



Кристофер М. Бишоп
Хью Бишоп

Глубокое обучение

Принципы
и концепции

Кристофер М. Бишоп, Хью Бишоп

Глубокое обучение: принципы и концепции

Christopher M. Bishop, Hugh Bishop

Deep Learning

Foundations and Concepts



Кристофер М. Бишоп, Хью Бишоп

Глубокое обучение: принципы и концепции



Москва, 2025

УДК 004.85
ББК 16.6
Б67

Бишоп К. М., Бишоп Х.
Б67 Глубокое обучение: принципы и концепции / пер. с англ. В. И. Бахура. – М.: ДМК Пресс, 2025. – 740 с.: ил.

ISBN 978-5-93700-281-5

Эта книга предлагает исчерпывающее описание фундаментальных идей, лежащих в основе глубокого обучения. Она разбита на небольшие главы с последовательным изложением материала. Особое внимание уделяется практической ценности изучаемых методов в реальном мире. Сложные концепции рассмотрены в нескольких ракурсах, включая текстовые описания, диаграммы, математические формулы и программные псевдокоды.

Издание адресовано как новичкам в машинном обучении, так и опытным специалистам в этой области.

УДК 004.85
ББК 16.6

First published in English under the title.

An Introduction to Statistical Learning; with Deep Learning: Foundations and Concepts
by Christopher M. Bishop and Hugh Bishop.

This edition has been translated and published under licence from Springer Nature Switzerland AG.

Springer Nature Switzerland AG takes no responsibility and shall not be made liable for the accuracy of the translation.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-3-031-45467-7 (англ.)

ISBN 978-5-93700-281-5 (рус.)

© Springer Nature Switzerland AG
2024
© Перевод, оформление, издание,
ДМК Пресс, 2025

Содержание

От издательства	15
Предисловие	16
Глава 1. Революция глубокого обучения	22
1.1 Влияние глубокого обучения.....	23
1.1.1 Медицинская диагностика.....	23
1.1.2 Структура белка	24
1.1.3 Синтез изображений.....	25
1.1.4 Большие языковые модели	26
1.2 Учебный пример.....	28
1.2.1 Синтетические данные.....	28
1.2.2 Линейные модели.....	30
1.2.3 Функция ошибки	30
1.2.4 Сложность модели	31
1.2.5 Регуляризация.....	35
1.2.6 Выбор модели.....	37
1.3 Краткая история машинного обучения	39
1.3.1 Однослойные сети	40
1.3.2 Обратное распространение	42
1.3.3 Глубокие сети	44
Глава 2. Вероятности	47
2.1 Правила вероятности.....	49
2.1.1 Пример медицинского обследования	49
2.1.2 Правила суммы и произведения.....	50
2.1.3 Теорема Байеса	53
2.1.4 Повторное медицинское обследование	53
2.1.5 Априорные и апостериорные вероятности.....	56
2.1.6 Независимые переменные.....	56
2.2 Плотность распределения вероятностей.....	56
2.2.1 Примеры распределений	58
2.2.2 Ожидания и ковариации	60
2.3 Гауссово распределение	61
2.3.1 Среднее значение и дисперсия	63
2.3.2 Функция правдоподобия.....	63
2.3.3 Ошибка максимального правдоподобия	65

2.3.4	Линейная регрессия	67
2.4	Преобразование плотностей	69
2.4.1	Многомерное распределение	72
2.5	Теория информации	73
2.5.1	Энтропия.....	73
2.5.2	Физическая перспектива.....	75
2.5.3	Дифференциальная энтропия	77
2.5.4	Максимальная энтропия	78
2.5.5	Дивергенция Кульбака–Лейблера	79
2.5.6	Условная энтропия	82
2.5.7	Взаимная информация	82
2.6	Байесовские вероятности	83
2.6.1	Параметры модели.....	84
2.6.2	Регуляризация.....	85
2.6.3	Байесовское машинное обучение	86
	Упражнения	87
	Глава 3. Стандартные распределения	94
3.1	Дискретные переменные	95
3.1.1	Распределение Бернулли.....	95
3.1.2	Биноминальное распределение	96
3.1.3	Полиноминальное распределение	97
3.2	Многомерное гауссово распределение	99
3.2.1	Геометрия гауссова распределения	101
3.2.2	Моменты.....	104
3.2.3	Ограничения	105
3.2.4	Условное распределение	107
3.2.5	Маргинальное распределение	110
3.2.6	Теорема Байеса	113
3.2.7	Максимальное правдоподобие	115
3.2.8	Последовательная оценка	117
3.2.9	Гауссовые смеси.....	117
3.3	Периодические переменные	121
3.3.1	Распределение фон Мизеса	121
3.4	Семейство экспоненциальных распределений	127
3.4.1	Достаточная статистика	130
3.5	Непараметрические методы	131
3.5.1	Гистограммы	132
3.5.2	Ядерная оценка плотности	134
3.5.3	Методика ближайших соседей	137
	Упражнения	140
	Глава 4. Однослойные сети: регрессия	147
4.1	Линейная регрессия	147
4.1.1	Базисные функции	148
4.1.2	Функция правдоподобия.....	150

4.1.3	Максимальное правдоподобие	151
4.1.4	Геометрия наименьших квадратов	153
4.1.5	Последовательное обучение	154
4.1.6	Регуляризованный метод наименьших квадратов	154
4.1.7	Множественные выходы	155
4.2	Теория принятия решений	157
4.3	Обратное отношение между смещением и дисперсией	161
	Упражнения	166

Глава 5. Однослойные сети: классификация 170

5.1	Дискриминантные функции	171
5.1.1	Два класса	171
5.1.2	Множественные классы	173
5.1.3	Кодирование «1 из K»	175
5.1.4	Наименьшие квадраты для классификации	175
5.2	Теория принятия решений	178
5.2.1	Коэффициент ошибок классификации	179
5.2.2	Ожидаемые потери	182
5.2.3	Опция отказа	183
5.2.4	Вывод и принятие решения	184
5.2.5	Точность классификатора	188
5.2.6	ROC-кривая	190
5.3	Генеративные классификаторы	193
5.3.1	Непрерывные входные данные	195
5.3.2	Решение методом максимального правдоподобия	197
5.3.3	Дискретные параметры	199
5.3.4	Экспоненциальное семейство	200
5.4	Дискриминационные классификаторы	201
5.4.1	Функции активации	201
5.4.2	Фиксированные базисные функции	202
5.4.3	Логистическая регрессия	203
5.4.4	Логистическая регрессия для нескольких классов	205
5.4.5	Пробит-регрессия	207
5.4.6	Канонические функции связей	209
	Упражнения	211

Глава 6. Глубокие нейронные сети 215

6.1	Ограничения фиксированных базисных функций	215
6.1.1	Проклятие размерности	216
6.1.2	Пространства большой размерности	219
6.1.3	Многообразие данных	221
6.1.4	Базисные функции на основе данных	223
6.2	Многослойные сети	224
6.2.1	Матрицы параметров	226
6.2.2	Универсальная аппроксимация	227
6.2.3	Функции активации скрытых элементов	228

6.2.4	Симметрии весового пространства	231
6.3	Глубокие сети	232
6.3.1	Иерархические представления	233
6.3.2	Распределенные представления	234
6.3.3	Обучение представлений	234
6.3.4	Трансферное обучение	236
6.3.5	Контрастивное обучение.....	238
6.3.6	Основные сетевые архитектуры	241
6.3.7	Тензоры.....	242
6.4	Функции ошибок	242
6.4.1	Регрессия	242
6.4.2	Бинарная классификация.....	244
6.4.3	Многоклассовая классификация	245
6.5	Сети смешанной плотности.....	246
6.5.1	Пример кинематики робота	247
6.5.2	Распределение условного смешивания	248
6.5.3	Градиентная оптимизация	251
6.5.4	Прогнозируемое распределение	252
	Упражнения	254
	Глава 7. Градиентный спуск	259
7.1	Поверхности ошибок	260
7.1.1	Локальная квадратичная аппроксимация.....	261
7.2	Оптимизация методом градиентного спуска	264
7.2.1	Использование градиентной информации.....	264
7.2.2	Пакетный градиентный спуск	265
7.2.3	Стохастический градиентный спуск	265
7.2.4	Мини-батчи.....	266
7.2.5	Инициализация параметров	268
7.3	Сходимость.....	269
7.3.1	Импульс.....	271
7.3.2	График скорости обучения.....	274
7.3.3	RMSProp и Adam	274
7.4	Нормализация	277
7.4.1	Нормализация данных	277
7.4.2	Пакетная нормализация.....	278
7.4.3	Нормализация слоев	281
	Упражнения	282
	Глава 8. Обратное распространение	286
8.1	Оценка градиентов	287
8.1.1	Однослойные сети	287
8.1.2	Общие сети с прямой передачей	288
8.1.3	Простой пример.....	291
8.1.4	Численное дифференцирование	292
8.1.5	Матрица Якоби	294

8.1.6	Матрица Гессе	296
8.2	Автоматическое дифференцирование	299
8.2.1	Прямой режим автоматического дифференцирования.....	301
8.2.2	Обратный режим автоматического дифференцирования.....	305
Упражнения		306
Глава 9. Регуляризация.....		310
9.1	Индуктивное смещение	311
9.1.1	Обратные задачи.....	311
9.1.2	Теорема об отсутствии бесплатного обеда.....	312
9.1.3	Симметрия и инвариантность	314
9.1.4	Эквивариантность.....	317
9.2	Уменьшение весов.....	318
9.2.1	Последовательные регуляризаторы	320
9.2.2	Обобщенное уменьшение весов	323
9.3	Кривые обучения.....	324
9.3.1	Ранняя остановка	325
9.3.2	Двойной спуск.....	327
9.4	Совместное использование параметров	330
9.4.1	Мягкое разделение весов	331
9.5	Остаточные связи.....	333
9.6	Усреднение модели	337
9.6.1	Прореживание.....	340
Упражнения		342
Глава 10. Сверточные сети		347
10.1	Компьютерное зрение	348
10.1.1	Данные изображений	349
10.2	Сверточные фильтры.....	350
10.2.1	Детекторы признаков	351
10.2.2	Эквивариантный перенос	352
10.2.3	Заполнение	355
10.2.4	Свертки со сдвигом	356
10.2.5	Многомерные свертки	356
10.2.6	Пулинг	358
10.2.7	Многослойные свертки.....	360
10.2.8	Примеры сетевых архитектур	361
10.3	Визуализация обученных CNN.....	364
10.3.1	Зрительная кора головного мозга.....	364
10.3.2	Визуализация обученных фильтров.....	366
10.3.3	Карты значимости.....	368
10.3.4	Состязательные атаки.....	369
10.3.5	Синтетические изображения.....	371
10.4	Определение объектов	372
10.4.1	Ограничительные рамки.....	373
10.4.2	Пересечение по объединению.....	374

10.4.3	Скользящие окна	375
10.4.4	Обнаружение в разных масштабах	377
10.4.5	Немаксимальное подавление	378
10.4.6	Быстрая региональная CNN	379
10.5	Сегментация изображений	380
10.5.1	Сверточная сегментация	380
10.5.2	Повышающая дискретизация	381
10.5.3	Полностью сверточные сети	383
10.5.4	Архитектура U-net	384
10.6	Перенос стиля	385
	Упражнения	387
Глава 11. Структурированные распределения		390
11.1	Модели графов	391
11.1.1	Ориентированные графы	391
11.1.2	Факторизация	392
11.1.3	Дискретные переменные	394
11.1.4	Гауссовые переменные	397
11.1.5	Бинарный классификатор	399
11.1.6	Параметры и наблюдения	400
11.1.7	Теорема Байеса	402
11.2	Условная независимость	403
11.2.1	Три примера графов	404
11.2.2	Объяснения	408
11.2.3	D-разделение	410
11.2.4	Наивный Байес	411
11.2.5	Генеративные модели	413
11.2.6	Покрытие Маркова	415
11.2.7	Графы в качестве фильтров	416
11.3	Модели последовательностей	417
11.3.1	Латентные переменные	420
	Упражнения	421
Глава 12. Трансформеры		425
12.1	Внимание	426
12.1.1	Обработка трансформеров	428
12.1.2	Коэффициенты внимания	430
12.1.3	Самовнимание	431
12.1.4	Сетевые параметры	432
12.1.5	Масштабируемое самовнимание	435
12.1.6	Многоголовое внимание	436
12.1.7	Слои трансформера	438
12.1.8	Вычислительная сложность	440
12.1.9	Позиционное кодирование	440
12.2	Естественный язык	444
12.2.1	Векторное представление слов	444

12.2.2	Лексическая обработка	446
12.2.3	Мультимножество слов	448
12.2.4	Модели авторегрессии	449
12.2.5	Рекуррентные нейронные сети	450
12.2.6	Обратное распространение во времени	452
12.3	Языковые модели трансформеров	453
12.3.1	Декодирующие трансформеры	454
12.3.2	Стратегии выборки	457
12.3.3	Кодирующие трансформеры	460
12.3.4	Трансформеры последовательности в последовательность	462
12.3.5	Большие языковые модели	464
12.4	Мультимодальные трансформеры	467
12.4.1	Визуальные трансформеры	468
12.4.2	Генеративные визуальные трансформеры	470
12.4.3	Аудиоданные	473
12.4.4	Преобразование текста в речь	474
12.4.5	Визуальные и языковые трансформеры	476
	Упражнения	478

Глава 13. Графовые нейронные сети 482

13.1	Машинное обучение на графах	483
13.1.1	Свойства графов	484
13.1.2	Матрица смежности	485
13.1.3	Эквивариантность перестановок	486
13.2	Нейронный обмен сообщениями	488
13.2.1	Сверточные фильтры	488
13.2.2	Графовые сверточные сети	490
13.2.3	Операторы агрегации	491
13.2.4	Операторы обновления	494
13.2.5	Классификация узлов	495
13.2.6	Классификация ребер	496
13.2.7	Классификация графов	496
13.3	Общие графовые сети	497
13.3.1	Графовые сети с вниманием	497
13.3.2	Встраивание ребер	498
13.3.3	Вложения графов	499
13.3.4	Чрезмерное слаживание	500
13.3.5	Регуляризация	501
13.3.6	Геометрическое глубокое обучение	501
	Упражнения	502

Глава 14. Выборка 505

14.1	Основные алгоритмы выборки	505
14.1.1	Ожидаемые значения	505
14.1.2	Стандартные распределения	507
14.1.3	Выборка с отклонением	509

14.1.4	Адаптивная выборка с отклонением.....	511
14.1.5	Выборка по важности	513
14.1.6	Выборка и повторная выборка по значимости	515
14.2	Метод Монте-Карло с цепями Маркова.....	517
14.2.1	Алгоритм Метрополиса	517
14.2.2	Марковские цепи	519
14.2.3	Алгоритм Метрополиса–Гастингса	521
14.2.4	Выборка Гиббса	523
14.2.5	Выборка по предкам	527
14.3	Выборка Ланжевена	528
14.3.1	Модели на основе энергии.....	529
14.3.2	Максимизация правдоподобия	530
	14.3.3 Динамика Ланжевена.....	532
	Упражнения	534

Глава 15. Дискретные латентные переменные 537

15.1	Кластеризация K-средних	538
15.1.1	Сегментация изображений	542
15.2	Гауссовые смеси.....	544
15.2.1	Функция правдоподобия.....	547
15.2.2	Максимальное правдоподобие	549
15.3	Алгоритм ожидания-максимизации.....	554
15.3.1	Гауссовые смеси.....	557
15.3.2	Сравнение с алгоритмом K-средних	559
15.3.3	Смеси распределений Бернули	560
15.4	Нижняя граница доказательств	564
15.4.1	Новый взгляд на ЕМ	566
15.4.2	Независимые и одинаково распределенные данные	568
15.4.3	Априорные параметры.....	568
15.4.4	Обобщенный ЕМ.....	569
	15.4.5 Последовательный ЕМ	570
	Упражнения	571

Глава 16. Непрерывные латентные переменные 575

16.1	Анализ главных компонентов.....	576
16.1.1	Определение максимальной дисперсии.....	577
16.1.2	Определение минимальной ошибки.....	579
16.1.3	Сжатие данных.....	582
16.1.4	Отбеливание данных	583
16.1.5	Данные высокой размерности	585
16.2	Вероятностные латентные переменные.....	586
16.2.1	Генеративная модель	587
16.2.2	Функция правдоподобия.....	588
16.2.3	Максимальное правдоподобие	590
16.2.4	Факторный анализ	594
	16.2.5 Анализ независимых компонентов.....	595

