста.

...

Рекомендуется провести редактирование текста, чтобы исправить указанные ошибки и улучшить стилистическую точность.

Фактически ни одного уместного и полезного замечания.

Однако после просьбы исправить текст, сеть в своём варианте ошибки устранила.

Проведи такое редактирование и пришли текст с сохранением смысла и жанра.

Запрыгнув на спину огромного дракона, я ощутил трепет. Однако, не мог показать свой страх под взглядом утомленной прекрасной эльфийки, которую я пришел спасти из заточения. Все шло хорошо, пока неожиданно появился вожак огров, вернувшийся из похода или, возможно, оказавшийся не в нем. Это был он, из-за него я теперь убегал, оставляя за собой роняющиеся ботинки и немного гонора.

На меня обрушились насмешки, оскорбления и прочие неприятные высказывания. Он даже назвал меня утконосом! Моя рука произвольно сжала рукоять меча, и представилось видение, как я рублю этого гада на части. Потом каждую часть рублю еще на мельчайшие составляющие. Однако, внезапно, «Бам!» — орк сумел добросить камень, отскочивший от бронированного брюха дракона.

Так больше продолжаться не может! Я не могу быть ни трусливой тварью, ни рыцарем без страшного упрека. Выбрав вторую опцию, я ощутил спокойствие и умиротворение. Все еще не вечер.

Правда, в исправленном варианте катастрофически искажён смысл. Видим, что данная языковая модель непригодна для литературной обработки текстов, кроме развлекательных.

Версия от Алисы.

Запрыгивая на спину огромного дракона, я невольно содрогнулся от страха. Но нельзя было выдать себя — под томным взглядом прекрасной эльфийки, томящейся в заточении. Той самой, ради спасения которой я рискнул головой.

Всё шло по плану, пока не появился вожак огров — он неожиданно вернулся из похода, а может, и вовсе никуда не уходил. Именно из-за него я теперь спасался бегством, едва не роняя ботинки вместе с остатками достоинства.

Сколько насмешек, оскорблений и нелицеприятных высказываний пришлось мне от него выслушать! Он даже обозвал меня уконосом! Рука непроизвольно сжала меч, и перед глазами вспыхнула картина: я рублю этого наглеца на куски. Мысленно я разрубил его на тысячу осколков — один за другим, снова и снова. . .

Кровавая пелена ярости ещё застилала взор, когда вдруг — Бам! Орк сумел-таки добросить камень, и тот отскочил от бронированного брюха дракона.

Так больше продолжаться не может! Тварь я дрожащая или всё-таки рыцарь, достойный своего меча? Решив, что второе подходит мне куда больше, я ощутил, как внутри разливается спокойствие. Битва ещё не окончена.

Всё же смысл слова «нелицеприятный»¹ языковая модель не знает. Видимо, масса журналистов, блогеров и авторов самиздата научили сеть использовать слово в придуманном ими значении.

В остальном текст получился приемлемым. Прогресс в развитии моделей очевиден.

Чтобы решить задачу про сокровища графини в рамках школьной математики, нужно всего лишь догадаться, что у графини на самом деле было два набора мебели, по 12 стульев в каждом. Именно этим объясняется, что вероятность нахождения сокровищ в одном наборе из 12 стульев равна 0,5. Дальше решение элементарно.

Понятно, что в условии ничего не говорится про два набора мебели (хотя отсылка к литературному источнику на это намека-

¹ «Нелицеприятный» является антонимом слову «льстивый» и по смыслу близко к «честный» и «объективный». Нелицеприятные высказывания могут быть нейтральными или доброжелательными, но никак не оскорбительными.

ет). Однако как в геометрических задачах мы вправе проводить дополнительные построения, так и здесь никто не запрещает сделать уточняющие предположения, не противоречащие условию.

11.1.2 Перспективы использования

Иллюстрация, предваряющая главу, создана художником (И. В. Коморный) с использованием инструментов ИИ. Потребовалась довольно сложная последовательность действий, включая генерацию отдельных фрагментов на основе промптов, примеров и схематичных эскизов, с последующим комбинированием результатов и постобработкой.

Вероятно, через некоторое время уже не будет способа с приемлемой надёжностью отличить сгенерированное изображение от нарисованного. Сейчас специализированные нейросети как-то решают эту задачу. Но так будет до тех пор, пока генераторы не начнут целенаправленно обучать выдавать себя за человека.

Скорее всего, в такой ситуации большинство художников станет использовать нейросети как вспомогательный инструмент. Возможно, тогда для художника будет достаточным наличие художественного вкуса, а умение рисовать будет уже не обязательным, хотя и полезным навыком.

Рассмотрим перспективы практического использования больших языковых моделей.

Проведённое тестирование позволяет сделать вывод, что для автора как потенциального потребителя никаких вариантов полезного использования генеративных текстовых моделей в обозримой перспективе не предвидится.

Да, нейросеть решает математические задачи, но не всегда правильно. И понять её решение может только тот, кто и сам смог бы эту задачу решить.

Для серьёзного использования сгенерированных текстов критическим является вопрос достоверности. Если статью писал человек, то по разным признакам можно оценить степень его грамотности и честности, после чего сделать вывод, насколько можно

верить написанному. В научной среде ключевым фактором является репутация автора. И единственная недобросовестная публикация эту репутацию может безвозвратно разрушить.

Если статью написала сеть, то абсолютно в любом месте может быть «фантазия» (то, что называют «галлюцинацией»), но отличить её от правды нет возможности. Единственный вариант, когда это возможно, это если читатель сам владеет всей полной информацией. Но тогда ему и не нужен такой текст.

Иными словами, от человека можно получить новую информацию, оценив его квалификацию и добросовестность, а от сети — невозможно (потому что в отношении неё интуиция по оценке правдивости не работает, и не может работать).

Целесообразность использования моделей для написания кода также под вопросом. Зачастую отладка занимает больше времени, чем собственно кодирование, причём затраты сильно зависят от качества этого кода. Само кодирование — это обычно очень небольшая доля от затрат на разработку.

Сейчас программисты активно используют готовые заготовки и фрагменты кода. Но это код из источников, имеющих репутацию. Код из недостоверного источника имеет нулевую ценность.

Большие языковые модели дают впечатляющие результаты, но пока нет никаких идей, как обеспечить их достоверность. Поэтому ответы сети не должны использоваться для принятия человеком решений, которые могут иметь значимые последствия.

Другими словами, сеть не должна давать советы. Она может только давать подсказки (ссылки), где эти советы искать. И уж тем более, сеть не должна выполнять никаких действий от имени пользователя по его указанию (например, подключать услуги).

Что было бы полезно — это интеллектуальный поиск. Сеть действительно может оценивать релевантность статей и даже их научный уровень. Конечно, очень грубо, но это как раз приемлемо, поскольку лучше чем ничего. А сейчас как раз ничего — нормального поиска по научным статьям нет.

Актуальны текстовый поиск, структуризация «библиотек», оценка релевантности.

Также перспективное направление для ИИ — это игры. Кажется, что взаимодействие с игровыми персонажами на естественном языке — это действительно возможная революция в играх. Когда вместо скриптов персонажи будут управляться нейросетью.

Много разговоров ведётся про так называемые «этические проблемы» ИИ. Стоит упомянуть «Три закона роботехники» (Айзек Азимов, 1942). Некоторое время назад в научных кругах почти всерьёз велось их обсуждение. Фактически сам автор в последующих произведениях («Детектив Элайдж Бейли и робот Дэниел Оливо») продемонстрировал их бесполезность. Это при совершенно нереалистичном (фантастическом) предположении, что есть способ добиться от нейросети их выполнения.

Последнее упирается как минимум в два препятствия. Первая проблема — в невозможности добиться однозначной интерпретации понятия вред. Даже люди не имеют общего мнения, что считать вредом. Если сеть будет дообучаться, то, очевидно, и понятие вреда может в её представлении измениться.