16.2.6	Фильтры Калмана	597
16.3	Нижняя граница доказательств	599
16.3.1	Максимизация ожидания.....	600
16.3.2	EM для PCA	603
16.3.3	EM для факторного анализа.....	604
16.4	Нелинейные модели латентных переменных	605
16.4.1	Нелинейные многообразия	606
16.4.2	Функция правдоподобия.....	608
16.4.3	Дискретные данные	609
16.4.4	Четыре метода генеративного моделирования.....	610
	Упражнения	612
	Глава 17. Генеративные состязательные сети	617
17.1	Состязательное обучение.....	617
17.1.1	Функция потерь	619
17.1.2	Практическое обучение GAN	620
17.2	GAN для обработки изображений.....	623
17.2.1	CycleGAN	624
	Упражнения	628
	Глава 18. Нормализующие потоки	631
18.1	Потоки сопряжения	633
18.2	Потоки авторегрессии	637
18.3	Непрерывные потоки	639
18.3.1	Нейронные дифференциальные уравнения	639
18.3.2	Обратное распространение нейронных ОДУ.....	640
18.3.3	Потоки нейронных ОДУ	642
	Упражнения	644
	Глава 19. Автокодировщики	647
19.1	Детерминированные автокодировщики	647
19.1.1	Линейные автокодировщики	648
19.1.2	Глубокие автокодировщики.....	649
19.1.3	Разреженные автокодировщики.....	651
19.1.4	Шумоподавляющие автокодировщики.....	651
19.1.5	Маскированные автокодировщики	652
19.2	Вариационные автокодировщики.....	655
19.2.1	Амортизированный вывод	657
19.2.2	Метод перепараметризации.....	659
	Упражнения	663
	Глава 20. Диффузионные модели	666
20.1	Прямой кодировщик.....	667
20.1.1	Диффузионное ядро.....	668
20.1.2	Условное распределение	669

20.2	Обратное декодирование.....	670
20.2.1	Обучение декодера.....	673
20.2.2	Нижняя граница доказательств.....	673
20.2.3	Переименование ELBO	675
20.2.4	Прогнозирование шума.....	677
20.2.5	Генерация новых выборок	679
20.3	Соответствие оценок	681
20.3.1	Оценка функции потерь	682
20.3.2	Модифицированная оценка потерь	682
20.3.3	Дисперсия шума	684
20.3.4	Стохастические дифференциальные уравнения.....	685
20.4	Управляемая диффузия	686
20.4.1	Наведение классификатора	687
20.4.2	Наведение без классификатора.....	688
	Упражнения	691
	Приложение А. Линейная алгебра.....	696
A.1	Матричные тождества	696
A.2	Следы и определители.....	697
A.3	Производные матрицы.....	698
A.4	Собственные векторы.....	700
	Приложение В. Вариационное исчисление	704
	Приложение С. Множители Лагранжа.....	707
	Список литературы	711
	Предметный указатель.....	731

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Предисловие

Глубокое обучение с использованием многослойных нейронных сетей, натренированных на больших массивах данных с целью решения сложных задач обработки информации, считается наиболее успешной парадигмой в области машинного обучения. За последнее десятилетие глубокое обучение произвело революцию во многих предметных областях, включая компьютерное зрение, распознавание речи и обработку естественного языка. Оно находит все большее применение в здравоохранении, промышленности, коммерции, финансовой сфере, науке и многих других отраслях. Совсем недавно было установлено, что масштабные нейронные сети, также известные как большие языковые модели с количеством обучаемых параметров порядка триллиона, демонстрируют первые признаки общих свойств искусственного интеллекта. Сегодня они выступают в качестве основного движущего фактора крупнейшего в истории технологического прорыва.

Основные задачи этой книги

Рост популярности глубокого обучения сопровождается стремительным увеличением количества и разнообразия научных публикаций в области машинного обучения наряду с ускорением инновационных процессов. Для новичков в этой области даже освоение базовых идей может показаться весьма сложной задачей, не говоря уж о переходе к передовым научным исследованиям. Исходя из этого, книга «*Глубокое обучение: принципы и концепции*» призвана обеспечить начинающим специалистам в области машинного обучения, а также тем, кто уже имеет опыт работы в этой сфере, глубокое понимание как фундаментальных идей, лежащих в основе глубокого обучения, так и ключевых концепций современных архитектур и методов глубокого обучения. Этот материал поможет читателю получить крепкую основу для будущей специализации. В связи с масштабностью и темпами изменений в этой области авторы намеренно отказались от попыток составить всеобъемлющий обзор всех последних исследований. Вместо этого значительная ценность книги заключается в представлении ключевых идей. Учитывая, что эта область, как можно ожидать, продолжит свое стремительное развитие, эти основы и концепции, по всей видимости, смогут выдержать испытание временем. Например, на момент написания книги большие языковые модели развивались

очень быстро, однако лежащая в их основе архитектура преобразования и механизм концентрации внимания оставались практически неизменными в течение последних пяти лет, в то время как множество основных принципов машинного обучения хорошо известны на протяжении десятилетий.

Ответственное применение технологий

Глубокое обучение представляет собой мощную технологию с обширными возможностями применения. Она обладает огромным потенциалом для создания новых ценностей и решения наиболее актуальных проблем современного общества. Однако эти же качества обуславливают возможность преднамеренного злоупотребления глубоким обучением или причинения неумышленного вреда. Мы не стали обсуждать этические и социальные аспекты использования глубокого обучения, поскольку эти темы настолько важны и сложны, что требуют более тщательного рассмотрения, чем это возможно в подобном техническом издании. Вместе с тем рассуждения на такие темы должны опираться на глубокие знания об основах технологии и ее работе, поэтому мы надеемся, что эта книга станет ценным вкладом в подобные серьезные исследования. При этом мы настоятельно рекомендуем читателям не оставлять без внимания более широкие последствия своей работы и изучать вопросы ответственного использования глубокого обучения и искусственного интеллекта наряду с изучением самой технологии.

Структура этой книги

Книга состоит из достаточно большого количества компактных глав, каждая из которых посвящена определенной теме. Книга построена по линейному принципу, поскольку каждая глава опирается только на материал, рассмотренный в предыдущих главах. Она хорошо подходит для преподавания курса машинного обучения в течение двух семестров для студентов или аспирантов, но в равной степени актуальна и для тех, кто активно занимается исследованиями или самообразованием.

Четкое понимание принципов машинного обучения может быть достигнуто только с помощью определенного уровня знаний математики. Говоря конкретно, в основе машинного обучения лежат три области математики: теория вероятности, линейная алгебра и многомерный математический анализ. Книга обеспечивает последовательное введение в необходимые концепции теории вероятности и включает приложение, в котором обобщены некоторые полезные практические методы линейной алгебры. Предполагается, что читатель уже знаком с основными концепциями многомерного анализа, однако в приложениях есть вводные сведения по вариационному исчислению и методу множителей Лагранжа. Основное внимание в книге тем

не менее уделено донесению до читателя четкого понимания изложенных концепций с акцентом на методы, имеющие реальную практическую ценность, а не на абстрактную теорию. Там, где это возможно, мы постарались представить более сложные концепции с нескольких взаимодополняющих точек зрения, включая текстовое описание, диаграммы и математические формулы. Кроме того, многие обсуждаемые в тексте ключевые алгоритмы кратко изложены в отдельных врезках. Они не связаны с вопросами эффективности вычислений, но служат дополнением к математическим выкладкам в тексте. Поэтому мы надеемся, что материал этой книги окажется доступным для читателей с разным уровнем подготовки.

В концептуальном плане эту книгу, скорее всего, стоит рассматривать как продолжение книги «Нейронные сети для распознавания образов» (*Neural Networks for Pattern Recognition*, Bishop, 1995b), в которой были впервые даны исчерпывающие сведения о нейронных сетях с позиций математической статистики. Ее также можно рассматривать как дополнение к книге «Распознавание образов и машинное обучение» (*Pattern Recognition and Machine Learning*, Bishop, 2006), в которой рассматривается более широкий спектр тем машинного обучения, несмотря на то что она была написана до начала революции глубокого обучения. Вместе с тем, чтобы обеспечить самодостаточность этой новой книги, соответствующий материал был перенесен из книги Bishop (2006) и переработан так, чтобы сосредоточиться именно на тех основополагающих идеях, которые важны при глубоком обучении. Это означает, что многие интересные темы в машинном обучении, рассмотренные в Bishop (2006), остаются интересными и сегодня, но в новой книге они не затрагиваются. Например, в Bishop (2006) довольно подробно рассматриваются байесовские методы, в то время как в этой книге они почти полностью исключены.

Справочная информация

Следуя принципу акцентирования внимания на ключевых идеях, мы не пытаемся представить всеобъемлющий обзор литературы, что в любом случае было бы невозможно с учетом масштабов и темпов изменений в этой сфере. Тем не менее мы приводим ссылки на некоторые важнейшие научные работы, а также обзорные статьи и другие источники для дополнительного чтения. Во многих случаях они также содержат важные детали практического применения, которые мы опускаем в тексте с целью не отвлекать читателя от обсуждения ключевых концепций.

На тему машинного обучения в целом и глубокого обучения в частности уже написано множество книг. Среди наиболее близких по уровню и стилю к этой книге можно отметить Bishop (2006), Goodfellow, Bengio and Courville (2016), Murphy (2022), Murphy (2023) и Prince (2023).

За последнее десятилетие специфика научной деятельности в области машинного обучения существенно изменилась: многие работы публикуются на

архивных сайтах до или даже вместо направления на экспертные конференции и в журналы. Самый популярный из таких сайтов – arXiv, что означает «архив», и доступен он по адресу <https://arXiv.org>.

Этот сайт позволяет обновлять статьи, что часто приводит к появлению нескольких версий в разные календарные годы, и это может привести к возникновению разнотечений в отношении цитирования той или иной версии и того или иного года. Кроме того, на сайте имеется бесплатный доступ к PDF-файлу каждой статьи. Поэтому мы используем простой подход – ссылааться на статью в соответствии с годом ее первой загрузки, хотя рекомендуем читать самую последнюю версию.

Статьи на arXiv индексируются с использованием обозначения arXiv:YYMM.XXXX, где YY и MM означают год и месяц первой загрузки соответственно. Последующие версии обозначаются добавлением номера версии N в форме arXiv:YYMM.XXXXXvN.

Упражнения

В конце каждой главы приводится подборка упражнений для закрепления ключевых идей, изложенных в тексте, или для их углубленного анализа и обобщения. Эти упражнения являются важной частью текста, и каждое из них оценивается по степени сложности: от (*), обозначающей простое упражнение на несколько минут, до (**), обозначающей существенно более сложное упражнение. Настоятельно рекомендуется выполнять упражнения, поскольку активная работа с материалом значительно повышает эффективность обучения. Решения всех упражнений доступны в виде PDF-файла, который можно загрузить с веб-сайта книги.

Математические обозначения

Мы используем те же обозначения, что и в книге (Bishop 2006). Обзор математики в контексте машинного обучения представлен в книге (Deisenroth, Faisal and Ong 2020).

Векторы обозначаются строчными полужирными латинскими буквами, например \mathbf{x} , а матрицы обозначаются прописными полужирными латинскими буквами, например \mathbf{M} . Если не указано иное, предполагается, что все векторы являются векторами-столбцами. Надстрочный индекс T обозначает транспонирование матрицы или вектора, так что \mathbf{x}^T будет представлять собой строчный вектор. Обозначение (w_1, \dots, w_M) подразумевает вектор строки с M элементами, а соответствующий вектор столбца записывается как $\mathbf{w} = (w_1, \dots, w_M)^T$. Матрица тождественности $M \times M$ (также известная как единичная матрица) обозначается \mathbf{I}_M , и, если нет двусмыслинности относительно ее размерности, она будет сокращаться до \mathbf{I} . Ее элементы I_{ij} равны 1, если $i = j$, и 0, если

$i \neq j$. Элементы единичной матрицы иногда обозначают как δ_{ij} . Обозначение **1** означает вектор-столбец, в котором все элементы имеют значение 1. Запись **a** \oplus **b** обозначает конкатенацию векторов **a** и **b**, так что если **a** = (a_1, \dots, a_N) и **b** = (b_1, \dots, b_M), то **a** \oplus **b** = ($a_1, \dots, a_N, b_1, \dots, b_M$). Запись $|x|$ обозначает модуль (положительную часть) скаляра x , также известного как абсолютное значение. Мы используем запись $\det \mathbf{A}$ для обозначения определителя матрицы **A**.

Запись $x \sim p(x)$ означает, что x выбирается из распределения $p(x)$. Там, где возможны разночтения, мы будем использовать подстрочные индексы, как $p_x(\cdot)$, чтобы выразить, о какой плотности идет речь. Математическое ожидание функции $f(x, y)$ относительно случайной переменной x обозначается $\mathbb{E}_x[f(x, y)]$. В ситуациях, когда нет двусмыслинности относительно того, по какой переменной производится усреднение, суффикс опускается, например $\mathbb{E}[x]$. Если распределение x зависит от другой переменной z , то соответствующее условное ожидание будет записано как $\mathbb{E}_x[f(x)|z]$. Аналогично дисперсия $f(x)$ обозначается $\text{var}[f(x)]$, а для векторных переменных ковариация записывается как $\text{cov}[\mathbf{x}, \mathbf{y}]$. Мы также будем использовать $\text{cov}[\mathbf{x}]$ как сокращенное обозначение для $\text{cov}[\mathbf{x}, \mathbf{x}]$.

Символ \forall означает «для всех», так что $\forall t \in \mathcal{M}$ обозначает все значения t в пределах множества \mathcal{M} . Мы используем R для обозначения вещественных чисел. На графе множество соседних узлов i обозначается как $\mathcal{N}(i)$, которое не следует путать с гауссовым или нормальным распределением $\mathcal{N}(x|\mu, \sigma^2)$. Функционал обозначается $f[y]$, где $y(x)$ – это некоторая функция. Понятие функционала рассматривается в приложении B. Фигурные скобки $\{ \}$ обозначают множество. Запись $g(x) = \mathcal{O}(f(x))$ означает, что $|f(x)/g(x)|$ ограничено при $x \rightarrow \infty$. Например, если $g(x) = 3x^2 + 2$, то $g(x) = \mathcal{O}(x^2)$. Обозначение $[x]$ означает «нижнюю целую часть числа» x , т. е. наибольшее целое число, которое меньше или равно x .

Если имеется N независимых и одинаково распределенных (independent and identically distributed, i.i.d.) значений $\mathbf{x}_1, \dots, \mathbf{x}_N$ D -мерного вектора $\mathbf{x} = (x_1, \dots, x_D)^T$, мы можем объединить эти результаты наблюдений в матрицу данных \mathbf{X} размерности $N \times D$, в которой n -я строка \mathbf{X} соответствует вектору строк \mathbf{x}_n^T . Так, n, i элемент \mathbf{X} соответствует i -му элементу n -го наблюдения \mathbf{x}_n и записывается как x_{ni} . Для одномерных переменных мы обозначаем такую матрицу через \mathbf{x} , т. е. вектор-столбец, n -й элемент которого равен x_n . Обратите внимание, что для обозначения \mathbf{x} (размерность N) используется другой шрифт, чтобы отличить его от \mathbf{x} (размерность D).

Благодарности

Мы хотели бы выразить искреннюю благодарность всем тем, кто просмотрел проекты глав и предоставил ценные отзывы. В частности, выражаем благодарность Samuel Albanie, Cristian Bodnar, John Bronskill, Wessel Bruinsma, Ignas Budvytis, Chi Chen, Yaoyi Chen, Long Chen, Fergal Cotter, Sam Devlin, Aleksander Durumeric, Sebastian Ehlert, Katarina Elez, Andrew Foong, Hong Ge, Paul Gladkov, Paula Gori Giorgi, John Gossman, Tengda Han, Juyeon Heo, Katja Hofmann, Chin-

Wei Huang, Yongchaio Huang, Giulio Isacchini, Matthew Johnson, Pragya Kale, Atharva Kelkar, Leon Klein, Pushmeet Kohli, Bonnie Kruft, Adrian Li, Haiguang Liu, Ziheng Lu, Giulia Luise, Stratis Markou, Sergio Valcarcel Macua, Krzysztof Maziarz, Matej Mezera, Laurence Midgley, Usman Munir, Felix Musil, Elise van der Pol, Tao Qin, Isaac Reid, David Rosenberger, Lloyd Russell, Maximilian Schebek, Megan Stanley, Karin Strauss, Clark Templeton, Marlon Tobaben, Aldo Sayeg Pasos-Trejo, Richard Turner, Max Welling, Furu Wei, Robert Weston, Chris Williams, Yingce Xia, Shufang Xie, Iryna Zaporozhets, Claudio Zeni, Xieyuan Zhang и многим другим коллегам, которые внесли свой неоценимый вклад в наши обсуждения.

Мы также хотели бы поблагодарить нашего редактора Пола Другаса (Paul Drougas) и многих других сотрудников издательства Springer, а также корректора Джонатана Уэбли (Jonathan Webley) за их поддержку во время работы над книгой.

Особую благодарность мы хотели бы выразить Маркусу Свенсену (Markus Svensen), который оказал существенную помощь в работе над рисунками и набором текста для книги (Bishop, 2006), в том числе над файлами стилей LATEX, которые были использованы и в этой новой книге. Мы также благодарны многим ученым, которые разрешили нам воспроизвести диаграммы из их публикаций. Благодарности за конкретные рисунки приведены в подписях к ним.

Крис хотел бы выразить искреннюю благодарность компании Microsoft за создание стимулирующей исследовательской среды и за предоставленную возможность написать эту книгу. Однако взгляды и мнения, выраженные в этой книге, отражают точку зрения авторов и не обязательно совпадают с позицией Microsoft или ее аффилированных лиц. Для меня было огромной привилегией и удовольствием сотрудничать с моим сыном Хью в процессе подготовки этой книги, начавшейся как совместный проект во время первой изоляции Covid.

Хью хотел бы поблагодарить компанию Wayve Technologies Ltd. за любезное разрешение работать неполный рабочий день ради возможности участия в написании этой книги, а также за создание вдохновляющей и благоприятной среды, в которой он работал и учился. Мнения, выраженные в этой книге, не обязательно совпадают с позицией компании Wayve или ее аффилированных лиц. Он хотел бы выразить благодарность своей невесте Джемайме за постоянную поддержку, а также за консультации по грамматике и стилистике. Он также хотел бы поблагодарить Криса, который был отличным коллегой и вдохновлял Хью на протяжении всей его жизни.

В заключение мы оба хотели бы выразить огромную благодарность членам нашей семьи Дженне и Марку за очень многое, что даже невозможно здесь перечислить. Кажется, что это было очень давно, когда мы все собирались на пляже в Антальи, чтобы посмотреть на полное затмение солнца и сделать семейное фото для страницы посвящения журнала *Pattern Recognition and Machine Learning!*

Крис Бишоп и Хью Бишоп.
Кембридж, Великобритания.
Октябрь, 2023 г.

Глава 1

Революция глубокого обучения

На сегодняшний день машинное обучение является одним из наиболее приоритетных и быстро развивающихся технологических направлений. Приложения машинного обучения получают повсеместное распространение, а решения на основе полученных данных все больше замещают традиционные ручные алгоритмы. Это ведет не только к повышению эффективности уже существующих технологий, но также открывает возможности для реализации широкого спектра новых идей, которые были бы немыслимы при разработке новых алгоритмов вручном режиме.

Одним из направлений машинного обучения, известным как *глубокое обучение* (*deep learning*), является чрезвычайно эффективная и универсальная система обучения на основе данных. Глубокое обучение основано на использовании вычислительных моделей, называемых *нейронными сетями* (*neural networks*), на создание которых изначально повлияли механизмы обучения и обработки информации в человеческом мозге. Область *искусственного интеллекта*, или ИИ, стремится воссоздать в машинах мощнейшие возможности мозга, и сегодня термины «машинаное обучение» и «ИИ» зачастую используются как взаимозаменяемые. Многие из используемых в настоящее время систем ИИ представляют собой приложения машинного обучения, предназначенные для решения очень специфических и узкоспециальных задач, и, несмотря на их исключительную ценность, они все еще далеки от колоссального спектра возможностей человеческого мозга. Именно поэтому для обозначения перспектив создания машин с намного большей функциональностью был введен термин «искусственный интеллект общего назначения» (*artificial general intelligence*, или AGI). После многих десятилетий стабильного прогресса машинное обучение вступило в фазу чрезвычайно быстрого развития. Совсем недавно масштабные системы глубокого обучения, называемые большими языковыми моделями (*large language models*, см. главу 12), смогли продемонстрировать поразительные возможности, которые уже называют первыми признаками искусственного интеллекта назначения (Bubeck et al., 2023).

1.1. Влияние глубокого обучения

Знакомство с машинным обучением мы начнем с изучения четырех примеров из разных областей для иллюстрации обширных возможностей применения этой технологии и введения некоторых базовых принципов и терминологии. Особенность этих и многих других примеров заключается в том, что все они рассматриваются как варианты использования одной и той же фундаментальной концепции глубокого обучения. Это резко контрастирует с традиционными подходами, когда для решения различных задач используются совершенно разные и специфические методы. Следует подчеркнуть, что выбранные нами примеры представляют собой лишь незначительную область применения глубоких нейронных сетей, и почти каждая область, где вычисления играют определенную роль, может получить преобразующий эффект от глубокого обучения.

1.1.1. Медицинская диагностика

Для начала рассмотрим применение машинного обучения для диагностики рака кожи. Меланома – самый опасный вид рака кожи, но при условии раннего обнаружения она излечима. На рис. 1.1 показаны примеры снимков кожных поражений: в верхнем ряду – злокачественные меланомы, в нижнем – доброкачественные невусы. Безусловно, отличить эти два класса образов очень сложно, и было бы практически невозможно написать вручную алгоритм, который смог бы успешно классифицировать такие образцы с приемлемой степенью точности.

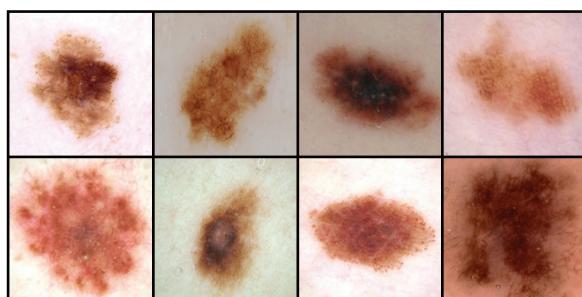


РИС. 1.1 Примеры поражений кожи: опасные злокачественные меланомы (верхний ряд), доброкачественные невусы (нижний ряд)

Эту проблему удалось успешно решить с помощью глубокого обучения (Esteva et al., 2017). Решение было создано с использованием большого набора снимков новообразований, известного как *обучающий набор (training set)*. Каждый из них помечен как злокачественный или доброкачественный, а сами метки получены из теста биопсии, который используется для определения истинного класса новообразования. Набор обучающих данных ис-

пользуется для определения значений порядка 25 млн настраиваемых параметров, которые называют *весами (weight)*, в глубокой нейронной сети. Этот процесс определения значений параметров на основе данных известен как *обучение (learning)*, или *тренировка (training)*. Задача состоит в том, чтобы обученная сеть могла предсказать правильную метку для новообразования только на основании полученных изображений, не прибегая к трудоемкому этапу взятия биопсии. Это пример задачи обучения под наблюдением, поскольку для каждого обучающего примера сети указывается правильная метка. Это также пример задачи *классификации (classification problem)*, поскольку каждый входной сигнал должен быть отнесен к дискретному набору классов (в данном случае – доброкачественных или злокачественных новообразований). Прложения, в которых выход состоит из одной или нескольких непрерывных переменных, называются задачами *регрессии (regression problems)*. Примером регрессионной задачи может служить прогнозирование выхода продукции в процессе химического производства, в котором в качестве входных данных выступают температура, давление и концентрация реагентов.

Интересно отметить, что количество доступных обучающих изображений (примерно 129 000) считается относительно небольшим, поэтому глубокая нейронная сеть сначала обучалась на гораздо большем наборе данных, состоящем из 1,28 млн изображений привычных объектов (таких как собаки, здания и грибы), а затем настраивалась на наборе данных с изображениями новообразований. Это пример *трансферного обучения (transfer learning)*, при котором сеть изучает общие свойства естественных изображений на основе большого набора данных о повседневных объектах, а затем специализированно подстраивается под конкретную задачу классификации новообразований. Благодаря использованию глубокого обучения точность классификации изображений поражений кожи оказалась выше, чем у профессиональных дерматологов (Brinker et al., 2019).

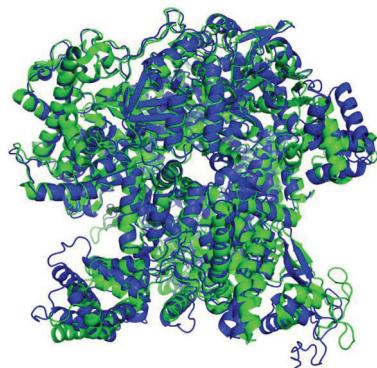
1.1.2. Структура белка

Белки часто называют строительными блоками живых организмов. Они представляют собой биологические молекулы из одной или нескольких длинных цепочек элементов, называемых аминокислотами, которых насчитывается 22 разновидности, при этом разновидности белков определяются именно последовательностью аминокислот. После завершения синтеза белка в живой клетке он образует сложную трехмерную структуру, характеристики и взаимосвязи которой во многом определяются его формой. Расчет такой трехмерной структуры с учетом последовательности аминокислот уже полвека является фундаментальной нерешенной проблемой биологии, и до появления глубокого обучения прогресс в этой области был относительно скромным.

Определение 3D-структуры возможно с помощью таких экспериментальных методов, как рентгеновская кристаллография, криогенная электронная микроскопия или спектроскопия ядерного магнитного резонанса. Однако

эти методы требуют значительных затрат времени, а для некоторых белков могут оказаться слишком сложными, например, из-за проблем с получением чистого образца или из-за фактической зависимости структуры от среды. С другой стороны, последовательность аминокислот белка может быть определена экспериментально с меньшими затратами и большей скоростью. Именно поэтому наблюдается значительный интерес к способам прогнозирования 3D-структуры белков непосредственно по их аминокислотным последовательностям для лучшего понимания биологических процессов или для практического применения, например для создания лекарств. Модель глубокого обучения может быть обучена для получения на входе аминокислотной последовательности и формирования 3D-структуры на выходе, при этом обучающие данные представляют собой набор белков, для которых известны и последовательность аминокислот, и 3D-структура. Определение структуры белка – это еще один пример контролируемого обучения. После обучения система может использовать новую последовательность аминокислот в качестве входных данных и определять соответствующую 3D-структуру (Jumper et al., 2021). На рис. 1.2 показано сравнение прогнозируемой 3D-структуры белка и реальной структуры, снимок которой получен с помощью рентгеновской кристаллографии. Зеленым цветом показана истинная структура, определенная с помощью рентгеновской кристаллографии, наложенная синяя структура – это прогноз с помощью модели глубокого обучения AlphaFold.

РИС. 1.2 Иллюстрация трехмерной формы белка T1044/6VR4. [C разрешения (Jumper et al., 2021)]



1.1.3. Синтез изображений

В двух рассмотренных ранее приложениях нейронная сеть была обучена преобразовывать входные данные (изображение кожи или последовательность аминокислот) в выходные (классификация новообразований или трехмерная структура белка соответственно). Теперь обратимся к примеру, в котором обучающие данные состоят из простого набора образцов изображений, а целью обученной сети является создание новых изображений подобного рода. В отличие от примеров с классификацией новообразований и структурой

белка, это пример *обучения без контроля (unsupervised learning)*, поскольку изображения не маркированы. На рис. 1.3 показаны примеры синтетических изображений, созданных глубокой нейронной сетью. Она была обучена на наборе изображений человеческих лиц, снятых в студии на обычном фоне. Такие синтетические изображения отличаются исключительно высоким качеством, и их порой достаточно сложно отличить от фотографий реальных людей.



РИС. 1.3 Синтетические изображения лиц авторства глубокой нейронной сети, обученной с помощью бесконтрольного обучения. [С сайта <https://generated.photos>]

Это пример *генеративной модели (generative model)*, поскольку она способна производить новые выходные примеры, отличающиеся от используемых для ее обучения, но обладающие теми же статистическими свойствами. Один из вариантов этого подхода позволяет генерировать изображения в зависимости от содержания входной текстовой строки, которую называют «подсказкой» (*prompt*), чтобы содержание изображения отражало семантику введенного текста. Термин «генеративный ИИ» (*generative AI*) используется для описания (см. главу 10) моделей глубокого обучения, которые генерируют выходные данные в виде изображений, видео, аудио, текста, молекул-кандидатов лекарств или других структур.

1.1.4. Большие языковые модели

Одним из наиболее важных достижений в области машинного обучения за последние годы стала разработка эффективных моделей для обработки естественного языка и других форм последовательных данных, таких как исходный код. *Большая языковая модель (Large Language Model, или LLM)* использует глубокое обучение для построения подробных внутренних представлений, отражающих семантические свойства языка. Значительный класс больших

языковых моделей, называемых *авторегрессионными языками моделями* (*autoregressive language models*), позволяет генерировать на выходе языковые данные, поэтому они являются разновидностью генеративного ИИ. Такие модели получают на входе последовательность слов, а на выходе генерируют одно слово, которое представляет собой следующее слово в последовательности. Дополненная последовательность с добавлением нового слова в конце может быть снова пропущена через модель для получения следующего слова, и этот процесс может неоднократно повторяться для создания длинной последовательности слов. Такие модели также способны выводить специальное слово «стоп», сигнализирующее об окончании генерации текста, что позволяет им выводить текст конечной длины с последующим окончанием процесса. В этот момент пользователь может добавить к последовательности свою собственную серию слов, а затем прогнать всю последовательность через модель для дальнейшей генерации слов. Это позволяет человеку общаться с нейронной сетью.

Такие модели можно обучать на больших наборах текстовых данных, извлекая обучающие пары слов из случайно выбранной последовательности слов на входе и известного следующего слова в качестве целевого выхода. Это пример *самоконтролируемого обучения* (*self-supervised learning*), в ходе которого выполняется обучение функции от входных данных к выходным, но при этом маркированные выходные данные получаются автоматически из входных обучающих данных без необходимости в отдельных метках, присваиваемых человеком. Поскольку большие объемы текста можно получить из множества источников, этот подход позволяет масштабировать его до очень больших обучающих наборов и, соответственно, очень больших нейронных сетей.

Большие языковые модели могут демонстрировать необыкновенные возможности, которые уже называют первыми признаками зарождающегося искусственного общего интеллекта (Bubeck et al., 2023), и мы подробно обсуждаем такие модели далее в этой книге (см. главу 12). Ниже приведен иллюстративный пример генерации языка на основе модели GPT-4 (OpenAI, 2023) в ответ на запрос «Напишите доказательство того, что существует бесконечно много простых чисел; сделайте это в стиле пьесы Шекспира через диалог между двумя сторонами, спорящими о доказательстве» (на английском языке).

GPT-4: Proof of infinitely many primes

FIRST PARTY:

Thou dost not understand, my friend,
There simply is no upper end
To primes, those numbers indivisible,
No matter if it seems impossible.

SECOND PARTY:

But how dost thou so surely know?
To prove this truth, what canst thou show?

FIRST PARTY:

Consider this, my doubtful peer,
A clever proof that will make clear:
Assume that there's a final prime,
The largest one we'll see in time.
Let's call this number "P" for short,
And then, my friend, we will purport
To build a number "Q" that's new,
One larger than "P" and prime too.

SECOND PARTY:

But how, my friend, will this be done?
The proof is not yet fairly won.

FIRST PARTY:

It's simple, friend, just wait and see,
We'll multiply the primes, that's key.
All primes from two to "P" combined,
Will make this new "Q" we will find.
But one thing more we must include,
To make the proof more watertight,
We'll add a one onto the end,
And now the proof is in our sight.

SECOND PARTY:

I see it now, it's crystal clear,
For "Q" must be a prime, my dear.
It can't divide by any prime,
All smaller than itself in time.
And since we chose "P" as the last,
This proves that our assumption's past.
There is no final prime, we see,
The numbers go on endlessly.

1.2. Учебный пример

Для новичков в области машинного обучения многие основные концепции и терминология могут быть представлены в контексте простого примера, связанного с подгонкой полинома к небольшому синтетическому набору данных (Bishop, 2006). Это одна из форм задачи контролируемого обучения, в рамках которой мы хотим получить прогноз для целевой переменной с учетом значения входной переменной.

1.2.1. Синтетические данные

Обозначим входную переменную как x , а целевую переменную как t , и предположим, что обе переменные будут иметь непрерывные значения на действительной оси. Предположим, что имеется обучающий набор из N наблюде-

ний x , обозначенных как x_1, \dots, x_N , вместе с соответствующими наблюдениями значений t , обозначенных как t_1, \dots, t_N . Наша цель – определить значение t для некоторого нового значения x . Возможность точно определять значение t по неизвестным ранее входным данным является ключевой задачей машинного обучения и называется *обобщением* (*generalization*).

Это можно проиллюстрировать на примере синтетического набора данных, созданного путем выборки из синусоидальной функции. На рис. 1.4 показан график обучающего набора, состоящего из $N = 10$ точек данных, в котором входные значения были получены путем выбора значений x_n для $n = 1, \dots, N$, равномерно распределенных в диапазоне $[0, 1]$. Соответствующие целевые значения данных были получены путем вычисления значений функции $\sin(2\pi x)$ для каждого значения x , а затем добавления небольшого уровня случайного шума (управляемого гауссовым распределением, см. раздел 2.3) к каждой такой точке для получения соответствующего целевого значения t_n . Получая данные таким образом, можно уловить важное свойство многих наборов реальных данных, а именно то, что они обладают основной закономерностью, которая и является целью изучения, но при этом отдельные наблюдения испорчены случайным шумом. Этот шум может быть вызван внутренними стохастическими (т. е. случайными) процессами, такими как радиоактивный распад, но чаще всего он обусловлен наличием источников вариативности, которые сами по себе не наблюдаются. На рис. 1.4 зеленая кривая показывает функцию $\sin(2\pi x)$, используемую для генерации данных. Наша цель – предсказать значение t для некоторого нового значения x без информации о зеленой кривой.

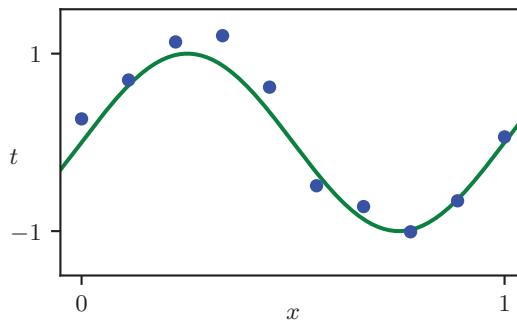


РИС. 1.4 График обучающего набора данных из $N = 10$ точек, показанных синими кружками, каждая из которых представляет собой наблюдение входной переменной x и соответствующей целевой переменной t

В этом учебном примере нам известен истинный процесс, породивший данные, а именно синусоидальная функция. В практическом применении машинного обучения наша цель заключается в выявлении основных тенденций в данных с учетом имеющегося конечного обучающего набора. Тем не менее знание процесса, породившего данные, позволяет наглядно продемонстрировать важные концепции машинного обучения.