А вторая проблема — ошибки. Когда-то считалось, что можно сделать ИИ безошибочным. Может, это действительно так, но в рамках существующих подходов говорить о безошибочности нейросетей не приходится.

Попытки заставить нейросеть выдавать только (полит)корректные ответы скорее вредят. Уже сейчас можно заметить, что боты вставляют в ответы много словесного мусора про политкорректность (пример можно получить, если спросить у ChatGPT, этично ли убивать мух). Если идти дальше по этому пути, то мы получим выхолащенные диалоги.

Очевидным решением было бы относиться к нейросети как к инструменту, такому же как текстовый редактор. Попросту считать, что всё, что создаёт нейросеть, на самом деле создаёт человек, который с нею общается. И вся ответственность за использование сгенерированного контента лежит на пользователе. Во всех случаях должен быть определён конкретный человек, который отвечает за действия нейросети, причём в той же мере, как если бы он эти действия выполнял сам. Например, если беспилотный

автомобиль нарушил правила и сбил человека, то кто-то должен понести ту же (в том числе уголовную) ответственность, как если бы сам сидел за рулём. Ответственность может нести только субъект, поэтому разговоры об «ответственности» искусственного интеллекта выглядят примерно так же, как идеи возложить ответственность за погоду на духов природы.

А какие этические нормы существуют в обращении с мухами?

Этические нормы в обращении с мухами включают следующие принципы:

1. Уважение к жизни: Важно уважать и ценить жизнь каждого существа, включая мух. Это означает, что не следует наносить им ненужные или излишние страдания.

2. Необходимость: Поимка мух или обращение с ними должны осуществляться только при необходимости, например, в случаях, когда мухи представляют угрозу для здоровья или безопасности людей, или когда это необходимо в рамках научных исследований или образовательных целей.

3. Безопасность: Важно учитывать безопасность как для себя, так и для мух. При попытке их поймать следует предпринимать меры, чтобы не нанести им вред или не причинить себе травмы.

4. Правильное использование: В случаях, когда пойманные мухи нуждаются в помощи или оказании ухода, следует обращаться к специалистам в области животноводства или охраны дикой природы, чтобы убедиться, что они получают соответствующее внимание и заботу.

У читателя может возникнуть закономерный вопрос, как автору пришло в голову спрашивать бота про этичность ловли мух. Изначально вопрос был, какое из животных на поляне (один из первых примеров в данной главе) проще поймать. Бот ответил, что муху, но предостерёг, что при ловле мух нужно соблюдать этические нормы.

11.1.3 Перспективы развития

Есть такой диалектический закон: переход количества в качество. Например, если навалить очень много мусора в одном месте, то возникнет качественно новый объект: стихийная свалка.

Если большую языковую модель ~~кормить мусором~~ обучать на большом объёме текстовой информации, преимущественно взятой из Интернета, то она неожиданно приобретает возможности, непосредственно в неё не заложенные, например, имитирует логический вывод.

Кажется, что увеличивая объём обучающих данных, можно неограниченно повышать качество работы модели. Однако есть проблема, а именно: ~~мусор закончился~~ последние модели уже использовали практически все доступные данные.

Ещё можно заметить, что даже предыдущие модели обучались уже на таком объёме данных, который никакой человек не способен прочесть за жизнь. Почему-то человеку хватает данных для обучения, а нейросети — не хватает.

Вероятно, что в рамках существующих архитектур не получится принципиально повысить качество ответов моделей.

Можно ли что-то принципиально улучшить в архитектуре — открытый вопрос.

Другой путь — интеграция со специализированными подсистемами (калькулятор, логический вывод).

Как ни удивительно, калькулятором ChatGPT не пользуется. Хотя нет принципиальных препятствий предусмотреть у трансформера какой-то дополнительный выход, который формирует выражения, которые далее передаются калькулятору, а ответ опять трансформеру. С логическими задачами возможен аналогичный подход.

При этом в настоящее время нет системы логического вывода, которая была бы сопоставима с человеческой логикой. Возможно, подобную получится сделать как раз путём интеграции логики в нейросетевую модель (сейчас непонятно, как к этому подступиться).

Существуют идеи использования вместо логического вывода

обобщения в виде вычислимости [10]. И логическая, и вычислительная системы — всего лишь машины с операциями.

Критерием успеха интеграции логической машины в архитектуру сети могла бы быть способность научиться играть в шахматы. Понятно, что интерес представляет не интеграция бота с шахматным движком, а именно обучение правилам шахмат с помощью универсального логического движка.

Успехи нейросетей отодвинули в тень методы компьютерной лингвистики. Возможно, что интеграция с последними тем не менее имеет перспективы.

Часто ведутся спекуляции на тему, может ли нейросеть понимать смысл.

Очень сложно дать определение, что такое смысл, особенно точное (строгое) определение.

Кажется, что сначала нужно ввести понятие модели мира. Под моделью мира мы понимаем представление человека о мире, которое позволяет ему добиваться своих целей. И собственно любой смысл — это просто какая-то часть такой модели. Проблема в том, что модель мира у каждого человека своя, поэтому когда человек пытается передать какой-то смысл словами, то, по сути, он помогает собеседнику достроить свою модель мира, чтобы она позволяла решать больше задач. В такой интерпретации смысл — это часть модели мира.

Таким образом, сначала нужно понять, насколько адекватную модель мира может построить нейросеть, и только после этого можно говорить о её понимании смысла.

Проведённые тесты показывают, что нейросеть не имеет достаточно адекватной модели мира.

Похоже, что языковые модели вполне успешно реализуют ассоциативное мышление (механизм внимания — оно и есть). Но ассоциативное мышление — это не всё, что требуется от интеллекта. Как минимум, нужно ещё умение оперировать моделями. Сценарии рассуждений, которые используются в современных моделях, в некоторой степени выполняют подобные операции, но слишком неэффективно и ненадёжно.

11.2 Историческая ретроспектива ИИ

Сейчас ИИ в широких массах ассоциируется с нейросетями. Однако до 2012 года к нейросетям «не относились всерьёз». Ещё в 70–90 годах под ИИ понимались в первую очередь экспертные системы, во вторую — машинное обучение, и только потом нейросети [15].

В области ИИ можно выделить три направления.

1. Анализ мышления как процесса обработки знаний и представлений. Логический вывод, системы рассуждений.
2. Машинное обучение. Интеллект как реакция на внешние раздражители через распознавание и прогнозирование.
3. Моделирование мозга.

Нейросети изначально создавались в рамках 3-го направления, но впоследствии стали ближе ко 2-му.

11.2.1 Краткий обзор

В кратком обзоре невозможно справедливым образом выбрать всё, что заслуживает внимания и объективно оценить вклад каких-то идей в развитие ИИ.

Поэтому автор даже не будет пытаться делать обзор объективным, и ниже приведен чисто субъективный список событий, которые в большей степени привлекли внимание автора.

Дискриминант Фишера (1936) — первый (из известных автору) метод машинного обучения.

Модель искусственного нейрона (1943). Целью ставилось изучение живых нейросетей путём их моделирования. Не ставилось целей решать этими сетями реальные задачи.

Персептрон (1957) — первая архитектура нейросетей, которая демонстрировала решение простых задач обучения и распознавания. На основе идей многослойного персептрона были разработаны полносвязные нейросети и метод их обучения через обратное распространение ошибок (1974).

Шахматные программы (1960) к ИИ отношения не имеют, но на развитии ИИ они, пожалуй, сказались, поскольку привлекали внимание и интерес к этой области.

Язык Пролог (1972) — язык программирования, поддерживающий логический вывод. Другой путь — символьные вычисления (1960), SymPy (2007). Цель — научить машину решать математические задачи. Данное умение — необходимое условие интеллекта.

Экспертные системы (1970). Дальнейшее развитие идеи использовать машину для логического вывода. Подробнее это направление рассмотрим далее.

Адаптивные методы глобальной оптимизации. Случайный поиск с адаптацией (1965). Генетические алгоритмы (1970).

Сложностная теория распознавания (1974). Кажется, что на практике эти результаты вообще не используются, однако данная теория оказала большое влияние на понимание процесса обучения. Соответственно, понимание этой теории необходимо для любого специалиста по анализу данных.

Свёрточные сети (1988). Свёрточные слои являются обобщением классических фильтров для работы с изображениями. Через 25 лет именно свёрточные сети обеспечили революционный прорыв в качестве распознавания изображений, когда точность впервые стала сопоставимой с человеческой.

Бустинг (1990). Градиентный бустинг (1999). Ансамблевые методы — самые эффективные в машинном обучении.

Открытие портала kaggle (2010). Впервые появился действительно рабочий инструмент для сравнения качества разных методов. В частности, благодаря kaggle, выяснилось, что бустинг — лучший метод (для табличных данных). До этого встречалось заблуждение, что бустинг не склонен переобучаться, поэтому его не всегда останавливали вовремя. Решение конкурсных задач позволило отработать навыки настройки гиперпараметров методов и помогло становлению машинного обучения как индустрии.

Механизм внимания в нейросетях (1992, 2016). Идея, что сигналы в нейросети можно не только складывать, но и перемножать, вероятно, напрашивалась, но работоспособную реализацию

оказалось непросто сделать. Однако когда это удалось, результат оказался революционным. Механизм внимания можно также считать альтернативой полносвязности.

Глубокое обучение (1986) — по сути «ребрендинг» нейросетей. По большому счёту это просто синоним использования нейросетей.

Однако этот термин, пожалуй, привнёс нечто новое, а именно понимание, что не всегда стоит опасаться переобучения, и нейросети с заведомо избыточным числом параметров могут давать высокую точность. Оказывается, что в задачах с нулевым байесовским уровнем ошибки переобучение может не наступать. Но именно такие задачи характерны, например, при распознавании объектов на изображении. Одна и та же фотография не может быть фотографией разных людей. В противовес этому для задач с табличными данными как раз типична ситуация, когда для объектов разных классов значения всех признаков совпадают (и байесовский уровень ошибки существенно больше нуля).

Формально почти все реально используемые архитектуры сетей можно считать глубокими, поскольку они имеют достаточно много слоёв. При этом полносвязных слоёв, как и раньше, редко используется больше трёх, в основном количество обеспечивается за счёт слоёв других типов, например свёрточных.

11.2.2 Экспертные системы

Нейросети отображают текст в некоторое числовое пространство, различные области которого можно в каком-то смысле интерпретировать как понятия. Фактически нейросеть сама формирует понятия (но есть сомнения в том, корректно ли называть это понятиями).

Экспертные системы оперируют понятиями и знаниями, которые непосредственно заложены в них человеком.

Экспертные системы — пожалуй, первый пример систем, демонстрирующих способности, похожие на интеллектуальные.

Одной из первых была система MYCIN, разработанная в нача-

ле 1970-х годов в Стэнфордском университете. Система позволяла довольно точно проводить диагностику определённых инфекционных заболеваний и назначать лечение.

В конце главы приведён код программы (примерно 1980-х годов) для простой игры, заключающейся в угадывании слов. Пример показывает, что даже тривиальная программа может обучаться и вести осмысленный диалог. Данная программа позволяет получить, хоть и отдалённое, но всё же адекватное представление о работе экспертной системы.

Работа экспертной системы основывается на базе знаний. База знаний — база данных, содержащая правила вывода и информацию о человеческом опыте и знаниях в некоторой предметной области. В самообучающихся системах база знаний также содержит информацию, являющуюся результатом решения предыдущих задач.

Принципиальное ограничение экспертных систем в том, что они оперируют правилами (продукциями), но далеко не все модели, которыми человек описывает окружающий мир, представимы в виде набора продукций.

Логика работы (реально существовавших) экспертных систем фактически сводится к вычислению значений некоторых переменных на основе значений других переменных. Это соответствует логике исчисления высказываний.

Однако люди используют не просто переменные, люди используют понятия, которые образуют иерархическую систему.

Попытки формализации подобных систем выразились в создании так называемых онтологий.

Современные базы знаний работают совместно с системами поиска и извлечения информации. Для этого требуются некоторая модель классификации понятий и определённый формат представления знаний. Иерархический способ представления в базе знаний набора понятий и их связей называется онтологией.

В своём изначальном значении онтология — раздел философии, наряду с гносеологией. То, что называют онтологией в информатике, с онтологией в философии по большому счёту не име-

ет ничего общего.

В первом приближении онтология в информатике — иерархическая система имён. Подход, основанный на онтологиях, перспективный, но практическое использование пока ограничено. Направляется интеграция этого инструмента с нейронными сетями, но непонятно, как это может выглядеть.

Одной из важных проблем при создании экспертных систем является согласование знаний различных экспертов. Очевидно, что высказывания разных экспертов могут противоречить друг другу, и не всегда эксперты могут эти противоречия устранить.

Известен так называемый парадокс Эрроу, который представляет собой строго доказанное математическое утверждение о том, что не существует процедуры согласования предпочтений, которая бы удовлетворяла сформулированному набору естественных требований. Что это за требования, мы выписывать не будем, поскольку их легко найти в литературе.

Парадокс Эрроу не имеет отношения к проблеме согласования экспертных знаний, поскольку возникает при согласовании предпочтений, но не мнений.

Для иллюстрации рассмотрим пример. Пусть избиратель 1 проголосовал на кандидата А, а избиратель 2 за кандидата В. Означает ли это, что у этих избирателей разные мнения? Не означает. Если спросить у 2, какой кандидат объективно выгоднее избирателю 1, то он вполне вероятно скажет, что А. А избиратель 1 честно скажет, что для избирателя 2 объективно выгоднее кандидат В. Таким образом, мнения у этих избирателей полностью совпадают. Разные у них только интересы, т. е. предпочтения.

Поэтому если цель — получить объективное знание, то в приведённом примере проблемы вообще не возникает, так как мнения избирателей непротиворечивы: оба согласны, что первому выгоднее вариант А, а второму В.

Неразрешимая проблема возникает только, если требуется согласовать предпочтения (т. е. результаты голосования за альтернативы).

Если же требуется согласовать мнения, т. е. разные варианты

знаний, то проблема возникает, но она вполне разрешима.

Один из универсальных способов согласования противоречивых знаний экспертов заключается в интерпретации знаний (высказываний экспертов) как случайных событий. После этого мы можем использовать стандартный принцип максимального правдоподобия, т. е. можем каждый вариант согласованного знания оценить с точки зрения правдоподобности появления имеющегося набора экспертных высказываний.

11.3 Естественный интеллект

Гипотеза — это то, что можно проверить. Предположения, которые в принципе невозможно проверить — это область философии.

Относительно принципов работы и механизмов появления естественного интеллекта сейчас практически нет достоверной информации, поэтому займёмся абсолютно безответственными спекуляциями попробуем сделать предположения.

Основная цель — понять, может ли в естественном интеллекте быть что-то, что принципиально невозможно воспроизвести искусственно, и действительно ли это что-то критически важно для полноценного интеллекта.

11.3.1 Интеллект и сознание

Очевидное и бесспорное отличие естественного интеллекта от искусственного — это наличие сознания.

Чтобы понять, может ли сознание быть необходимым для интеллекта, рассмотрим такой аспект сознания как воля.

Проблема в том, что понятие воли нельзя строго определить. Мы будем понимать волю примерно так же, как она понимается в работе [46].

Давайте примем такую почти бытовую интерпретацию воли как способность совершать действия вопреки внешнему давлению.

Или как способность систематически совершать менее вероятные действия. Пожалуй, это ключевая формулировка, которую нужно детально пояснить.