1.2.2. Линейные модели

Наша цель – использовать этот обучающий набор для определения значения \hat{t} целевой переменной для некоторого нового значения \hat{x} входной переменной. Как будет показано далее, это подразумевает стремление неявно обнаружить лежащую в основе функцию $\sin(2\pi x)$. По своей сути это сложная задача, поскольку нам приходится производить обобщение с ограниченного набора данных на всю функцию. Кроме того, наблюдаемые данные испорчены шумом, и, следовательно, для заданного \hat{x} существует (см. главу 2) неопределенность в отношении подходящего значения \hat{t} . Теория вероятностей обеспечивает основу для выражения такой неопределенности точным и количественным образом, а теория принятия решений (см. главу 5) позволяет задействовать это вероятностное представление для составления предложений, которые являются оптимальными в соответствии с соответствующими критериями. Изучение вероятностей на основе данных лежит в основе машинного обучения и подробно рассматривается в этой книге.

Для начала, впрочем, ограничимся неформальным подходом и рассмотрим простое решение на основе подбора кривых. В частности, для подгонки данных будет использована полиномиальная функция вида

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^M w_j x^j, \quad (1.1)$$

где M означает порядок полинома, а x^j обозначает x , возведенное в степень j . Коэффициенты полинома w_0, \dots, w_M обозначаются вектором \mathbf{w} . Обратите внимание, что, хотя полиномиальная функция $y(x, \mathbf{w})$ является нелинейной функцией x , она линейна по коэффициентам \mathbf{w} . Подобные полиномиальные функции, линейные по неизвестным параметрам, обладают важными свойствами, а также существенными ограничениями, и называются *линейными моделями* (*linear models*, см. главу 4).

1.2.3. Функция ошибки

Значения коэффициентов определяются путем подгонки полинома под обучающие данные. Это может быть сделано благодаря минимизации функции ошибки (*error function*), которая измеряет несоответствие между функцией $y(x, \mathbf{w})$ для любого заданного значения \mathbf{w} и точками данных обучающего набора. Одним из наиболее простых и широко используемых вариантов функции ошибки является сумма квадратов разностей между оценками $y(x_n, \mathbf{w})$ для каждой точки данных x_n и соответствующим целевым значением t_n , которая определяется следующим образом:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y_n(x_n, \mathbf{w}) - t_n\}^2, \quad (1.2)$$

где коэффициент $1/2$ добавлен для удобства расчетов в дальнейшем. Позже (см. раздел 2.3.4) эту функцию ошибки мы выведем из теории вероятностей.

Здесь просто отметим, что это неотрицательная величина, которая будет равна нулю тогда и только тогда, когда функция $y(x, w)$ будет проходить точно через каждую точку обучающих данных. Геометрическая интерпретация функции ошибки суммы квадратов показана на рис. 1.5.

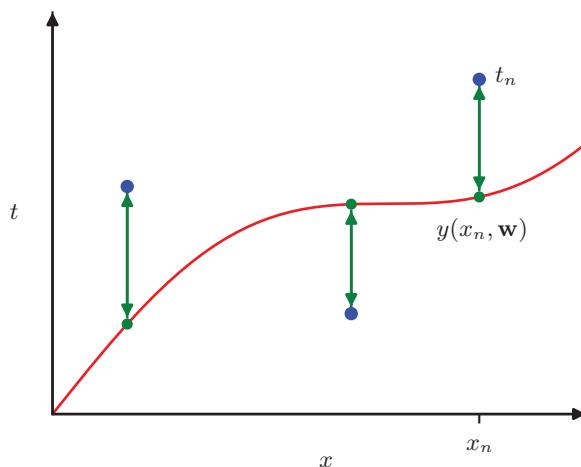


РИС. 1.5 Функция ошибки (1.2) соответствует (одной половине) суммы квадратов смещений (показаны вертикальными зелеными стрелками) каждой точки данных от функции $y(x, w)$

Можно решить задачу подгонки кривой путем выбора такого значения w , при котором значение $E(w)$ будет как можно меньше. Поскольку функция ошибки является квадратичной функцией коэффициентов w , ее производные по коэффициентам будут линейны по элементам w , и, таким образом, минимизация функции ошибки имеет единственное решение, обозначаемое w^* (см. упражнение 4.1), которое может быть найдено в замкнутой форме. Результирующий полином задается функцией $y(x, w^*)$.

1.2.4. Сложность модели

Остается проблема выбора порядка M полинома, и, как будет показано далее, это окажется примером важного определения, называемого *сравнением моделей* (*model comparison*) или *выбором модели* (*model selection*). На рис. 1.6 приведены четыре примера результатов подгонки полиномов с порядками $M = 0, 1, 3$ и 9 к набору данных, показанному на рис. 1.4.

Обратите внимание, что полиномы постоянного ($M = 0$) и первого ($M = 1$) порядков плохо согласуются с данными и, следовательно, плохо представляют функцию $\sin(2\pi x)$. Полином третьего порядка ($M = 3$), по всей видимости, наилучшим образом отражает функцию $\sin(2\pi x)$ из примеров, показанных на рис. 1.6. При переходе к полиному более высокого порядка ($M = 9$) мы получаем отличную подгонку к обучающим данным. Фактически полином проходит точно через каждую точку данных и $E(w^*) = 0$. Однако подогнанная кривая

сильно колеблется и дает очень плохое представление о функции $\sin(2\pi x)$. Такое поведение называют чрезмерной подгонкой (*over-fitting*).

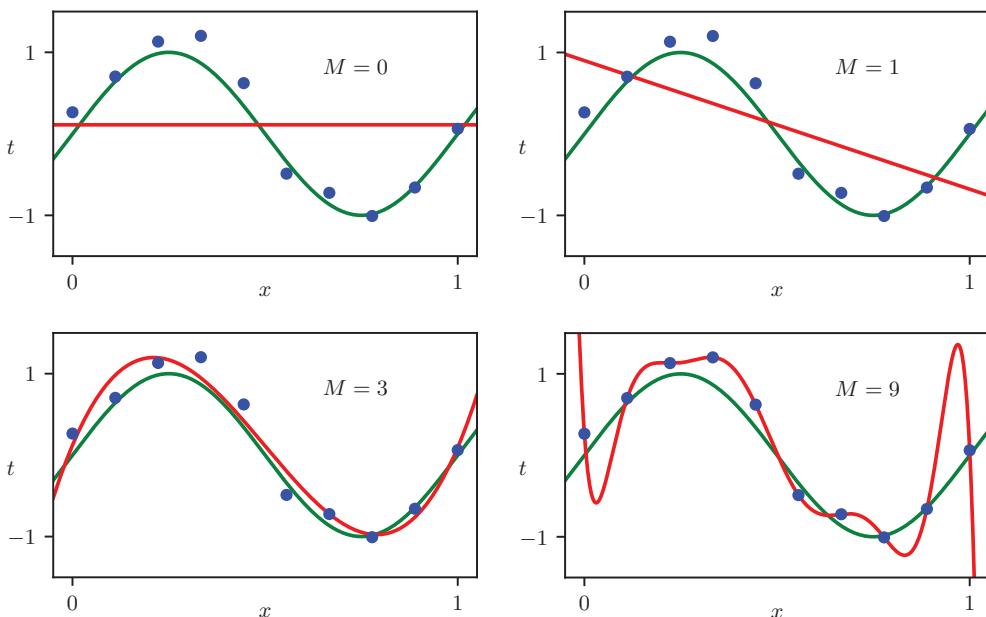


РИС. 1.6 Графики полиномов различных порядков M (красные кривые), подогнанные к набору данных на рис. 1.4 путем минимизации функции ошибки (1.2)

Целью этой задачи является достижение качественного обобщения путем составления точных предположений для новых данных. Некоторое количественное представление о зависимости эффективности обобщения от M можно получить при анализе отдельного набора данных, известного как *тестовый набор* (*test set*). Он состоит из 100 точек данных, полученных с помощью той же процедуры, которая использовалась для генерации точек обучающего набора. Для каждого значения M можно оценить остаточное значение $E(\mathbf{w}^*)$, заданное (1.2) для обучающих данных, а также оценить $E(\mathbf{w}^*)$ для тестового набора данных. Вместо оценки функции ошибки $E(\mathbf{w})$ иногда удобнее использовать среднеквадратичную ошибку (root-mean-square, RMS), определяемую как

$$E_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2}, \quad (1.3)$$

в которой деление на N позволяет сравнивать наборы данных разного размера на равных условиях, а извлечение квадратного корня гарантирует, что E_{RMS} измеряется по той же шкале (и в тех же единицах), что и целевая переменная t . Графики ошибок RMS обучающего и тестового наборов для различных значений M показаны на рис. 1.7. Ошибка тестового набора – это

показатель того, насколько хорошо удается предсказать значения t по новым наблюдениям данных x . Обратите внимание на рис. 1.7, что малые значения M дают относительно большие значения ошибки тестового набора, что можно объяснить тем фактом, что соответствующие полиномы относительно негибки и не способны уловить колебания в функции $\sin(2\pi x)$. Значения M в диапазоне $3 \leq M \leq 8$ дают небольшие значения ошибки тестового набора, и они также дают разумные представления образующей функции $\sin(2\pi x)$, как видно для $M = 3$ на рис. 1.6.

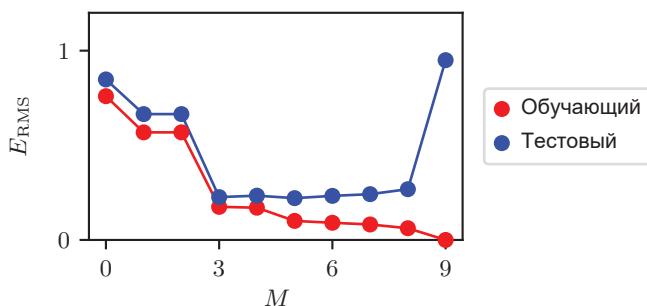


РИС. 1.7 Графики среднеквадратичной ошибки, определяемой по (1.3), оцененной на обучающем множестве и на независимом тестовом множестве, для различных значений M

Для $M = 9$ ошибка обучающего набора равна нулю, как и следовало ожидать, поскольку этот полином содержит 10 степеней свободы, соответствующих 10 коэффициентам w_0, \dots, w_9 , и, следовательно, может быть настроен точно на 10 точек данных в обучающем наборе. Однако ошибка тестового набора стала очень большой, и, как показано на рис. 1.6, соответствующая функция $y(x, w^*)$ демонстрирует значительные колебания.

Это может показаться парадоксальным, поскольку полином данного порядка содержит все полиномы более низкого порядка в качестве частных случаев. Поэтому полином $M = 9$ способен давать результаты, по крайней мере, не хуже, чем полином $M = 3$. Более того, можно предположить, что лучшим прогностическим средством для новых данных будет функция $\sin(2\pi x)$, на основе которой эти данные были получены (и позже будет показано, что это действительно так). Нам известно, что разложение в степенной ряд функции $\sin(2\pi x)$ включает в себя члены всех порядков, поэтому можно ожидать, что результаты будут стабильно улучшаться по мере увеличения M .

Некоторое представление об этой проблеме можно получить при изучении значений коэффициентов w^* , полученных из полиномов различных порядков, как показано в табл. 1.1. Как можно видеть, с увеличением M величина коэффициентов обычно становится больше. В частности, для полинома $M = 9$ коэффициенты уже достаточно точно подогнаны к данным. Они имеют большие положительные и отрицательные значения, так что соответствующая полиномиальная функция точно соответствует каждой из точек данных, но между точками данных (особенно вблизи концов диапазона) функция

демонстрирует большие колебания, наблюдаемые на рис. 1.6. Интуитивно понятно, что более гибкие полиномы с большими значениями M все больше подстраиваются под случайный шум на целевых значениях.

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0^*	0,11	0,90	0,12	0,26
w_1^*		-1,58	11,20	-66,13
w_2^*			-33,67	1 665,69
w_3^*			22,43	-15 566,61
w_4^*				76 321,23
w_5^*				-217 389,15
w_6^*				370 626,48
w_7^*				-372 051,47
w_8^*				202 540,70
w_9^*				-46 080,94

ТАБЛИЦА 1.1 Коэффициенты w^* для полиномов разного порядка. Типичная величина коэффициентов резко возрастает с увеличением порядка полинома

Более глубокое понимание этого явления можно получить в процессе изучения поведения обученной модели при изменении размера набора данных, как показано на рис. 1.8. Здесь видно, при заданной сложности модели проблема чрезмерной подгонки становится менее серьезной по мере увеличения размера набора данных. По-другому это можно выразить следующим образом: при большем наборе данных можно позволить подгонку под них более сложной (т. е. более гибкой) модели. Один из приблизительных эвристических подходов, который порой используется в классической статистике, заключается в том, что количество точек данных должно быть не меньше некоторого кратного (скажем, 5 или 10) числа обучаемых параметров в модели. Тем не менее, когда позже в этой книге будет рассматриваться глубокое обучение, будет показано, что отличные результаты могут быть получены при использовании моделей со значительно большим числом параметров, чем количество точек обучающих данных (см. раздел 9.3.2).

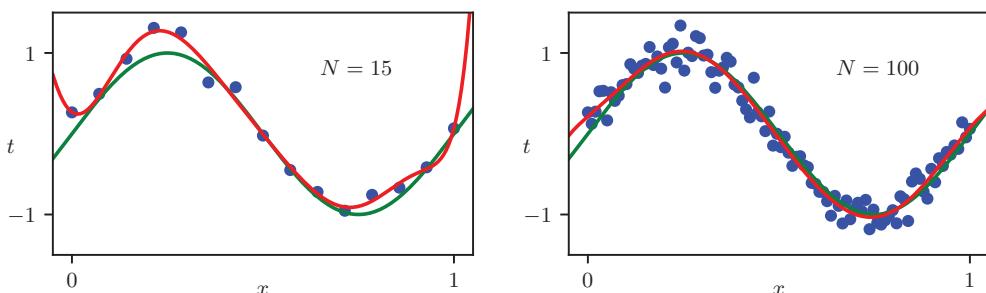


РИС. 1.8 Графики решений, полученных при минимизации функции ошибки по сумме квадратов (1.2) с помощью полинома $M = 9$ для $N = 15$ точек данных (левый график) и $N = 100$ точек данных (правый график). При увеличении размера набора данных проблема избыточной подгонки уменьшается

1.2.5. Регуляризация

Ограничение числа параметров в модели в зависимости от размера доступного обучающего набора вызывает чувство неудовлетворенности. Более разумным представляется выбор сложности модели в зависимости от сложности решаемой задачи. В качестве альтернативы ограничению числа параметров для решения проблемы избыточной подгонки нередко используется метод регуляризации (*regularization*), который заключается в добавлении штрафного слагаемого (*penalty term*) в функцию ошибки (1.2), чтобы не допустить появления коэффициентов с большими разбросами. Простейшее подобное штрафное слагаемое имеет вид суммы квадратов всех коэффициентов, что в результате приводит к изменению функции ошибки вида

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2, \quad (1.4)$$

где $\|\mathbf{w}\|^2 \equiv \mathbf{w}^T \mathbf{w} = w_0^2 + w_1^2 + \dots + w_M^2$, а коэффициент λ определяет относительную важность регуляризационного слагаемого по сравнению со слагаемым суммы квадратов ошибок. Заметим, что зачастую коэффициент w_0 опускается из регуляризатора, поскольку его включение приводит к зависимости результатов от выбора источника целевой переменной (Hastie, Tibshirani and Friedman, 2009), либо он может быть включен, но со своим собственным (см. раздел 9.2.1) коэффициентом регуляризации. Опять же, функция ошибки в (1.4) может быть минимизирована именно в замкнутой форме (см. упражнение 4.2). Подобные методы известны в литературе по статистике как *методы усадки (shrinkage methods)*, поскольку они позволяют сократить значение коэффициентов. В контексте нейронных сетей этот подход известен как *уменьшение весов (weight decay)*, так как параметры в нейронной сети называются весами, а такой регуляризатор способствует их уменьшению вплоть до нуля.

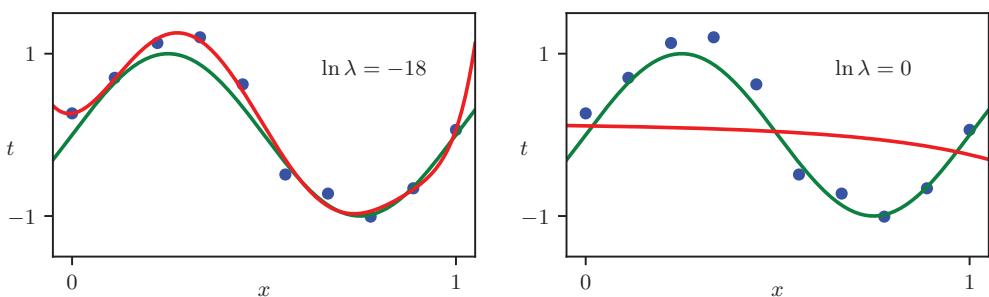


РИС. 1.9 Графики полиномов $M=9$, подогнанных под набор данных на рис. 1.4 с помощью регуляризованной функции ошибки (1.4) для двух значений параметра регуляризации λ , соответствующих $\ln \lambda = -18$ и $\ln \lambda = 0$. Случай отсутствия регуляризатора, т. е. $\lambda = 0$, соответствующий $\ln \lambda = -\infty$, показан в правом нижнем углу рис. 1.6

На рис. 1.9 показаны результаты подгонки полинома порядка $M = 9$ к тому же набору данных, что и раньше, но теперь с использованием регуляризованной функции ошибки, заданной в (1.4). Как видно, при значении $\ln \lambda = -18$ избыточную подгонку удалось пресечь, и теперь получилось гораздо более близкое представление базовой функции $\sin(2\pi x)$. Однако если использовать слишком большое значение λ , то можно вновь получить плохую подгонку, как показано на рис. 1.9 для $\ln \lambda = 0$. Соответствующие коэффициенты подогнанных полиномов приведены в табл. 1.2. Они свидетельствуют о положительном эффекте регуляризации, который заключается в уменьшении величины коэффициентов. Отметим, что $\ln \lambda = -\infty$ соответствует модели без регуляризации, т. е. графику в правом нижнем углу на рис. 1.6. Как видно, с увеличением значения λ величина типичного коэффициента становится меньше.