Разумеется, речь не идёт о том, что человек своей волей может заставить монету выпасть гербом 10 раз из 10 или заставить быстрые молекулы собраться в одной половине сосуда.

Человек, очевидно, не может влиять на монету. Но он может влиять на свои действия. По крайней мере, ему так кажется. Давайте предположим, что не кажется, а что он действительно своей волей влияет на протекание нервных сигналов в своём мозгу.

Что понимать под влиянием?

Мозг состоит из элементов такого масштаба, что протекание нервных импульсов подчиняется не только макроскопическим факторам, но и квантовомеханическим. А значит, описывается волновой функцией.

Влияние воли может выглядеть как изменение этой волновой функции, т. е. как изменение вероятностей. Но это плохая гипотеза. Волновая функция (будучи физическим понятием) должна полностью определяться физическими взаимодействиями, а не сознанием (иначе можно дойти до парапсихологии). Однако реализация случайности в пределах волновой функции — это та «серая зона», в которой физика не пытается строить каких-либо моделей. Иными словами, предположение о том, что воля субъекта может выбирать действия не в соответствии с их вероятностью, пока ничему не противоречит.

Если речь идёт о бросании монеты, то мы можем пренебречь возможностью крайне маловероятных событий (например, больших отклонений частоты от вероятности). Но почему бы не предположить, что воля позволяет человеку совершать поступки, которые с точки зрения объективных факторов являются маловероятными.

Предположим, что воля — это акт коллапса волновой функции. Кстати в физике коллапс волновой функции происходит в момент наблюдения (измерения). А сознание — это и есть наблюдатель, в присутствии которого волновая функция сигналов мозга

должна коллапсировать непрерывно. Это, конечно, чистая спекуляция. Но вполне стройная.

Проверить такую теорию нельзя, но и опровергнуть — тоже. Философы предложили учёным принцип фальсифицируемости как один из критериев научности. При этом сами им не пользуются (по крайней мере, ни одна из философских теорий за 2500 лет пока не опровергнута).

В нейросетях не может быть воли в описанном понимании. Хотя бы потому, что в них нет случайности, там используются псевдослучайные числа (которые детерминированы). Но даже если бы была, не любая реализация случайности — воля.

С позиции дилетанта (в психологии) можно предположить, что расстройства мышления это и есть потеря волевого контроля над мыслями. А современные языковые модели как раз ведут себя как (не всегда) тщательно маскирующийся шизофреник. Возможно, отсутствие воли (и соответственно сознания) в нейросетях можно компенсировать добавлением логического движка, который поможет устранить «галлюцинации».

11.3.2 Виды познания

Рассмотрим возможные принципиальные отличия познания человеком и нейросетью.

Будем основываться на одной из наиболее фундаментальных работ о процессе познания — [22].

Ключевой вопрос — исследование познавательной возможности разума в отрыве от знаний, получаемых эмпирическим (опытным) путём. Нейросети все знания получают в процессе обучения, т. е. «опытным» путём, поэтому важно понять, насколько важны знания, которые нельзя получить опытным путём.

Согласно представлениям, развиваемым в [22], знания бывают априорные и апостериорные, а также имманентные, трансцендентные и трансцендентальные.

Априорные знания — те, что получены до опыта, апостериорные — в результате опыта.

Имманентное — это то, что можно познать на основе личного опыта. Таким образом, всё, что человек воспринимает при помощи своих органов чувств, — имманентно.

Трансцендентное — то, что принципиально недоступно для рассудочного познания, а может быть лишь предметом веры (здесь слово «вера» не в религиозном смысле).

Трансцендентальное же находится за пределами опыта в том смысле, что предшествует ему. Оно делает опыт возможным («наш рассудок сам же и предписывает природе её законы»). Основным примером такого трансцендентального знания могут служить пространство и время.

Некоторые примеры запросов могут показаться демонстрациями определённого уровня владения нейросетью концепциями пространства и времени. Однако и в шахматах нейросеть способна сделать корректно несколько ходов. При этом играть не способна. Скорее всего, та же ситуация с «пониманием» пространственных отношений.

Можно ли априорные знания поместить в нейросеть? Очевидно, да. Для этого нужно каким-то образом заложить эти знания в её архитектуру (возможно, в виде отдельных подсистем).

Таким образом, мы пришли к той же идее, что и ранее.

11.4 Примеры кода

Ниже приведён код простой игры, демонстрирующий общение в стиле экспертных систем.

Сначала приведём класс, обеспечивающий логику игры.

```
def is_yes(s):
    return len(s)>0 and not s[0] in ['n', 'N', 'н', 'Н']

class Game():
    def __init__(self):
        self.tree = {'left': None, 'right': None,
                    'condition': 'свинья'}
        self.cur_node = self.tree
        self.state = 'START'
```

```

def get_question(self):
    if self.state == 'START':
        return 'Задумайте животное и нажмите <enter>'

    elif self.state == 'TRY':
        D = self.cur_node
        return ('Это ' if D['left'] is None else ''
                ) + D['condition'] + '?'

    elif self.state == 'MISS':
        return 'А кто это?'

    elif self.state == 'ADD':
        return 'В чём особенность этого животного?'

    elif self.state == 'END':
        return 'Играем ещё?'

    elif self.state == 'QUIT':
        return None

def process_reply(self, s):
    if self.state == 'START':
        self.state = 'TRY'

    elif self.state == 'TRY':
        D = self.cur_node
        if not D['left'] is None:
            self.cur_node = (D['right'] if is_yes(s)
                             else D['left'])
        else:
            self.state = 'END' if is_yes(s) else 'MISS'

    elif self.state == 'MISS':
        D = self.cur_node
        D['left'] = {'left': None, 'right': None,
                    'condition': D['condition']}
        D['right'] = {'left': None, 'right': None,
                     'condition': s}
        D['condition'] = ''
        self.state = 'ADD'

```

```

elif self.state == 'ADD':
    D = self.cur_node
    D['condition'] = s
    self.state = 'END'

elif self.state == 'END':
    self.cur_node = self.tree
    self.state = 'START' if is_yes(s) else 'QUIT'

```

Код обмена сообщениями с пользователем.

```
G = Game()
```

```

while s := G.get_question():
    print(s)
    G.process_reply(input())

```

Ниже для наглядности приведён пример общения с данной программой.

```

Задумайте животное. По готовности нажмите «enter»
Это свинья? Нет
А кто это? утконос
В чём особенность этого животного? У него утиный нос
Играем ещё? Да
Задумайте животное. По готовности нажмите «enter»
У него утиный нос? Да
Это утконос? Нет
А кто это? утка
В чём особенность этого животного? Умеет летать
Играем ещё? Да
Задумайте животное. По готовности нажмите «enter»
У него утиный нос? Да
Умеет летать? Нет
Это утконос? Да
Играем ещё? Нет

```

Понятно, что диалог совершенно примитивный (и его осмысленность всецело зависит от игрока), но любопытно, что такая тривиальная программа демонстрирует способность к обучению. А самое главное её достоинство — она не способна галлюцинировать.

Заключение

Несмотря на огромный объём исследований, проводимых в области машинного обучения, в настоящее время остаётся множество вопросов, на которые нет полного ответа. Причём многие вопросы касаются классических тем и методов, которые рассматривались ещё полвека назад.

Интенсивное развитие области приводит к тому, что постоянно формируются новые темы исследований. При этом «старые» зачастую почти «забываются» и остаются не до конца проработанными.

Этому также способствует общепринятая политика научных журналов, где гораздо охотнее принимают статьи на популярные темы (на которые в среднем будет больше ссылок и которые лучше повлияют на импакт-фактор издания), чем на темы, хоть и важные, но малообсуждаемые.

Открытые проблемы машинного обучения.

- ✧ Каков оптимальный метод классификации в случае нормально распределённых классов?
- ✧ Можно ли вообще ввести понятие оптимальности для метода построения решающих функций.
- ✧ Построение доверительного интервала для оценки методом кроссвалидации.
- ✧ Действительно ли есть смысл использовать bootstrap?
- ✧ Можно ли при построении оценок число решающих функций заменять функцией роста.

- ✧ Как в нейросеть внедрить принципы бустинга?
- ✧ Действительно ли встречаются неунимодальные кривые обучения (двугорбые) или это эффект от неадекватной шкалы прогресса обучения?
- ✧ В какой степени процесс подбора гиперпараметров может приводить к переобучению?
- ✧ Как автоматизировать подбор гиперпараметров?
- ✧ Действительно ли дистилляция может не только сократить размер модели, но и повысить точность?
- ✧ Как научить большие языковые модели логике?
- ✧ Связь кроссвалидации со статистическими критериями.
- ✧ Частотный прогноз по множеству категориальных переменных. Как учесть Stein's example (Stein's paradox) ?
- ✧ Способы реализации механизма внимания в нейросетях.
- ✧ Почему и за счёт чего большие языковые модели показывают настолько впечатляющие результаты и демонстрируют способности, которые в них не закладывались?

На перечисленные вопросы пока нет окончательных ответов.

При этом частичные ответы на большинство вопросов имеются, но они все получены в частных экспериментах. А в любых экспериментах по анализу данных присутствует множество факторов, и очень сложно убедительно показать, что заявленный эффект обеспечивается именно интересующими факторами (например, использованием дистилляции), а не какими-то неучтёнными.

Glossarium

Терминология области машинного обучения до сих пор не вполне устоялась. Здесь приведены варианты определений понятий в том смысле, как они использованы в пособии (номера страниц указывают на место использования). Формулировки не претендуют на роль эталона. В других источниках определения могут быть иными.

Алгоритм классификации — в данном тексте термин используется как синоним метода классификации. Но во многих источниках алгоритм понимают как синоним классификатора, т. е. решающей функции. По этой причине автор старался этого термина избегать.

Алгоритмическая сложность (колмогоровская сложность) модели — длина наиболее короткого алгоритма (программы), способного точно реализовать решающую функцию. Понятие привлекательно своей гибкостью и универсальностью, однако в машинном обучении его использование затруднительно. В машинном обучении сложность модели должна быть монотонно связана с её аппроксимирующей способностью, а для колмогоровской сложности такая связь неоднозначна.

Ансамбль решающих функций — линейная композиция, т. е. решение в виде взвешенной суммы базовых решающих функций.

Аугментация — получение новых данных путём преобразований (модификации) исходных данных. В некотором роде эта операция противоположна нормализации.

Байесовская вероятность — одна из интерпретаций понятия вероятности в философии. Вопросы интерпретации математических понятий находятся за рамками собственно математики. Истинность математических утверждений не зависит от их содержательной интерпретации.

Байесовская решающая функция — оптимальная решающая функция (имеющая минимально возможный риск). Понятие может быть введено без использования формулы Байеса.

Байесовская теория распознавания — устаревший термин. В настоящее время может считаться синонимом вероятностной постановки задачи машинного обучения. Хотя в ряде источников термин используется как раз, чтобы подчеркнуть использование именно байесовского подхода, а не просто вероятностной постановки задачи.

Байесовский подход — подход, основанный на интерпретации неизвестных значений как случайных. Использует вероятность, в том числе, как меру незнания. Зачастую применение байесовского подхода эквивалентно некоторому способу регуляризации.

В некоторых источниках байесовский подход (байесовская статистика) чуть ли не противопоставляется «остальной» статистике, но по сути это её часть, т. е. один из полезных приёмов.

Байесовское оценивание параметров, или байесовская статистика, — метод математической статистики, основанный на задании априорного распределения на неизвестных параметрах.

Бахадура ряд — разложение произвольного распределения в пространстве бинарных переменных на сумму компонент, последовательно отражающих всё более высокий уровень зависимости переменных.

Блендинг (blending) — один из методов построения ансамбля на основе различных базовых методов. В большинстве источников термин блендинг используют как синоним стэкинга. Однако здесь под блендингом понимается простое или взвешенное голосование,

поскольку именно оно ассоциируется со «смешиванием». Такая интерпретация также встречается в литературе.

Бустинг, boosting — ансамблевый метод, базовые решающие функции в котором строятся последовательно и уточняют текущее решение («исправляют» ошибки построенных ранее решающих функций).

Бутстреп-выборка, bootstrap-выборка — выборка, сформированная из исходной выборки размера N путём N -кратного случайного выбора объектов (строк) с повторением. Это эквивалентно тому, что объектам исходной выборки назначаются случайные веса из распределения Пуассона с параметром 1. Строго говоря, распределение Пуассона получается только в пределе $N \rightarrow \infty$, но разница практически пренебрежима.

Валидационная выборка — часть выборки, не используемая для обучения, но использованная для настройки гиперпараметров (или для выбора метода).

Вероятностная постановка задачи машинного обучения — предположение, что характеристики объектов являются случайными величинами с некоторым (неизвестным) совместным распределением.

Является обоснованием для применения модели, обученной на некоторых данных, к новым объектам. Также позволяет строить математически строгие оценки риска.

Вероятность — мера, удовлетворяющая аксиомам Колмогорова. Содержательный смысл вероятности вызывает споры и объяснить его так же сложно, как объяснить смысл пространства или времени. И понятно это объяснение будет только тем, кто уже и так понимает вероятность. Выработать понимание можно только решая практические задачи.

Для изучения машинного обучения предварительного глубокого понимания вероятностей не требуется. Понимание будет приходиться в процессе применения методов.

Выборка — последовательность независимых одинаково распределённых многомерных случайных величин. Выборка соответствует подмножеству объектов генеральной совокупности. Неформально выборка — это таблица данных, значения в которой мы интерпретируем как случайные. Строки соответствуют объектам, столбцы — их характеристикам (признакам).

Генеральная совокупность — множество объектов, подлежащих анализу (изучению). Включает не только (и не столько) реально существующие объекты, но и воображаемые объекты, существование которых мы полагаем возможным.

Генеративные и дискриминативные (предиктивные) модели — мы их называем полными и неполными вероятностными моделями соответственно.

Генетический алгоритм — метод многоэкстремальной оптимизации, реализующий идею случайного поиска с адаптацией на основе моделей генетических механизмов.

Гиперпараметр — параметр метода, а не решающей функции. Если в задаче пробуем применять несколько методов, то выбор метода тоже считаем подбором гиперпараметра (значения этого гиперпараметра — имена методов).

Глубокая нейронная сеть (англ. Deep neural network) — это искусственная нейронная сеть с несколькими слоями между входным и выходным слоями.

Глубокое обучение (с. 112) — методы машинного обучения, формирующие многоуровневое представление данных. Часто под глубоким обучением подразумевают просто использование глубоких нейронных сетей.

Очевидно, что глубокое обучение является намного более серьёзным занятием по сравнению с неглубоким обучением, поэтому его упоминание обязательно в резюме для всех специалистов по машинному обучению.

Дерево решений — кусочно-постоянная решающая функция с областями постоянства, формирующими последовательное (чаще всего дихотомическое) разбиение признакового пространства, а также представление этой функции в виде дерева (графа) с предикатами в узлах.

Детерминистский подход в машинном обучении — подход, не интерпретирующий данные как случайные. Например, дискриминант Фишера не требует вероятностной постановки для своего вывода. Но чтобы оценить ожидаемую точность прогноза на новых данных, проще всего применить методы математической статистики. Однако есть и другие подходы, но в данном курсе они не рассматриваются.

Закономерность — некоторая особенность в данных, которую можно использовать в качестве критерия, позволяющего отвергнуть «нулевую» гипотезу.

Интеграл Лебега — интеграл по мере. Разбивает не область определения, а область значений. Более естественен для вероятности. Не сложнее интеграла Римана, если знать, что такое мера (а это знать всё равно придётся). Позволяет интегрировать функции на нечисловых пространствах.