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0,26	0,26	0,11
w_1^*	-66,13	0,64	-0,07
w_2^*	1 665,69	43,68	-0,09
w_3^*	-15 566,61	-144,00	-0,07
w_4^*	76 321,23	57,90	-0,05
w_5^*	-217 389,15	117,36	-0,04
w_6^*	370 626,48	9,87	-0,02
w_7^*	-372 051,47	-90,02	-0,01
w_8^*	202 540,70	-70,90	-0,01
w_9^*	-46 080,94	75,26	0,00

ТАБЛИЦА 1.2 Коэффициенты w^* для $M = 9$ полиномов с различными значениями параметра регуляризации λ

Влияние регуляризации на ошибку обобщения можно проследить путем построения графика зависимости среднеквадратичной ошибки (1.3) для обучающего и тестового наборов от $\ln \lambda$, как показано на рис. 1.10. Как видно, λ теперь контролирует эффективную сложность модели и, следовательно, определяет степень избыточной подгонки.

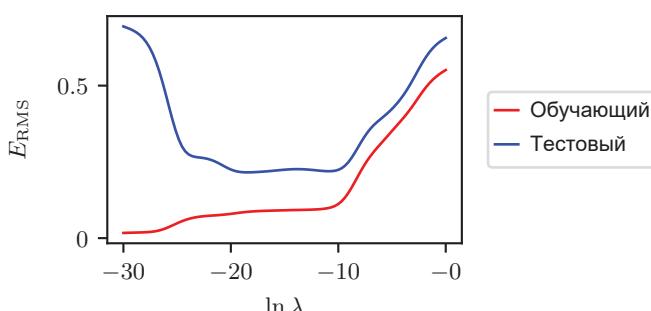


РИС. 1.10 График зависимости среднеквадратичной ошибки (1.3) от $\ln \lambda$ для полинома $M = 9$

1.2.6. Выбор модели

Величина λ служит примером *гиперпараметра* (*hyperparameter*), значения которого фиксируются при минимизации функции ошибки для определения параметров модели w . Нельзя просто определить значение λ посредством совместной минимизации функции ошибки по отношению к w и λ , поскольку это приведет к $\lambda \rightarrow 0$ и избыточной подгонке модели с малой или нулевой ошибкой обучения. Аналогичным образом порядок M полинома является гиперпараметром модели, и простая оптимизация ошибки обучающего набора по отношению к M приведет к большим значениям M и соответствующей чрезмерной подгонке. Поэтому необходимо найти подходящий способ определения гиперпараметров. Приведенные выше результаты предлагают простой способ достижения этой цели, а именно: взять имеющиеся данные и разделить их на обучающий набор для определения коэффициентов w и отдельный *валидационный* набор, также называемый *удержанным набором* (*hold-out set*) либо *набором для разработки* (*development set*). Затем необходимо выбрать модель с наименьшей ошибкой на валидационном наборе. Если модель разрабатывается многократно с использованием набора данных ограниченного размера, может произойти чрезмерная подгонка к валидационным данным, поэтому может потребоваться третий *тестовый набор* (*test set*), на котором в конечном итоге будет оценена эффективность выбранной модели.

Для некоторых приложений количество данных для обучения и тестирования может быть ограниченным. Для построения хорошей модели необходимо использовать как можно больше доступных данных для обучения. Однако если валидационный набор слишком мал, он даст достаточно зашумленную оценку эффективности предсказания. Одним из решений этой дилеммы является использование *перекрестной валидации* (*cross-validation*), как показано на рис. 1.11. Это позволяет использовать часть $(S - 1)/S$ имеющихся данных для обучения и при этом использовать все данные для оценки эффективности. Если данных особенно мало, целесообразно рассмотреть случай $S = N$, где N – общее количество точек данных, что позволяет использовать *технику перекрестной валидации по отдельным образцам* (*leave-one-out technique*). На рисунке техника S -кратной перекрестной валидации показана для случая $S = 4$ и включает получение имеющихся данных с разбиением их на S групп одинакового размера. Затем $S - 1$ групп используются для обучения набора моделей, которые далее оцениваются на оставшейся группе. После этого эта процедура повторяется для всех S возможных вариантов для оставшейся группы, обозначенных здесь красными блоками, и оценки производительности, полученные в результате S прогонов, усредняются.

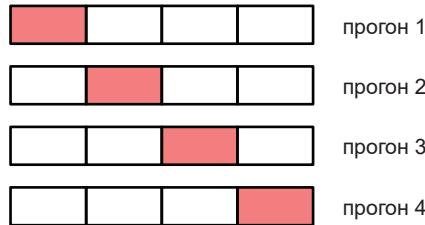


РИС. 1.11 Техника *S*-кратной перекрестной валидации для случая $S = 4$

Основной недостаток перекрестной валидации заключается в необходимости увеличения количества обучающих прогонов в S раз, что может оказаться проблематичным для моделей с высокими вычислительными затратами. Еще одна проблема методов, подобных перекрестной валидации, где для оценки эффективности используются отдельные данные, заключается в необходимости использования нескольких гиперпараметров сложности для одной модели (например, может быть несколько гиперпараметров регуляризации). Исследование комбинаций настроек таких гиперпараметров в худшем случае может вызвать необходимость в проведении множества этапов обучения, экспоненциально зависящих от количества гиперпараметров. Современный уровень развития машинного обучения предполагает использование очень больших моделей, обученных на таких же больших наборах данных. Соответственно, возможности для исследования настроек гиперпараметров ограничены, поэтому приходится полагаться на результаты тестирования небольших моделей и эвристику.

Этот простой пример подгонки полинома к синтетическому набору данных, полученных из синусоидальной функции, позволил продемонстрировать многие ключевые идеи машинного обучения, и в последующих главах этот пример будет использован неоднократно. Однако реальные задачи машинного обучения отличаются по некоторым важным параметрам. Размер наборов данных для обучения может быть на много порядков больше, и, как правило, в них гораздо больше входных переменных, возможно, исчисляемых миллионами, например для анализа изображений, а также множество выходных переменных. Обучаемая функция, связывающая выходы с входами, управляет классом моделей, известных как нейронные сети, и они могут иметь большое количество параметров, порой до сотен миллиардов, а функция ошибки будет в высшей степени нелинейной функцией этих параметров. Функция ошибки больше не может быть минимизирована с помощью закрытого решения, вместо этого ее необходимо минимизировать с помощью итерационных методов оптимизации на основе оценки производных функции ошибки относительно параметров, которые требуют специального вычислительного оборудования и значительных вычислительных ресурсов.

1.3. Краткая история машинного обучения

Машинное обучение имеет долгую и богатую историю с поиском разнообразных альтернативных концепций. Здесь мы сосредоточимся на эволюции методов машинного обучения на основе нейронных сетей, поскольку они представляют собой основу глубокого обучения и оказались наиболее эффективным подходом к машинному обучению для реальных приложений.

Изначально модели нейронных сетей разрабатывались на основе исследований обработки информации в мозге человека и других млекопитающих. Основными единицами обработки информации в мозге являются электрически активные клетки, называемые нейронами, как показано на рис. 1.12. При «срабатывании» нейрон посылает электрический импульс по аксону, где он достигает узлов под названием синапсы, которые формируют связи с другими нейронами. В синапсах выделяются химические сигналы, называемые нейротрансмиттерами, которые могут стимулировать или подавлять работу соседних нейронов.

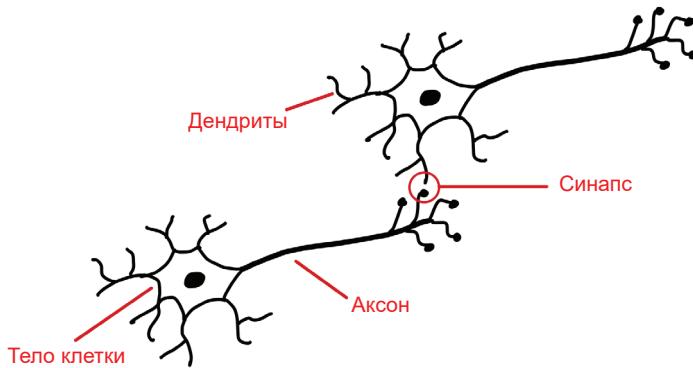


РИС. 1.12 Схематическое изображение двух нейронов человеческого мозга. Эти электрически активные клетки общаются через синапсы, интенсивность которых меняется по мере обучения сети

Человеческий мозг содержит в общей сложности порядка 90 млрд нейронов, каждый из которых имеет в среднем несколько тысяч синапсов с другими нейронами. Таким образом, сложная нейронная сеть насчитывает в общей сложности примерно 100 трлн (10^{14}) синапсов. Если конкретный нейрон получает достаточную стимуляцию от срабатывания других нейронов, он тоже может быть стимулирован к срабатыванию. Однако некоторые синапсы обладают отрицательным (или тормозным) эффектом, когда возбуждение входного нейрона снижает вероятность возбуждения выходного нейрона. Способность одного нейрона вызвать срабатывание другого зависит от силы синапса, и именно изменения в этой силе представляют собой ключевой механизм, с помощью которого мозг может хранить информацию и учиться на основе накопленного опыта.

Эти свойства нейронов были отражены в очень простых математических моделях, известных как *искусственные нейронные сети* (*artificial neural networks*), которые затем послужили основой для вычислительных методов обучения (McCulloch and Pitts, 1943). Многие из этих моделей описывают свойства одного нейрона путем формирования линейной комбинации выходов других нейронов, которая затем преобразуется с помощью нелинейной функции. Математически это можно выразить в виде

$$a = \sum_{i=1}^M w_i x_i, \quad (1.5)$$

$$y = f(a), \quad (1.6)$$

где x_1, \dots, x_M означают M входов, соответствующих активности других нейронов, поддерживающих связь с этим нейроном, а w_1, \dots, w_M – непрерывные переменные, называемые *весами* (*weights*), которые отражают силу связанных синапсов. Количественная величина a называется *предварительной активацией* (*pre-activation*), нелинейная функция $f(\cdot)$ называется *функцией активации* (*activation function*), а выход y называется *активацией* (*activation*). Как видно, полином (1.1) можно рассматривать в качестве частного случая такого представления, где входы x_i задаются степенями переменной x , а функция $f(\cdot)$ представляет собой тождество $f(a) = a$. Простая математическая формулировка, представленная в (1.5) и (1.6), является основой моделей нейронных сетей с 1960-х годов до наших дней и может быть представлена в виде диаграммы, как показано на рис. 1.13, где преобразования (1.5) и (1.6) описывают один нейрон. Полиномиальную функцию (1.1) можно рассматривать как частный случай этой модели.

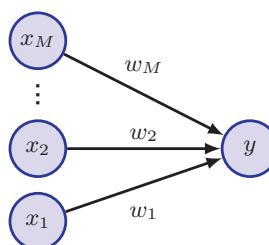


РИС. 1.13 Простая диаграмма нейронной сети, представляющая преобразования (1.5) и (1.6), описывающие один нейрон

1.3.1. Однослойные сети

Историю искусственных нейронных сетей можно условно разделить на три отдельных этапа в зависимости от уровня сложности сетей, который определяется количеством «слоев» обработки. Простую нейронную модель, описанную (1.5) и (1.6), можно рассматривать как сеть с одним слоем обработки,

соответствующим одному слою связей на рис. 1.13. Одной из наиболее важных моделей такого типа в истории нейронных вычислений является *перцептрон* (Rosenblatt, 1962), в котором функция активации $f(\cdot)$ представляет собой ступенчатую функцию вида

$$f(a) = \begin{cases} 0, & \text{если } a \leq 0, \\ 1, & \text{если } a > 0. \end{cases} \quad (1.7)$$

Это можно рассматривать как упрощенную модель возбуждения нейрона, в которой он возбуждается тогда и только тогда, когда суммарный взвешенный вход превышает порог 0. Перцептрон был впервые разработан Розенблаттом (Rosenblatt, 1962), который создал специальный алгоритм обучения с интересным свойством: если существует набор значений весов, для которых перцептрон может достичь идеальной классификации обучающих данных, то алгоритм гарантированно найдет решение за конечное число шагов (Bishop, 2006). Помимо алгоритма обучения, перцептрон также имел специальную аналоговую аппаратную реализацию, как показано на рис. 1.14. Типичная конфигурация перцептрана имеет несколько слоев обработки, но только один из этих слоев может обучаться на основе данных, поэтому перцептрон считается «однослоиной» нейронной сетью.

На фотографии слева показано, как входные данные были получены с помощью простой системы камер, где входная сцена – в данном случае это печатный символ – освещалась мощными лампами, а изображение фокусировалось на массиве фотоэлементов из сульфида кадмия размером 20×20 , давая примитивное 400-пиксельное изображение. Перцептрон также имел коммутационную плату, показанную на средней фотографии, которая позволяла опробовать различные конфигурации входных признаков. Зачастую они подключались в произвольном порядке для демонстрации способности перцептрана обучаться без необходимости точного подключения, в отличие от современного цифрового компьютера. На фотографии справа показана одна из стоек с обучаемыми весами. Каждый вес был реализован с помощью вращающегося переменного резистора (потенциометра), который приводился в движение электродвигателем, что позволяло алгоритму обучения автоматически регулировать значение веса.

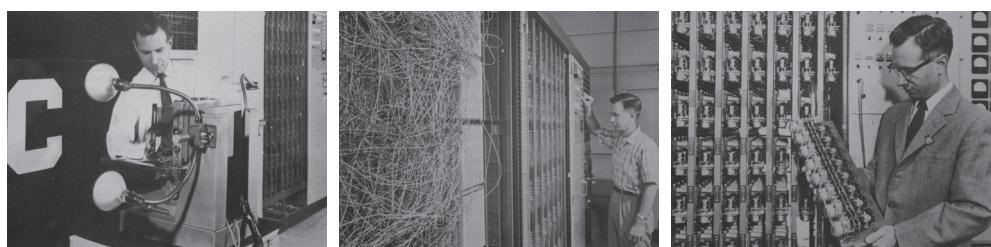


РИС. 1.14 Иллюстрация аппаратного обеспечения перцептрана *Mark 1*

Поначалу способность перцепtronов обучаться на данных подобно мозгу считалась просто поразительной. Однако вскоре стало очевидно, что у этой модели есть и серьезные ограничения. Свойства перцепtronов были проанализированы в работе (Minsky and Papert, 1969), где авторы определили формальное доказательство ограниченных возможностей однослойных сетей. Но они также предположили, что аналогичные ограничения будут распространяться и на сети с несколькими слоями обучаемых параметров. Хотя последняя гипотеза оказалась в корне неверной, ее следствием стало снижение энтузиазма в области использования моделей нейронных сетей, что способствовало падению интереса к нейронным сетям и отсутствию финансирования на протяжении 1970-х и начала 1980-х годов. Кроме того, исследователи не могли изучить свойства многослойных сетей из-за отсутствия эффективного алгоритма их обучения, поскольку такие методы, как алгоритм перцептрана, были характерны лишь для однослойных моделей. Обратите внимание, что, несмотря на исчезновение перцептранов из практического машинного обучения, их название по-прежнему актуально, поскольку современные нейронные сети также иногда называют *многослойными перцептранами* (*multilayer perceptron*, или *MLP*).

1.3.2. Обратное распространение

Решение проблемы обучения нейронных сетей с более чем одним слоем обучаемых параметров было найдено с помощью дифференциального исчисления и применения градиентных методов оптимизации. Важным изменением стала замена шаговой функции (1.7) на непрерывно дифференцируемые функции активации с ненулевым градиентом. Другим ключевым изменением стало введение дифференцируемых функций ошибки, которые определяют эффективность прогнозирования целевых переменных в обучающем множестве при заданном выборе значений параметров. Пример такой функции ошибки был рассмотрен при использовании функции ошибки суммы квадратов (1.2) для подгонки полиномов (см. раздел 1.2.3).

Благодаря этим изменениям появилась функция ошибки, производные которой можно оценивать по каждому из параметров сети. Теперь можно рассматривать сети с более чем одним слоем параметров. На рис. 1.15 показана простая сеть с двумя обучающими слоями. Узлы в среднем слое называются *скрытыми элементами* (*hidden units*), поскольку их значения не появляются в обучающем множестве, содержащем только значения для входов и выходов. Каждый из скрытых элементов и каждый из выходных элементов на рис. 1.15 вычисляет функцию в виде, заданном (1.5) и (1.6), где функция активации $f(\cdot)$ дифференцируема. Для заданного набора входных значений состояния всех скрытых и выходных элементов могут быть оценены путем повторного применения (1.5) и (1.6), при этом информация распространяется по сети в направлении стрелок. По этой причине такие модели иногда называют *нейронными сетями с прямой связью* (*feed-forward neural networks*).