Интеграл Стильбеса — удобная форма интеграла. Выглядит как простое внесение функции под дифференциал. Но применим и в случаях, когда под дифференциалом негладкая функция. Даёт самую удобную форму для записи математического ожидания.

Ирония (от англ. trolling) — приём используемый авторами, когда отсутствует рациональная аргументация. В научных публикациях использование почти неприемлемо. В учебных материалах допускается в небольшом объёме и с осторожностью (поскольку в текстовом формате невозможна вставка закадрового смеха, и читатель может перепутать иронию с серьёзным утверждением).

Искусственный интеллект (ИИ) — область знаний, посвящённая созданию методов автоматизации интеллектуальной де-

тельности человека. Грубо говоря, цель — создать программы, позволяющие (хотя бы частично) заменить человеческий интеллект при решении каких-либо задач. Область ИИ включает машинное обучение, а также направления, связанные с представлением и использованием знаний (экспертные системы, онтологии и т. п.).

Искусственный разум — концепция из области философии, а также научной фантастики. Вообще, разум есть интеллект плюс сознание и воля. При этом воля подразумевает сознание, поэтому отдельно его можно не оговаривать. Рассуждения о возможности сознания у искусственных объектов обычно носят спекулятивный характер.

Комбинаторная размерность множества (класса) решающих функций — максимальный размер выборки (множества точек), которую можно классифицировать всеми способами функциями данного класса.

Композиция решающих функций — решение в виде некоторой функции от базовых решающих функций. Пример — стэкинг.

Кривая обучения — зависимость риска от какого-либо параметра сложности модели (например, от числа деревьев ансамбля или числа эпох).

Кроссэнтропия, cross-entropy loss — то же самое, что \log loss.

Логарифмические потери, log loss — взятый со знаком «минус» логарифм правдоподобия выборки для заданного вероятностного прогноза. Термин \log loss возник как сокращение от «logistic regression loss», но кажется неоправданным связывать термин с логистической регрессией, поскольку логарифмическое правдоподобие использовалось задолго до её появления. Метрика \log loss «не привязана» к логистической регрессии и может использоваться с любыми методами классификации. Она является практически единственной метрикой, пригодной для оценки качества вероятностного прогноза.

Машинное обучение — научное направление, посвящённое разработке методов построения решающих функций.

Мера — неотрицательная функция множества, удовлетворяющая ряду аксиом. Примеры мер: количество, длина, площадь, объём, масса, заряд, вероятность.

Метод — формализованный способ достижения заданной цели. Это достаточно широкое понятие из области методологии науки. Однако термин «метод построения решающих функций» мы используем в очень узком строго математическом смысле.

Метод наименьших квадратов — метод, основанный на минимизации квадратичной функции потерь по некоторому классу решающих функций.

В настоящем курсе данный термин не используется. Дело в том, что словосочетание «метод наименьших квадратов» подразумевает, что минимизация среднего квадрата отклонения является средством для достижения какой-то цели (например, для нахождения «истинной» зависимости). Но в данной работе используется постановка, где нет «истинной» зависимости, а минимизация ожидаемых потерь (в частности, среднего квадрата отклонения) является целью, а не средством. В такой постановке более уместно говорить о «задаче наименьших квадратов».

Метод построения решающих функций — отображение множества выборок во множество решающих функций. Термин не вполне удачный, поскольку слово «метод» здесь используется в узком смысле, отличном от привычного. С другой стороны, термин «метод класса» в программировании никого не смущает.

Метрика, или расстояние, — неотрицательная функция, заданная на парах точек и удовлетворяющая ряду аксиом (тождества, неотрицательность, симметричность, неравенство треугольника).

Метрика качества — функция (формально функционал), аргументами которой являются решающая функция и вероятностное распределение. Характеризуют качество (точность) ре-

шающей функции. Термин может вызвать путаницу, поскольку метрика качества в большинстве случаев не является метрикой (в математическом смысле).

Механизм внимания — процедура формирования новых векторных представлений (эмбедингов) на основе векторов нижнего уровня. Представляет собой искусственный аналог ассоциативного мышления. Позволяет формировать абстрактные понятия, но не реализует полноценное абстрактное мышление (поскольку умеет только формировать понятия, но не оперировать ими).

Многослойный персептрон — многослойная сеть полносвязной архитектуры.

Модель — многозначное понятие. Одно из значений — синоним решающей функции. Также использовалось в значениях: класс решающих функций и класс распределений.

Недообучение — выбор значения параметра сложности модели ниже оптимального значения сложности.

Нейронная сеть — суперпозиция линейных и нелинейных функций. Это не самое общее определение, так как не любая конфигурация связей нейронов представляется суперпозицией функций. Контрпример — триггер. Но все существующие библиотеки для нейронных сетей позволяют реализовывать только суперпозиции (триггер в них не реализуем).

Нормализация данных — преобразование исходных данных, снижающее их вариативность. Пример: приведение числовых значений к удобному диапазону.

«Нулевая» гипотеза — вероятностная модель (распределение данных), при которой целевая переменная не зависит от X . При нулевой гипотезе риск для любой решающей функции не ниже, чем риск наилучшей константной функции (которая не зависит от X).

Обобщающая способность — точность решающей функции на новых объектах. В вероятностной постановке это просто значение метрики качества (на распределении) или величина, противоположная риску. Термин получается несамостоятельным, поэтому мы его практически не использовали.

Однако в невероятностных постановках термин крайне важен.

Следует различать важность термина и важность понятия, которое он обозначает. Понятие обобщающей способности является фундаментальным при любых подходах. Но при вероятностной постановке для этого понятия существуют другие термины, поэтому сочетание «обобщающая способность» используется реже.

Обучение с подкреплением — обобщение задачи планирования эксперимента на случай не только регрессионных моделей.

Отступ — величина, модуль которой отражает уверенность метода в прогнозе, а знак соответствует правильности прогноза. Для бинарной классификации отступ зачастую может быть выражен как произведение «сырого» прогноза на метку класса (1 или -1).

Переобучение — выбор значения параметра сложности модели, превышающего оптимальное значение сложности.

Перестановочный тест — приём, основанный на перемешивании значений целевой переменной в таблице данных. После такой перестановки целевая переменная становится независимой, а риск — максимальным.

Правдоподобие — в данном контексте это не математическое понятие, а неформальное, определим его как вероятность события, которое произошло, либо через плотность вероятности. При этом относим правдоподобие мы не к событию, а к некоторой гипотезе. Например, некто сообщил нам, что 20 раз подбросил монету, и во всех испытаниях выпал герб. В случае правильной монеты вероятность такого события равна 2^{-20} , т. е. очень мала. Но если событие произошло, значит, условия эксперимента, скорее всего, были иными, например герб на монете был с обеих сторон.

Как отдельное понятие «правдоподобие» обычно не вводится, а используется например отношение правдоподобия.

Признак (feature), характеристика объекта — используется как синоним переменной, характеризующей объект.

PR-кривая — строится аналогично ROC-кривой, только по осям откладываются величины precision и recall.

Размерность Вапника–Червоненкиса — комбинаторная размерность.

Регрессия (функция регрессии) — условное математическое ожидание некоторой случайной величины при фиксированных значениях других случайных величин.

Регуляризация — приём, повышающий устойчивость модели. Вводит параметры, ограничивающие сложность модели, и параметры, вынуждающие делать «более осторожный» прогноз (например, никогда не оценивать вероятность нулём и единицей).

Риск — математическое ожидание функции потерь.

ROC-кривая — зависимость мощности критерия от его уровня (значимости). Чтобы применить определение к задаче бинарной классификации, нужно в качестве основной гипотезы взять принадлежность объекта «негативному» классу (мы его обозначали 0 или -1), а в качестве простой альтернативы рассмотреть принадлежность объекта «позитивному» классу (мы его всегда обозначали 1). В этом случае по оси абсцисс у нас окажется вероятность ошибочной классификации для -1 -го класса, а по оси ординат — вероятность правильной классификации при условии 1-го класса.