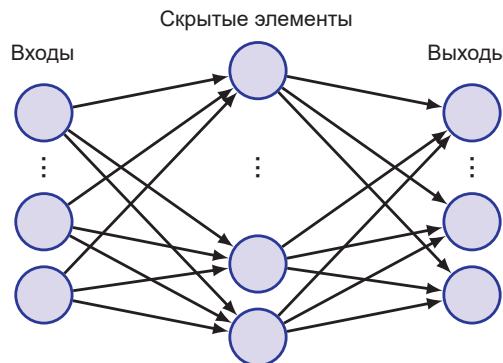


РИС. 1.15 Нейронная сеть с двумя слоями параметров, в которой стрелки обозначают направление потока информации по сети

Для обучения такой сети параметры сначала инициализируются с помощью генератора случайных чисел, а затем итеративно обновляются с помощью методов градиентной оптимизации. Это предполагает оценку производных функции ошибки, что можно эффективно сделать при помощи процесса, известного как *обратное распространение ошибки* (*error backpropagation*, см. главу 8). В процессе обратного распространения информация течет по сети в обратном направлении от выходов к входам (Rumelhart, Hinton and Williams, 1986). Существует множество различных алгоритмов оптимизации с использованием градиентов оптимизируемой функции, но наиболее распространенный в машинном обучении является также самым простым и известен как *стохастический градиентный спуск* (*stochastic gradient descent*, см. главу 7).

Возможность обучения нейронных сетей с несколькими слоями весов стала прорывом, который привел к возрождению интереса к этой области начиная приблизительно с середины 1980-х годов. Это был период, когда эта область вышла за рамки нейробиологического интереса и обрела более строгую и принципиальную основу (Bishop, 1995b). В частности, было признано, что важнейшую роль при разработке нейронных сетей и машинном обучении играет теория вероятностей и принципы из области статистики. Один из ключевых моментов заключается в том, что обучение на основе данных включает в себя некоторые исходные предположения, иногда называемые *предварительными знаниями* (*prior knowledge*) или *индуктивными смещениями* (*inductive biases*). Они могут быть заложены в явном виде, например при проектировании структуры нейронной сети, чтобы классификация поражения кожи не зависела от местоположения поражения на изображении, или же они могут принимать форму неявных предположений, возникающих из математической формы модели или способа ее обучения.

Использование обратного распространения и градиентной оптимизации значительно расширило возможности нейронных сетей при решении практических задач. Однако при этом было также установлено, что в сетях с большим количеством слоев полезные значения могут быть получены только

от весов в двух последних слоях. Лишь немногие исключения, в частности модели, используемые для анализа изображений и называемые *сверточными* (конволюционными) нейронными сетями (*convolutional neural networks*, LeCun et al., 1998), не нашли успешного применения в сетях с более чем двумя слоями (см. главу 10). Это опять же ограничивало сложность решаемых задач при использовании таких сетей. Чтобы добиться приемлемой производительности для многих приложений, приходилось использовать ручную *предварительную обработку* для преобразования входных переменных в некоторое новое пространство, где, как надеялись, задача машинного обучения будет решаться легче. Этот этап предварительной обработки иногда называют *извлечением признаков* (*feature extraction*). Несмотря на эффективность такого подхода, было бы гораздо лучше, если бы признаки можно было узнать из данных, а не создавать их вручную.

К началу нового тысячелетия существующие методы построения нейронных сетей вновь достигли предела своих возможностей. Исследователи приступили к изучению ряда альтернатив нейронным сетям, таких как ядерные методы, машины опорных векторов, гауссовые процессы и многие другие. Нейронные сети снова оказались в немилости, хотя ядро исследователей-энтузиастов по-прежнему занималось поиском действительно эффективного подхода к обучению сетей с большим количеством слоев.

1.3.3. Глубокие сети

Третий (нынешний) этап развития нейронных сетей начался во втором десятилетии XXI века. Ряд новых достижений дал возможность эффективно обучать нейронные сети с большим количеством слоев с весами, тем самым сняв прежние ограничения на возможности этих методик. Сети со многими слоями весов называют *глубокими нейронными сетями* (*deep neural networks*), а подобласть машинного обучения, которая фокусируется на таких сетях, называется *глубоким обучением* (*deep learning*, LeCun, Bengio and Hinton, 2015).

Одной из важнейших тем на пути становления глубокого обучения было значительное увеличение масштаба нейронных сетей, определяемого по количеству параметров. Если в 1980-х годах широко распространенными были сети с несколькими сотнями или тысячами параметров, то в дальнейшем их число неуклонно повышалось до миллионов, а затем и миллиардов, в то время как современные модели могут насчитывать порядка одного триллиона (10^{12}) параметров. Сети с большим количеством параметров требуют соразмерно больших наборов данных для получения хороших значений этих параметров с помощью обучающих сигналов. Сочетание массивных моделей и массивных наборов данных, в свою очередь, требует масштабных вычислений при обучении модели. Специализированные процессоры, называемые *графическими процессорами* (*graphics processing units*, или GPU), изначально созданные для очень быстрого рендеринга графических данных в таких приложениях, как видеоигры, прекрасно подошли для обучения нейронных сетей, поскольку функции, вычисляемые блоками одного слоя сети, могут оце-

ниваться параллельно, а это хорошо согласуется с массивным параллелизмом GPU (Krizhevsky, Sutskever and Hinton, 2012). Сегодня обучение самых больших моделей выполняется на больших массивах из тысяч графических процессоров, связанных специализированными высокоскоростными соединениями.

На рис. 1.16 показан рост числа вычислительных циклов, необходимых для обучения современной нейронной сети, с течением времени, при этом наблюдаются две различные фазы роста. Вертикальная ось имеет экспоненциальную шкалу и единицы измерения – петафлоп/с в день (petaflop/s-days), где петафлоп – это 10^{15} (тысяча триллионов) операций с плавающей запятой, а петафлоп/с – это один петафлоп в секунду. Один петафлоп/с в день представляет собой вычисления со скоростью петафлоп/с в течение 24 часов, что примерно равно 10^{20} операциям с плавающей запятой, и, следовательно, верхняя линия графика представляет собой впечатляющие 10^{24} операции с плавающей запятой. Прямая линия на графике отображает экспоненциальный рост, и мы видим, что со временем перцептрана и до 2012 года время удвоения скорости составляло около 2 лет, что соответствует общему росту вычислительной мощности по закону Мура. Начиная с 2012 года, когда наступила эра глубокого обучения, мы снова видим экспоненциальный рост, но время удвоения теперь составляет 3,4 месяца, что соответствует увеличению вычислительной мощности в 10 раз ежегодно!

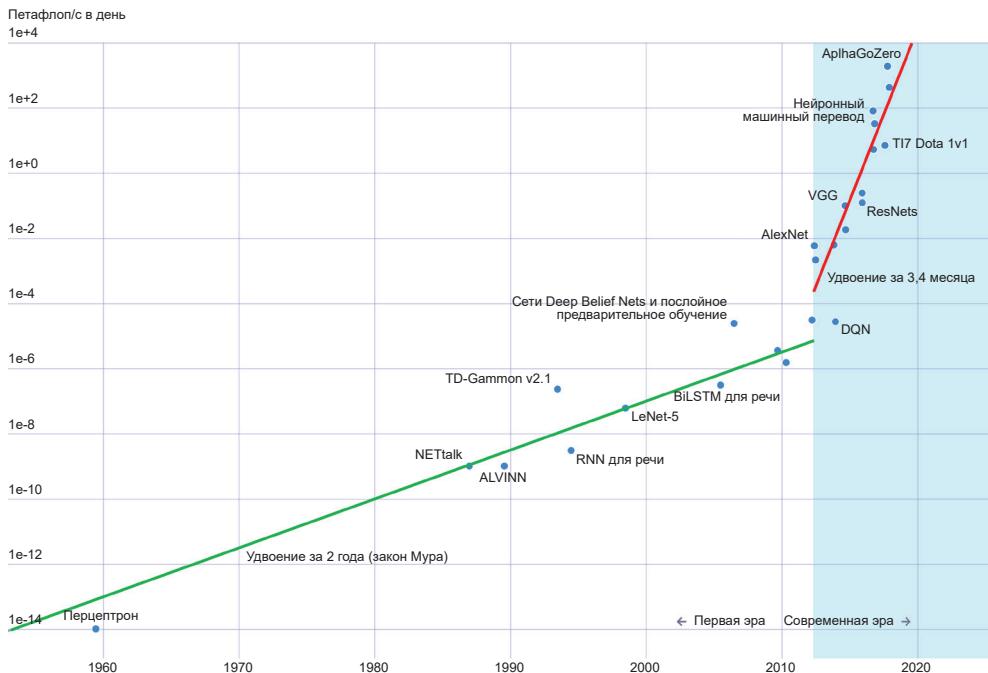


РИС. 1.16 Зависимость количества вычислительных циклов, измеряемых в петафлоп/с в день, необходимых для обучения современной нейронной сети, от даты. [С разрешения OpenAI]

Нередко оказывается, что улучшение производительности за счет инноваций в архитектуре или включения более сложных форм индуктивного смещения очень быстро сводится на нет простым увеличением количества обучающих данных, а также соразмерным увеличением размера модели и соответствующей вычислительной мощности, используемой для обучения (Sutton, 2019). Большие модели могут не только демонстрировать лучшую производительность при решении конкретной задачи, но также быть способными решать более широкий спектр различных задач с помощью одной и той же обученной нейронной сети. Ярким примером этого являются большие языковые модели, поскольку отдельная сеть не только обладает необычайно широкими возможностями, но также способна превзойти специализированные сети, созданные для решения конкретных задач (см. раздел 12.3.5).

Выше было показано, что глубина играет важную роль в достижении нейронными сетями высокой производительности. Один из способов оценить роль скрытых слоев в глубокой нейронной сети – это *обучение представлениям* (*representation learning*, Bengio, Courville and Vincent, 2012), в ходе которого сеть учится преобразовывать входные данные в новое представление с семантическим смыслом, создавая тем самым гораздо более простую задачу для конечного слоя или слоев. Такие внутренние представления могут быть использованы для решения смежных задач путем трансферного обучения, как это было показано в случае классификации кожных новообразований. Интересно отметить, что нейронные сети для обработки изображений могут обучаться внутренним представлениям, которые удивительно похожи на те, что наблюдаются в зрительной коре млекопитающих. Большие нейронные сети, которые можно адаптировать или тонко настраивать для решения различных задач, называются *базовыми моделями* (*foundation models*). Эти модели могут использовать преимущества больших разнородных наборов данных для создания моделей самого широкого применения (Bommasani et al., 2021, см. раздел 10.3).

Помимо возможности масштабирования, в успехе глубокого обучения сыграли роль и другие факторы. Например, в простых нейронных сетях обучающие сигналы становятся слабее по мере их обратного распространения через последовательные слои глубокой сети (см. раздел 9.5). Одним из методов решения этой проблемы является введение *остаточных соединений* (*residual connections*, He et al., 2015a), которые облегчают обучение сетей с сотнями слоев. Другой ключевой разработкой стало внедрение методов *автоматического дифференцирования* (*automatic differentiation*), при которых код, обеспечивающий обратное распространение для оценки градиентов функции ошибки, генерируется автоматически из кода, используемого для задания прямого распространения. Это позволяет разработчикам оперативно экспериментировать с различными архитектурами нейронных сетей и быстро комбинировать различные архитектурные элементы, поскольку в явном виде нужно кодировать только относительно простые функции прямого распространения. Кроме того, большая часть исследований в области машинного обучения проводится с открытым исходным кодом, что позволяет исследователям опираться на работы других, тем самым еще больше ускоряя темпы прогресса в этой области.

Глава 2

Вероятности

Практически в каждом случае применения машинного обучения приходится иметь дело с неопределенностью. Например, система классификации изображений новообразований на коже, доброкачественных или злокачественных, на практике никогда не сможет достичь идеальной точности. Можно выделить два вида неопределенности. К первому относится *эпистемическая неопределенность* (*epistemic uncertainty*, происходит от греческого слова *episteme*, означающего «знание»), которую иногда называют *систематической неопределенностью* (*systematic uncertainty*). Она возникает вследствие ограниченности наборов данных. По мере накопления данных, например большего количества примеров изображений доброкачественных и злокачественных новообразований, появляется возможность более точно предсказать класс нового случая. Однако даже при бесконечно большом наборе данных невозможно достичь идеальной точности по причине второго вида неопределенности, известного как алеаторная, или случайная неопределенность (*aleatoric uncertainty*), называемая также *собственная* (*intrinsic*) или *стохастическая неопределенность* (*stochastic uncertainty*), а иногда – просто шум (*noise*). Как правило, шум возникает из-за неполной информации о проходящем событии, и поэтому один из способов уменьшить этот источник неопределенности заключается в сборе различных видов данных. На рис. 2.1 (см. раздел 1.2) это проиллюстрировано с помощью расширения примера с синусоидальной кривой до двух измерений. На фрагменте (а) представлен график функции $y(x_1, x_2) = \sin(2\pi x_1)\sin(2\pi x_2)$. Данные получены путем выборки значений x_1 и x_2 , вычисления соответствующего значения $y(x_1, x_2)$, а затем добавления гауссова шума. Фрагмент (б) отражает график из 100 точек данных, где x_2 являются ненаблюдаемыми, и показывает высокий уровень шума. Фрагмент (с) отражает график из 100 точек данных, где x_2 зафиксированы на значении $x_2 = \pi/2$, что имитирует эффект возможности измерения x_2 так же, как и x_1 , и показывает гораздо более низкий уровень шума.

В качестве практического примера можно привести биопсию образца кожного новообразования, которая гораздо более информативна, чем просто изображение, и может значительно повысить точность определения наличия злокачественного новообразования. При условии анализа изображения и данных биопсии собственная неопределенность может быть очень мала, поэтому, собрав большой набор обучающих данных, можно снизить систематическую неопределенность.

тическую неопределенность до низкого уровня и, таким образом, получить прогноз класса новообразования с высокой точностью.

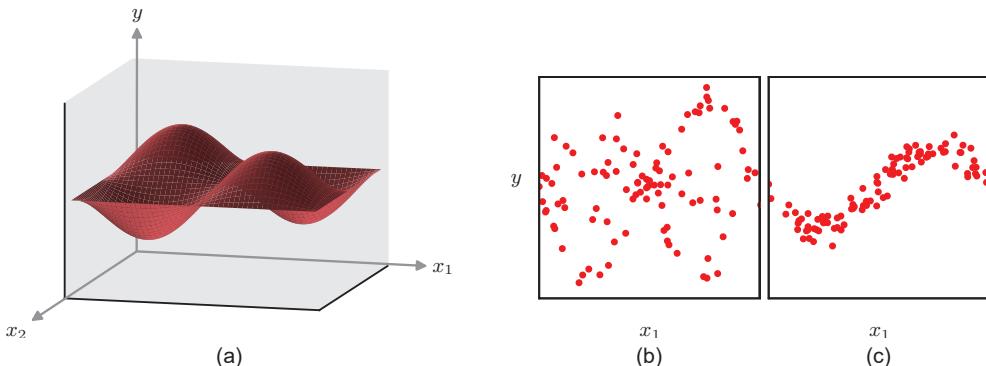


РИС. 2.1 Расширение простой задачи регрессии синусоидальной кривой на два измерения

Оба вида неопределенности могут быть обработаны с помощью *теории вероятностей*, которая обеспечивает последовательную парадигму для количественной оценки и обработки неопределенности, и тем самым служит одной из фундаментальных основ машинного обучения (см. раздел 2.1). В дальнейшем будет показано, что вероятности управляются двумя простыми формулами, известными как *правило суммы* (*sum rule*) и *правило произведения* (*product rule*) (см. раздел 5.2). В сочетании с *теорией принятия решений* (*decision theory*) эти правила позволяют (по крайней мере, в принципе) делать оптимальные прогнозы с учетом всей доступной информации, даже если эта информация бывает неполной или неоднозначной.

Концепция вероятности обычно описывается в терминах частоты повторяющихся событий. Рассмотрим, например, изогнутую монету, показанную на рис. 2.2, и предположим, что форма монеты такова, что, если подбрасывать ее большое количество раз, она в 60 % случаев приземляется вогнутой стороной вверх и, следовательно, в 40 % случаев – выпуклой стороной вверх. Можно сказать, что вероятность приземления монеты вогнутой стороной вверх равна 60 %, или 0,6. Строго говоря, вероятность определяется в пределе бесконечного числа «попыток», или в данном случае подбрасываний монеты.



РИС. 2.2 Вероятность можно рассматривать либо как частоту повторяющихся событий, либо как количественную оценку неопределенности

Поскольку монета всегда приземляется либо вогнутой, либо выпуклой стороной вверх, эти *вероятности* (*probability*) складываются в 100 %, или 1,0. Это определение вероятности в плане частоты повторяющихся событий является основой для *частотного* (*frequentist*) подхода в статистике.