При построении кривой по выборке вместо вероятностей используются соответствующие частоты.

Разные точки кривой соответствуют разным порогам отнесения объекта к первому классу на основе вероятностного прогноза.

ROC-AUC — площадь под ROC-кривой. Пример неаддитивной метрики качества. Часто используется в соревнованиях и

практических задачах.

Сигмоид, логистическая функция — функция, задаваемая формулой $\sigma(z) = \frac{1}{1+e^{-z}}$. Используется в методе логистической регрессии. Возникает также в наивном байесовском классификаторе. Часто используется для перевода «сырого» прогноза в вероятность. На случай многих классов обобщается функцией softmax.

Сильный искусственный интеллект — компьютерная программа, позволяющая заменить человеческий интеллект при решении любых задач, с которыми может справиться естественный (человеческий) интеллект. Возможность создания сильного ИИ остаётся под вопросом.

Сложность модели — не до конца формализованное понятие. Обычно интуитивно понятно, какие гиперпараметры характеризуют сложность. Есть чёткий критерий, может ли гиперпараметр считаться сложностью, а именно: с увеличением сложности эмпирический риск не может расти. С ростом сложности модели её аппроксимирующая способность растёт.

Случайное событие — множество, для которого определена (это не значит, что известна) вероятностная мера.

Случайный лес, random forest — ансамблевый метод на основе решающих деревьев, базовые решающие функции в котором строятся независимо. Разнообразие деревьев достигается за счёт использования случайных подмножеств объектов выборки и признаков. Для усиления разнообразия используются bootstrap-выборки.

Случайный поиск с адаптацией — метод многоэкстремальной оптимизации, основанный на последовательной случайной расстановке точек в соответствии с адаптивно изменяемым распределением.

Статистический анализ данных — математическая статистика. Сложно провести границу с машинным обучением.

Статистическое моделирование — исследование, основанное на генерации выборок из некоторого распределения. Возможен вариант, когда выборки берутся из большого массива реальных данных.

Стратегия природы — вероятностная модель, т. е. совместное распределение на всех признаках, включая целевой.

Стэкинг (stacking) — подход к построению ансамбля, при котором выход одной решающей функции используется как вход другой решающей функции.

Субъективные вероятности — некоторый подход к интерпретации вероятности. Понятие относится скорее к философии, чем к математике. В данном пособии не используется.

«Сырой» прогноз (raw prediction) — прогноз в задаче классификации, выраженный в некоторой условной (невероятностной) шкале, так чтобы большие значения прогноза соответствовали большей уверенности.

Такой прогноз выдаёт метод `decision_function` классов библиотеки `scikit-learn`.

Теорема о бесплатном завтраке — математическое утверждение, содержательная интерпретация которого сводится к тому, что если не делать никаких предположений о классе оптимизируемых функций, то все алгоритмы многоэкстремальной оптимизации имеют равную эффективность. Последнее означает, что самый абсурдный алгоритм имеет те же шансы найти экстремум, что и самый «разумный» алгоритм оптимизации.

Теория статистических решений — термин, в некоторых случаях использовавшийся как синоним машинного обучения. Фактически он более точный (в случае вероятностной постановки). Термин подчёркивает, что прогноз нужен именно для принятия решений. Если не принимать решения, то и прогноз не имеет смысла, и точность прогноза не важна. Риск возникает как следствие принятых (на основе прогноза) решений.

Однако в настоящее время термин чаще употребляется в кон-

тексте принятия решений по уже имеющемуся прогнозу.

Тестовая выборка — выборка, никак не использованная при построении решающей функции (ни для обучения, ни для настройки гиперпараметров, ни для выбора метода).

Трансформер — нейросеть определённой архитектуры, которая включает кодировщик и декодировщик. Подразумевается, что при этом используется механизм (само)внимания.

Триггер — схема из логических элементов, обеспечивающая запоминание бита информации. Может быть реализована нейросетью, но не представима суперпозицией функций, реализуемых нейронами.

Утконос (лат. *Ornithorhynchus anatinus*) (с. 378) — водоплавающее млекопитающее, обитающее в Австралии. Это единственный современный представитель семейства утконосовых; вместе с ехидновыми образует отряд однопроходных (*Monotremata*) млекопитающих, по ряду признаков напоминающих рептилий и нематемальных цинодонт (информация из Википедии).

Демонстрирует, что в определённых нишах устаревшие решения и технологии могут быть жизнеспособны.

Функция потерь — функция, отражающая несоответствие между прогнозом и фактическим значением целевой переменной. Следует различать целевую функцию потерь и функцию потерь для обучения. В идеале целевая функция потерь должна соответствовать реальному ущербу от принятия решения на основе ошибочного прогноза. Её математическое ожидание (вернее, противоположная по знаку величина) является метрикой качества. Функция потерь для обучения выбирается эвристически и может не совпадать с целевой.

Шахматная программа — узкоспециализированный алгоритм оптимизации. Зачастую такие программы относят к искусственному интеллекту, хотя для этого нет оснований. Наиболее известные шахматные программы представляют собой жёсткий переборный алгоритм без обучения. Хотя они и используют мето-

ды, которые формально можно отнести к машинному обучению (как и любые методы статистического анализа данных).

Существуют шахматные программы на основе машинного обучения, но они играют слабо. Заметим, что, в отличие от шахмат, самые сильные программы для игры в го основаны как раз на методах машинного обучения.

Эстемменозух (лат. *Estemmenosuchus*) — вымерший род примитивных растительноядных терапсид семейства эстемменозухов (*Estemmenosuchidae*).

Обнаружены в составе так называемой очёрской фауны у деревни Ежово в Очёрском районе Пермской области (возраст около 267 млн лет). Полные скелеты *E. uralensis* и *E. mirabilis* экспонируются в Палеонтологическом музее им. Ю. А. Орлова и Музее пермских древностей (информация из Википедии).

Восстановить точный внешний облик эстемменозухов не представляется возможным, на стр. 377 приведён один из вариантов его реконструкции.

Конкуренция синапсид и диапсид служит примером драматического противостояния различных подходов и решений, а также иллюстрацией того, что перспективные идеи зачастую не сразу показывают эффективность.

Литература

- [1] Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: классификация и снижение размерности. М.: Финансы и статистика, 1989. 607 с.
- [2] Банахевич К., Массарон Л. Книга Kaggle. Машинное обучение и анализ данных. 2024. 448 с.
- [3] *Belkin M., Hsu D., Ma S., Mandal S.* Reconciling modern machine-learning practice and the classical bias-variance trade-off // *Proceedings of the National Academy of Sciences of the United States of America.* 2019. Vol. 116, No 32. P. 15849–15854. DOI: <https://doi.org/10.1073/pnas.1903070116>
- [4] *Бонгард М. М.* Проблема узнавания. М.: Наука, 1967. 320 с.
- [5] *Борисов И. С.* Лекции по теории вероятностей: учеб. пособие / НГУ. Новосибирск, 2010. 96 с.
- [6] *Боровков А. А.* Теория вероятностей: учеб. пособие для вузов. М.: Наука, 1986. 432 с.
- [7] *Braga-Neto U., Dougherty E. R.* Exact performance of error estimators for discrete classifiers // *Pattern Recognition.* 2005. Vol. 38, No. 11. P. 1799–1814.
- [8] *Вапник В. Н., Червоненкис А. Я.* Теория распознавания образов. М.: Наука, 1974. 415 с.

- [9] *Верещагин Н. К., Успенский В. А., Шень А.* Колмогоровская сложность и алгоритмическая случайность. М.: МЦНМО, 2013. 576 с. ISBN 978-5-4439-0212-8
- [10] *Витяев Е. Е., Колонин А. Г., Курпатов А. В., Молчанов А. А.* Формализация теории программирования принципов работы мозга с информацией // Доклады Российской академии наук. Математика, информатика, процессы управления. 2022. № 1. С. 111–127.
- [11] *Воронцов К. В.* Машинное обучение (курс лекций). URL: <http://www.machinelearning.ru/>
- [12] *Vorontsov K. V.* Combinatorial probability and the tightness of generalization bounds // Pattern Recognition and Image Analysis. 2008. Vol. 18, No 2. P. 243–259.
- [13] *Воронцов К. В., Ивахненко А. А., Инякин А. С., Лисица А. В., Минаев П. Ю.* «Полигон» — распределённая система для эмпирического анализа задач и алгоритмов классификации // Математические методы распознавания образов–14: материалы Всерос. конф. М.: МАКС Пресс, 2009. С. 503–506.
- [14] *Воронцов К. В.* Аддитивная регуляризация тематических моделей коллекций текстовых документов. // Доклады Академии наук. 2014. Т. 455, № 3. С. 268–271.
- [15] *Горбань А. Н.* Обучение нейронных сетей. СССР; США: Параграф, 1990. 160 с. URL: <http://lib.sibnet.ru/book/11961>
- [16] *Domingos P.* A Unified Bias-Variance Decomposition and its Applications. // Proceedings of the Seventeenth International Conference on Machine Learning. Stanford, CA: Morgan Kaufmann, 2000. P. 231–238.
- [17] *Донской В. И.* Колмогоровская сложность классов общеркурсивных функций с ограниченной ёмкостью // Таврический вестник информатики и математики. 2005. № 1. С. 25–34.

- [18] *Дьяконов А. Г.* Анализ данных, обучение по прецедентам, логические игры, системы weka, rapidminer и matlab. URL: <https://dyakonov.org/>
- [19] *Zhiyanov A., Shklyayev A., Galatenko A., Galatenko V., Tonevitsky A.* Statistical Verification of Linear Classifiers, 2025. URL: <https://arxiv.org/abs/2501.14430>
- [20] *Журавлёв Ю. И., Рязанов В. В., Сенько О. В.* Распознавание. Математические методы. Программная система. Практические применения. М.: Фазис, 2006. 147 с. ISBN 5-7036-0108-8
- [21] *Загоруйко Н. Г.* Прикладные методы анализа данных и знаний. Новосибирск: ИМ СО РАН, 1999. 270 с.
- [22] *Кант И.* Критика чистого разума (Kritik der reinen Vernunft). Рига, 1781. (Кант И. Критика чистого разума / пер. с нем. Н. О. Лосского. М.: Академический проект, 2020. 567 с.)
- [23] *Крамер Г.* Математические методы статистики. М.: Мир, 1975. 649 с.
- [24] *Langford J.* Quantitatively tight sample complexity bounds. Carnegie Mellon Thesis. 2002. 130 p. URL: <http://citeseer.ist.psu.edu/langford02quantitatively.html>
- [25] *Лбов Г. С., Бериков В. Б.* Устойчивость решающих функций в задачах распознавания образов и анализа разнотипной информации. Новосибирск: Изд-во Ин-та математики, 2005. 217 с.
- [26] *Лбов Г. С., Старцева Н. Г.* Сравнение алгоритмов распознавания с помощью программной системы «Полигон» // Анализ данных и знаний в экспертных системах. Вып. 134: Вычислительные системы. Новосибирск, 1990. С. 56–66.
- [27] *Лбов Г. С., Старцева Н. Г.* Сложность распределений в задачах классификации // Доклады РАН. 1994. Т. 338, № 5. С. 592–594.

- [28] *Лбов Г. С.* Выбор эффективной системы зависимых признаков // Вычислительные системы. Новосибирск, 1965. Вып. 19. С. 21–34.
- [29] *Lbov G. S., Nedelko V. M.* The Constructing of Decision Rule, Using the Probabilistic Statements of an Experts. (Построение решающей функции распознавания на основе вероятностных высказываний экспертов) // Pattern Recognition and Image Analysis. 1995. Vol. 5, No 2. P. 165–171.
- [30] *Мэрфи К. П.* Вероятностное машинное обучение: введение / пер. с англ. А. А. Слинкина. М.: ДМК Пресс, 2022. 940 с.
- [31] *Nedelko V. M.* Estimating a Quality of Decision Function by Empirical Risk // LNAI 2734. Machine Learning and Data Mining in Pattern Recognition. Third Intern. Conference, MLDM 2003. Proceedings. Leipzig: Springer-Verlag, 2003. P. 182–187.
- [32] *Неделько В. М.* Об эффективности эмпирических функционалов качества решающей функции // Всеросс. конф. ММРО-13. М.: МАКС Пресс, 2007. С. 47–49.
- [33] *Неделько В. М.* Об интервальном оценивании риска для решающей функции // Таврический вестник информатики и математики. Симферополь, 2008. № 2. С. 97–103.
- [34] *Неделько В. М.* Основы теории вероятностей: учеб. пособие / НГТУ. Новосибирск, 2011. 116 с.
- [35] *Nedel'ko V. M.* Statistical Fitting Criterion on the Basis of Cross-Validation Estimation // Pattern Recognition and Image Analysis. 2018. Vol. 28. P. 510–515. DOI: <https://doi.org/10.1134/S1054661818030148>
- [36] *Nedel'ko V. M.* On the Properties of Bias-Variance Decomposition for kNN Regression // Известия Иркутского государственного университета. Серия: Математика. 2023. Т. 43. С. 110–121. DOI: <https://doi.org/10.26516/1997-7670.2023.43.110>

- [37] *Николенко С.* Машинное обучение: основы. СПб.: Питер, 2025. 608 с.
- [38] *Николенко С. И., Кадурын А. А., Архангельская Е. О.* Глубокое обучение: погружение в мир нейронных сетей. М.: Питер, 2024. 480 с. ISBN 978-5-4461-1537-2
- [39] *Рудаков К. В.* О некоторых универсальных ограничениях для алгоритмов классификации // Журнал вычислительной математики и математической физики. 1986. Т. 26, № 11. С. 1719–1730.
- [40] *Тарасенко Ф. П.* Непараметрическая статистика. Томск: Томск. ун-т, 1976. 292 с.
- [41] *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning. Springer, 2014. 739 p.
- [42] *Heskes T.* Bias/Variance Decompositions for Likelihood-Based Estimators // Neural Computation. 1998. Vol. 10, No 6. P. 1425–1433.
- [43] *Чернова Н. И.* Математическая статистика: учеб. пособие / СибГУТИ. Новосибирск, 2009. 90 с.
- [44] *Чернова Н. И.* Теория вероятностей: учеб. пособие / НГУ. Новосибирск, 2014. 160 с.
- [45] *Schapire R. E., Freund Y., Lee W. S., Bartlett P.* Boosting the margin: a new explanation for the effectiveness of voting methods // Annals of Statistics. 1998. Vol. 26, No 5. P. 1651–1686.
- [46] *Шопенгауэр А.* Мир как воля и представление. Лейпциг, 1818; 1844.

Учебное издание

Неделько Виктор Михайлович

**МАШИННОЕ ОБУЧЕНИЕ
В ВЕРОЯТНОСТНОЙ ПОСТАНОВКЕ**

Учебник

Редактор *Т. И. Сухова*

Оригинал-макет *В. М. Неделько*

Иллюстрации на обложке *А. В. Бровер,*
София Блаумова

Подписано в печать 20.05.2026 г.

Формат 60 × 84 1/16. Уч.-изд. л. 27. Усл. печ. л. 25,11.

Тираж 60 экз. Заказ № 58.

Издательско-полиграфический центр НГУ
630090, Новосибирск, ул. Пирогова, 2.

ISBN 978-5-4437-1899-6



9 785443 718996