Допустим, что, хотя и известно, что вероятность приземления монеты вогнутой стороной вверх равна 0,6, нам запрещено рассматривать саму монету, и нет возможности узнать, какая сторона – орел, а какая – решка. Если нам предложат сделать ставку на вероятность выпадения орла или решки при подбрасывании монеты, то, согласно принципу симметрии, ставка должна основываться на предположении, что вероятность выпадения орла равна 0,5, и, действительно, более тщательный анализ показывает, что в отсутствие какой-либо дополнительной информации это действительно рациональный выбор. В данном случае вероятности используются в более общем смысле, чем просто частота событий. Выпуклая сторона монеты – орел или решка, сама по себе не является повторяющимся событием, она попросту неизвестна. Использование вероятности в качестве количественной оценки неопределенности – это байесовская (*Bayesian*) концепция, которая носит более общий характер, поскольку включает в себя частотную вероятность как частный случай (см. раздел 2.6). Можно определить сторону монеты, которая является орлом, при условии предоставления результатов последовательности подбрасываний монеты с использованием байесовских логических рассуждений (см. упражнение 2.40). Чем больше таких результатов будет получено, тем меньше будет неопределенность в вопросе о сторонах монеты.

После неформального введения концепции вероятности перейдем к более детальному изучению вероятностей и обсудим их использование в количественных оценках. Концепции, изложенные в оставшейся части этой главы, станут основой для множества тем, обсуждаемых на протяжении всей книги.

2.1. Правила вероятности

В этом разделе будут выведены два простых правила, которые определяют особенности изменения вероятностей. При этом, несмотря на очевидную простоту, эти правила могут быть весьма эффективными и широко используемыми. Чтобы объяснить правила вероятности, сначала приведем простой пример.

2.1.1. Пример медицинского обследования

Рассмотрим проблему диагностики населения с целью выявления онкологических заболеваний на ранней стадии. Предположим, что 1 % населения действительно болен раком. В идеале наш тест на онкологию должен давать положительный результат для всех, у кого он есть, и отрицательный – для тех, у кого его нет. Однако тесты не идеальны, поэтому предположим, что при тестировании людей без онкологических заболеваний 3 % из них покажут

положительный результат. Это так называемые *ложноположительные результаты (false positives)*. Аналогичным образом, если проводить тестирование людей с онкологическими заболеваниями, 10 % из них дадут отрицательный результат. Это так называемые *ложноотрицательные результаты (false negatives)*. Различные коэффициенты ошибок приведены на рис. 2.3.

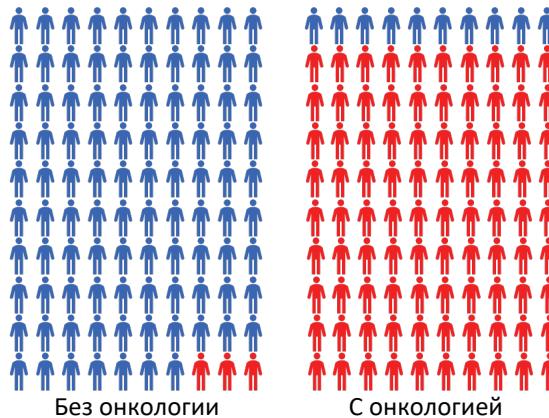


РИС. 2.3 Иллюстрация точности тестирования на онкологические заболевания

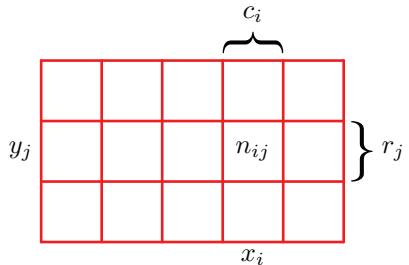
С учетом этой информации можно задаться следующими вопросами: (1) «Какова вероятность получения положительного результата при обследовании населения?», (2) «Какова вероятность обнаружения онкологического заболевания при положительном результате теста?». На эти вопросы можно было бы ответить при детальном анализе случаев диагностики рака. Однако вместо этого сделаем паузу в обсуждении этого конкретного примера и для начала выведем общие правила вероятности, известные как *правило суммы вероятностей (sum rule of probability)* и *правило произведения (product rule)*. Затем на основе этих правил дадим ответы на два наших вопроса.

2.1.2. Правила суммы и произведения

Чтобы вывести правила вероятности, рассмотрим несколько более общий пример, показанный на рис. 2.4, с двумя переменными X и Y . В нашем примере с онкологией X представляет собой наличие или отсутствие рака, а Y может быть переменной, обозначающей результат тестирования. Поскольку значения этих переменных могут меняться от одного человека к другому неизвестным образом, их называют *случайными переменными (random variables)*, или *стохастическими переменными (stochastic variables)*. Будем считать, что X может принимать любое из значений x_i , где $i = 1, \dots, L$, и что Y может принимать значения y_j , где $j = 1, \dots, M$. Рассмотрим в общей сложности N тестов, в которых производится выборка по обеим переменным X и Y , и пусть число таких тестов, в которых $X = x_i$ и $Y = y_j$, будет n_{ij} . Также пусть число исследова-

ний, в которых X принимает значение x_i (независимо от того, какое значение принимает Y), обозначается c_i , и аналогично пусть число исследований, в которых Y принимает значение y_j , обозначается r_j . В примере на рис. 2.4 $L = 5$ и $M = 3$. Если взять общее количество N случаев этих переменных, то число примеров, где $X = x_i$ и $Y = y_j$, можно обозначить через n_{ij} , что является количеством примеров в соответствующей ячейке массива. Количество примеров в столбце i , соответствующих $X = x_i$, обозначается c_i , а количество примеров в строке j , соответствующих $Y = y_j$, обозначается r_j .

РИС. 2.4 Правила суммы и произведения вероятностей на примере случайных переменных X и Y



Вероятность того, что X примет значение x_i , а Y примет значение y_j , записывается как $p(X = x_i, Y = y_j)$ и называется совместной (joint) вероятностью $X = x_i$ и $Y = y_j$. Она определяется числом точек, попадающих в ячейку i, j , как доля от общего числа точек, и, следовательно,

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}. \quad (2.1)$$

Здесь в неявной форме рассматривается предел $N \rightarrow \infty$. Аналогично вероятность того, что X принимает значение x_i независимо от значения Y , записывается как $p(X = x_i)$ и определяется долей от общего числа точек, приходящихся на столбец i , так что

$$p(X = x_i) = \frac{c_i}{N}. \quad (2.2)$$

Поскольку $\sum_i c_i = N$, можно видеть, что

$$\sum_{i=1}^L p(X = x_i) = 1 \quad (2.3)$$

и, следовательно, вероятности в сумме равны единице, как и должно быть. Поскольку количество случаев в столбце i на рис. 2.4 равно сумме количества случаев в каждой ячейке этого столбца, имеем $c_i = \sum_j n_{ij}$, и, следовательно, из (2.1) и (2.2) следует

$$p(X = x_i) = \sum_{j=1}^M p(X = x_i, Y = y_j), \quad (2.4)$$

что является правилом суммы вероятностей (sum rule of probability). Обратите внимание, что $p(X = x_i)$ иногда называют *безусловной*, или *предельной* (marginal), вероятностью, которая получается путем маргинализации, т. е. суммирования других переменных (в нашем случае Y).

Если рассматривать только те случаи, для которых $X = x_i$, доля таких случаев, для которых $Y = y_j$, записывается как $p(Y = y_j | X = x_i)$ и называется *условной* (conditional) вероятностью $Y = y_j$ при $X = x_i$. Она получается путем нахождения доли тех точек в столбце i , которые попадают в ячейки i, j , и, следовательно, определяется следующим образом:

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i} \quad (2.5)$$

Суммируя обе стороны по j и используя $\sum_i n_{ij} = c_i$, получаем

$$\sum_{j=1}^M p(Y = y_j | X = x_i) = 1, \quad (2.6)$$

что свидетельствует о правильной нормализации условных вероятностей. Из (2.1), (2.2) и (2.5) можно вывести следующее соотношение:

$$\begin{aligned} p(X = x_i | Y = y_j) &= \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N} \\ &= p(Y = y_j | X = x_i)p(X = x_i), \end{aligned} \quad (2.7)$$

что является правилом произведения вероятностей (product rule).

До сих пор здесь тщательно разграничивались случайная переменная, например X , и значения, которые она может принимать, например x_i . Таким образом, вероятность того, что X принимает значение x_i , обозначается $p(X = x_i)$. Хотя это и помогает избежать двусмысленности, но в итоге получается довольно громоздкая форма записи, а во многих случаях в такой детализации нет никакой необходимости. Вместо этого можно просто написать $p(X)$, чтобы обозначить распределение по случайной переменной X , или $p(x_i)$, чтобы обозначить распределение, оцененное для конкретного значения x_i , при условии, что интерпретация ясна из контекста.

Используя эти более компактные обозначения, можно записать два фундаментальных правила теории вероятностей в следующей форме:

$$\text{правило суммы } p(X) = \sum_Y p(X, Y), \quad (2.8)$$

$$\text{правило произведения } p(X, Y) = p(Y | X)p(X). \quad (2.9)$$

Здесь $p(X, Y)$ – это совместная вероятность, которая выражается как «вероятность X и Y ». Аналогично величина $p(Y | X)$ является условной вероятностью и выражается как «вероятность Y при условии X ». Наконец, величина $p(X)$ – это предельная вероятность, которая означает просто «вероятность X ». Эти два

простых правила лежат в основе всех вероятностных механизмов, которые будут использоваться в этой книге.

2.1.3. Теорема Байеса

Из правила произведения, а также свойства симметрии $p(X, Y) = p(Y, X)$ вытекает следующее соотношение между условными вероятностями:

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}. \quad (2.10)$$

Это соотношение называется *теоремой Байеса (Bayes' theorem)*. Эта теорема играет важную роль в машинном обучении. Обратите внимание и на то, как теорема Байеса связывает условное распределение $p(Y|X)$ в левой части уравнения с «обратным» условным распределением $p(X|Y)$ в правой части. С помощью правила суммы знаменатель в теореме Байеса можно выразить в виде величин, фигурирующих в числителе:

$$p(X) = \sum_Y p(X|Y)p(Y). \quad (2.11)$$

Таким образом, знаменатель в теореме Байеса можно рассматривать как константу нормализации, необходимую для того, чтобы сумма условного распределения вероятностей в левой части (2.10) по всем значениям Y равнялась единице.

На рис. 2.5 представлен простой пример с совместным распределением по двум переменным, который наглядно иллюстрирует концепцию маргинальных и условных распределений. Здесь из совместного распределения берется конечная выборка из $N = 60$ точек данных, которая показана в левом верхнем углу. В правом верхнем углу приведена гистограмма долей точек данных, имеющих каждое из двух значений Y . Из определения вероятности следует, что эти доли будут равны соответствующим вероятностям $p(Y)$ в пределе, когда размер выборки $N \rightarrow \infty$. Поэтому эту гистограмму можно рассматривать как простой способ моделирования распределения вероятностей при наличии лишь конечного числа точек, взятых из этого распределения (см. раздел 3.5.1). Оставшиеся два графика на рис. 2.5 показывают соответствующие оценкам гистограммы значения $p(X)$ и $p(X|Y=1)$.

2.1.4. Повторное медицинское обследование

Вернемся к нашему примеру с обследованием на наличие онкологических заболеваний и применим правила суммы и произведения вероятностей для получения ответов на наши два вопроса. Для ясности при работе над этим примером вновь будем четко различать случайные величины и их вариации. Наличие или отсутствие онкологического заболевания обозначим переменной C , которая может принимать два значения: $C = 0$ соответствует «нет рака»

и $C = 1$ соответствует «рак». Предположим, что один человек из ста в общей группе населения болен раком, поэтому имеем

$$p(C = 1) = 1/100, \quad (2.12)$$

$$p(C = 0) = 99/100, \quad (2.13)$$

соответственно. Обратите внимание, что они удовлетворяют условию $p(C = 0) + p(C = 1) = 1$.

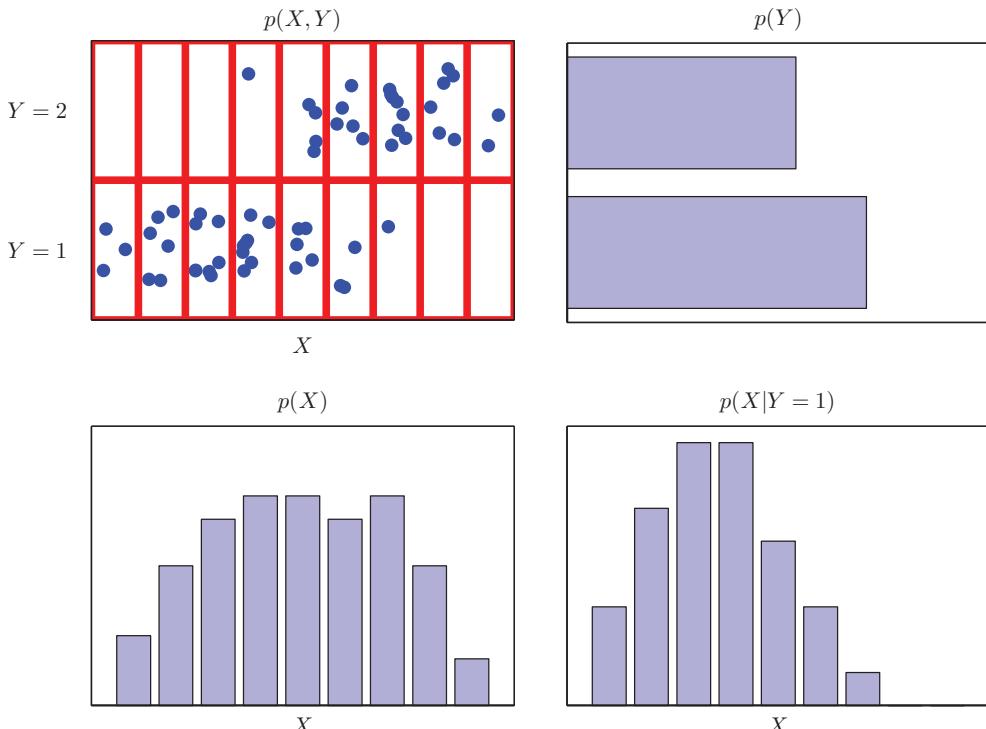


РИС. 2.5 Иллюстрация распределения по двум переменным: X , которая принимает девять возможных значений, и Y , которая принимает два возможных значения

Теперь введем вторую случайную переменную T , представляющую результат прохождения диагностического тестирования, где $T = 1$ обозначает положительный результат, свидетельствующий о наличии рака, а $T = 0$ представляет отрицательный результат, свидетельствующий об отсутствии рака. Как показано на рис. 2.3, нам известно, что для людей с онкологическими заболеваниями вероятность положительного результата теста составляет 90 %, а для тех, у кого рака нет, вероятность положительного результата теста равна 3 %. Теперь можно выразить все четыре условные вероятности:

$$p(T = 1 | C = 1) = 90/100, \quad (2.14)$$

$$p(T = 0 | C = 1) = 10/100, \quad (2.15)$$

$$p(T = 1 | C = 0) = 3/100, \quad (2.16)$$

$$p(T = 0 | C = 0) = 97/100. \quad (2.17)$$

Еще раз отметим, что эти вероятности нормализованы так, что

$$p(T = 1 | C = 1) + p(T = 0 | C = 1) = 1 \quad (2.18)$$

и, соответственно,

$$p(T = 1 | C = 0) + p(T = 0 | C = 0) = 1. \quad (2.19)$$

Теперь с помощью правил суммы и произведения вероятностей можно ответить на наш первый вопрос и оценить общую вероятность положительного результата теста у каждого случайного пациента:

$$\begin{aligned} p(T = 1) &= p(T = 1 | C = 0)p(C = 0) + p(T = 1 | C = 1)p(C = 1) \\ &= \frac{3}{100} \times \frac{99}{100} + \frac{90}{100} \times \frac{1}{100} = \frac{387}{10\,000} = 0,0387. \end{aligned} \quad (2.20)$$

Как видно, при случайном тестировании вероятность положительного результата составляет примерно 4 %, даже если вероятность того, что у человека действительно есть рак, составляет 1 %. Отсюда, используя правило суммы, следует, что $p(T = 0) = 1 - 387/10\,000 = 9613/10\,000 = 0,9613$ и, следовательно, вероятность того, что у человека нет рака, составляет примерно 96 %.

Теперь рассмотрим наш второй вопрос, который особенно интересует обследуемых людей: если тест положительный, какова вероятность того, что у этого человека действительно есть онкологическое заболевание? Для этого необходимо оценить вероятность рака при условии исхода теста, тогда как вероятности в (2.14)–(2.17) дают распределение вероятностей исхода теста при условии, что у человека есть рак. Мы можем решить проблему обратного изменения условной вероятности с помощью теоремы Байеса (2.10), чтобы получить

$$p(C = 1 | T = 1) = \frac{p(T = 1 | C = 1)p(C = 1)}{p(T = 1)} \quad (2.21)$$

$$= \frac{90}{100} \times \frac{1}{100} \times \frac{10\,000}{387} = \frac{90}{387} \approx 0,23. \quad (2.22)$$

Поэтому, если случайным образом протестируют какого-либо человека и тест окажется положительным, вероятность обнаружить у него онкологическое заболевание составляет 23 %. Из правила суммы следует, что $p(C = 0 | T = 1) = 1 - 90/387 = 297/387 \approx 0,77$, т. е. вероятность отсутствия рака составляет 77 %.

2.1.5. Априорные и апостериорные вероятности

На примере диагностики онкологических заболеваний можно построить важную интерпретацию теоремы Байеса. Если бы до прохождения обследования нас спросили, есть ли у человека вероятность заболеть раком, то наиболее полная доступная нам информация была бы представлена вероятностью $p(C)$. Это называют *априорной вероятностью* (*prior probability*), потому что она доступна до получения результатов обследования. Как только поступит информация о том, что этот человек получил положительный результат обследования, можно использовать теорему Байеса для вычисления вероятности $p(C|T)$, которую называют *апостериорной вероятностью* (*posterior probability*), поскольку она определяется после получения результата теста T .

В этом примере априорная вероятность обнаружения онкологического заболевания равна 1 %. Однако после положительного результата тестирования вероятность наличия рака увеличивается до 23 %, т. е., как мы интуитивно ожидаем, значительно возрастает. Однако следует отметить, что вероятность того, что человек с положительным результатом теста действительно болен раком, составляет всего 23 %, несмотря на то что тест, судя по рис. 2.3, является достаточно «точным» (см. упражнение 2.1). Этот вывод многим кажется нелогичным. Причина кроется в низкой априорной вероятности наличия онкологического заболевания. Хотя тест дает убедительные доказательства наличия рака, их необходимо объединить с априорной вероятностью с помощью теоремы Байеса для получения правильной апостериорной вероятности.

2.1.6. Независимые переменные

Наконец, если совместное распределение двух переменных сводится к произведению предельных значений, т. е. $p(X, Y) = p(X)p(Y)$, то говорят, что X и Y *независимы* (*independent*). Примером независимых событий могут служить последовательные подбрасывания монеты. Из правила произведения следует, что $p(Y|X) = p(Y)$, и поэтому условное распределение Y по X действительно не зависит от значения X . В нашем примере с обследованием на онкологию если вероятность положительного теста не зависит от наличия у человека рака, то $p(T|C) = p(T)$, а значит, из теоремы Байеса (2.10) следует $p(C|T) = p(C)$, и поэтому вероятность рака не меняется от просмотра результатов обследования. Разумеется, такое обследование будет бесполезным, поскольку результат тестирования ничего не скажет нам о том, болен ли человек раком.

2.2. Плотность распределения вероятностей

Помимо анализа вероятностей, определяемых на дискретных наборах значений, необходимо также принимать во внимание вероятности в отношении непрерывных переменных. Например, когда требуется определить дозировку лекарства для пациента. Поскольку в этом прогнозировании будет

присутствовать неопределенность, ее необходимо выразить количественно, и в этом случае также можно использовать вероятности. Однако невозможно просто напрямую применить рассмотренные до этого концепции вероятности, поскольку вероятность наблюдения конкретного значения непрерывной переменной с бесконечной точностью будет равна нулю. Вместо этого потребуется ввести в употребление концепцию *плотности (распределения) вероятностей (probability density)*. В этом разделе мы ограничимся лишь относительно неформальным описанием.

Плотность вероятности $p(x)$ для непрерывной переменной x определяется так, что вероятность попадания x в интервал $(x, x + \delta x)$ определяется как $p(x) \delta x$ для $\delta x \rightarrow 0$. Это показано на рис. 2.6. В этом случае вероятность появления x в интервале (a, b) определяется следующим образом:

$$p(x \in (a, b)) = \int_a^b p(x) dx. \quad (2.23)$$

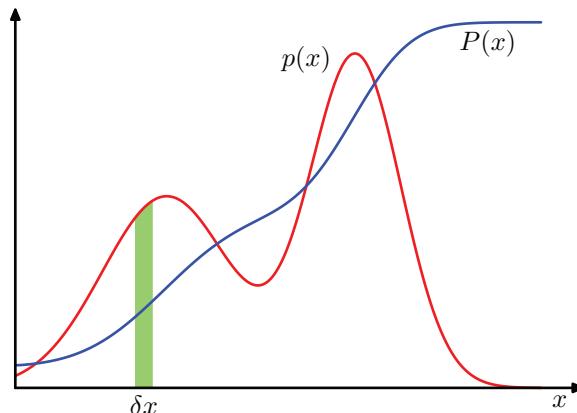


РИС. 2.6 Концепция плотности распределения вероятности $p(x)$ для непрерывной переменной x . Плотность распределения вероятности может быть выражена как производная от интегральной функции распределения $P(x)$

Поскольку вероятности неотрицательны и значение x находится где-то на вещественной оси, плотность распределения вероятностей $p(x)$ должна удовлетворять двум условиям:

$$p(x) \geq 0, \quad (2.24)$$

$$\int_{-\infty}^{\infty} p(x) dx = 1. \quad (2.25)$$

Вероятность того, что x лежит в интервале $(-\infty, z)$, задается интегральной функцией распределения (cumulative distribution function), определяемой как

$$P(z) = \int_{-\infty}^z p(x) dx, \quad (2.26)$$

которая удовлетворяет $P'(x) = p(x)$, как показано на рис. 2.6.

При наличии нескольких непрерывных переменных x_1, \dots, x_D , обозначаемых совокупно вектором \mathbf{x} , для них можно определить совместную плотность распределения вероятностей $p(\mathbf{x}) = p(x_1, \dots, x_D)$ таким образом, что вероятность того, что \mathbf{x} попадет в бесконечно малый промежуток $d\mathbf{x}$, содержащий точку \mathbf{x} , будет равна $p(\mathbf{x})d\mathbf{x}$. Эта многомерная плотность вероятности должна удовлетворять следующим условиям:

$$p(\mathbf{x}) > 0, \quad (2.27)$$

$$\int p(\mathbf{x})d\mathbf{x} = 1, \quad (2.28)$$

где интеграл берется по всему интервалу \mathbf{x} . В более общем случае можно также рассматривать совместные распределения вероятностей по комбинации дискретных и непрерывных переменных.

Правила суммы и произведения вероятностей, как и теорема Байеса, также применимы к плотностям распределения вероятностей, как и к комбинациям дискретных и непрерывных переменных. Если \mathbf{x} и \mathbf{y} представляют собой две вещественные переменные, то правила суммы и произведения имеют вид:

$$\text{правило суммы } p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y})d\mathbf{y}, \quad (2.29)$$

$$\text{правило произведения } p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} | \mathbf{x})p(\mathbf{x}). \quad (2.30)$$

Аналогично теорема Байеса может быть записана в виде

$$p(\mathbf{y} | \mathbf{x}) = \frac{p(\mathbf{x} | \mathbf{y})p(\mathbf{y})}{p(\mathbf{x})}, \quad (2.31)$$

где знаменатель определяется следующим образом:

$$p(\mathbf{x}) = \int p(\mathbf{x} | \mathbf{y})p(\mathbf{y})d\mathbf{y}. \quad (2.32)$$

Формальное обоснование правил суммы и произведения для непрерывных переменных связано с разделом математики, называемым *теорией меры* (*measure theory*) (Feller, 1966), и выходит за рамки обсуждения этой книги. Однако в их справедливости можно убедиться неформальным образом, разбив каждую вещественную переменную на интервалы шириной Δ и рассмотрев дискретное распределение вероятности по этим интервалам. Взятие предела $\Delta \rightarrow 0$ превращает суммы в интегралы и позволяет получить желаемый результат.

2.2.1. Примеры распределений

Существует множество форм распределения плотности вероятности, которые широко используются и которые важны как сами по себе, так и в качестве составных элементов для более сложных вероятностных моделей. Простейшей формой является распределение, в котором $p(x)$ – это константа, не

зависящая от x , но такое распределение нельзя нормализовать, поскольку интеграл в (2.28) будет расходиться. Распределения, которые нельзя нормализовать, называются *несобственными* (*improper*). Однако есть равномерное распределение, которое постоянно в конечной области, к примеру (c, d) , и равно нулю в других местах, и тогда из (2.28) следует

$$p(x) = 1/(d - c), \quad x \in (c, d). \quad (2.33)$$

Другой простой формой распределения плотности является *экспоненциальное распределение* (*exponential distribution*), которое имеет вид:

$$p(x|\lambda) = \lambda \exp(-\lambda x), \quad x > 0. \quad (2.34)$$

Вариант экспоненциального распределения, известный как *распределение Лапласа* (*Laplace distribution*), предусматривает перемещение пика в точку μ и задается выражением:

$$p(x|\mu, \gamma) = \frac{1}{2\gamma} \exp\left(-\frac{|x - \mu|}{\gamma}\right). \quad (2.35)$$

Постоянное, экспоненциальное и распределение Лапласа показаны на рис. 2.7.

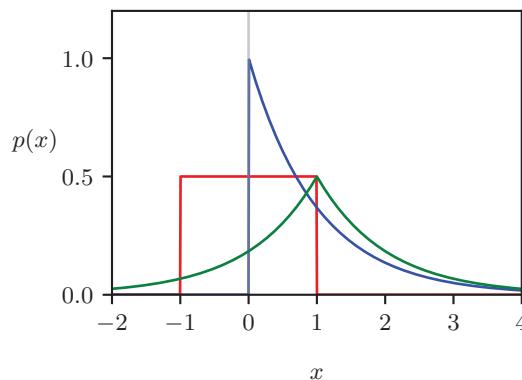


РИС. 2.7 Графики равномерного распределения в интервале $(-1, 1)$ – красный, экспоненциального распределения с $\lambda = 1$ – синий и распределения Лапласа с $\mu = 1$ и $\gamma = 1$ – зеленый

Еще одним важнейшим распределением является *дельта-функция Дирака* (*Dirac delta function*), которая записывается в виде

$$p(x|\mu) = \delta(x - \mu). \quad (2.36)$$

Она определяется как нулевая везде, кроме точки $x = \mu$, и обладает свойством интегрирования к единице в соответствии с (2.28). Фактически это можно представить в виде бесконечно узкого и бесконечно высокого пика, расположенного в точке $x = \mu$ и обладающего свойством единичной площади.

Наконец, если есть конечный набор данных наблюдений за x , заданный как $\mathcal{D} = \{x_1, \dots, x_N\}$, то с помощью дельта-функции можно построить *эмпирическое распределение* (*empirical distribution*) вида

$$p(x | \mathcal{D}) = \frac{1}{N} \sum_{n=1}^N \delta(x - x_n), \quad (2.37)$$

которое состоит из дельта-функции Дирака, отцентрированной на каждой из точек данных. Плотность вероятности, определяемая в (2.37), интегрируется в единицу, как и положено (см. упражнение 2.6).

2.2.2. Ожидания и ковариации

Одной из важнейших операций с вероятностями является нахождение средневзвешенных значений функций. Средневзвешенное значение некоторой функции $f(x)$ при распределении вероятностей $p(x)$ называется *ожиданием* (*expectation*) для $f(x)$ и обозначается $\mathbb{E}[f]$. Для дискретного распределения оно задается суммированием по всем возможным значениям x в виде

$$\mathbb{E}[f] = \sum_x p(x)f(x), \quad (2.38)$$

где среднее значение взвешивается по относительным вероятностям различных значений x . Для непрерывных переменных ожидания выражаются в форме интегрирования по соответствующей плотности распределения вероятностей:

$$\mathbb{E}[f] = \int p(x)f(x)dx. \quad (2.39)$$

В любом случае если задано конечное число N точек, взятых из распределения или плотности распределения вероятностей, то ожидание можно аппроксимировать конечной суммой по этим точкам (см. упражнение 2.7):

$$\mathbb{E}[f] \simeq \frac{1}{N} \sum_{n=1}^N f(x_n). \quad (2.40)$$

Приближение в (2.40) становится точным в пределе $N \rightarrow \infty$.

Иногда требуется оценить ожидания функций нескольких переменных, и в этом случае можно использовать подстрочный индекс, чтобы указать переменную, по которой производится усреднение; например

$$\mathbb{E}_x[f(x, y)] \quad (2.41)$$

обозначает среднее значение функции $f(x, y)$ относительно распределения x . Обратите внимание, что $\mathbb{E}_x[f(x, y)]$ будет функцией y .

Можно также рассмотреть *условное ожидание* (*conditional expectation*) относительно условного распределения, и тогда

$$\mathbb{E}_x[f|y] = \sum_x p(x|y)f(x), \quad (2.42)$$

что также является функцией y . Для непрерывных переменных условное ожидание имеет вид:

$$\mathbb{E}_x[f|y] = \int p(x|y)f(x)dx. \quad (2.43)$$

Дисперсия (variance) для $f(x)$ определяется как

$$\text{var}[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] \quad (2.44)$$

и дает представление о том, насколько сильно $f(x)$ колеблется относительно своего среднего значения $\mathbb{E}[f(x)]$.

Разложив эту формулу по квадратам, можно увидеть, что дисперсия также может быть записана в форме ожиданий $f(x)$ и $f(x)^2$ (см. упражнение 2.8):

$$\text{var}[f] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2. \quad (2.45)$$

В частности, можно рассматривать дисперсию самой переменной x , которая задается как

$$\text{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2. \quad (2.46)$$

Для двух случайных величин x и y показатель *ковариации (covariance)* определяет степень совместного изменения этих двух величин и выражается как

$$\begin{aligned} \text{cov}[x, y] &= \mathbb{E}_{x,y}[\{x - \mathbb{E}[x]\}\{y - \mathbb{E}[y]\}] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y]. \end{aligned} \quad (2.47)$$

Если x и y независимы, то их ковариация равна нулю (см. упражнение 2.9). Ковариация двух векторов x и y является матрицей, заданной как

$$\begin{aligned} \text{cov}[x, y] &= \mathbb{E}_{x,y}\{x - \mathbb{E}[x]\}\{y^T - \mathbb{E}[y^T]\} \\ &= \mathbb{E}_{x,y}[xy^T] - \mathbb{E}[x]\mathbb{E}[y^T]. \end{aligned} \quad (2.48)$$

Если рассматривать ковариацию компонентов вектора x друг с другом, то можно использовать несколько более простую запись: $\text{cov}[x] \equiv \text{cov}[x, x]$.

2.3. Гауссово распределение

Одним из наиболее важных распределений вероятностей для непрерывных переменных является *нормальное (normal)*, или *гауссово, распределение (Gaussian distribution)*, которое будет активно применяться на протяжении всей этой книги. Для отдельной вещественной переменной x гауссово распределение описывается как

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}, \quad (2.49)$$

что представляет собой плотность распределения вероятностей по x с управлением двумя параметрами: μ , который называется *средним значением (mean)*, и σ^2 , который называется *дисперсией (variance)*. Квадратный корень из дисперсии, выраженный через σ , называется *стандартным отклонением (standard deviation)*, а величина, обратная дисперсии и записываемая как $\beta = 1/\sigma^2$, называется *точностью (precision)*. Причины использования этой терминологии станут понятны очень скоро. На рис. 2.8 показан график гауссова распределения. Хотя форма гауссова распределения может показаться произвольной, позже будет показано, что она естественным образом вытекает из концепции максимальной энтропии (см. раздел 2.5.4) и концепции центральной предельной теоремы (см. раздел 3.2).

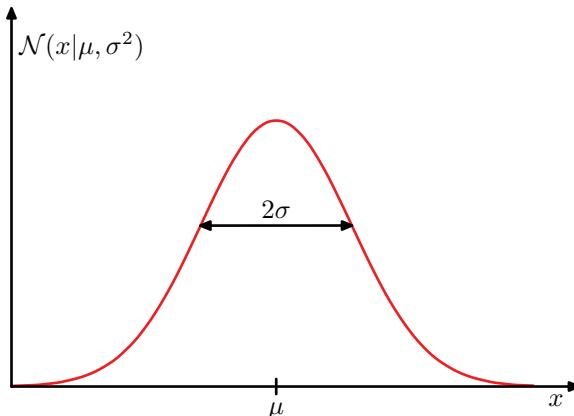


РИС. 2.8 График гауссова распределения для отдельной непрерывной переменной x со средним значением (μ) и стандартным (σ) отклонением

Из (2.49) следует, что гауссово распределение удовлетворяет условию

$$\mathcal{N}(x|\mu, \sigma^2) > 0. \quad (2.50)$$

Кроме того, достаточно просто можно показать, что гауссово распределение является нормализованным (см. упражнение 2.12), так что

$$\int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) dx = 1. \quad (2.51)$$

Получается, что выражение (2.49) удовлетворяет двум требованиям, предъявляемым к действительной плотности распределения вероятностей.

2.3.1. Среднее значение и дисперсия

Ожидания функций x можно без проблем найти по гауссовому распределению (см. упражнение 2.13). В частности, среднее значение x определяется как

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) x dx = \mu. \quad (2.52)$$

Поскольку параметр μ представляет собой усредненное значение x по распределению, он и называется средним значением. Интеграл в (2.52) называется *моментом первого порядка* распределения (*first-order moment*), поскольку он представляет расчетное ожидание x , возведенное в степень единицы. Аналогичным образом можно оценить момент второго порядка:

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) x^2 dx = \mu^2 + \sigma^2. \quad (2.53)$$

Из (2.52) и (2.53) следует, что дисперсия x задается соотношением

$$\text{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2 \quad (2.54)$$

и, следовательно, σ^2 именуют *параметром дисперсии* (*variance parameter*). Максимум распределения называется *модой* (вершиной) распределения (*mode*), (см. упражнение 2.14). Для гауссова распределения мода совпадает со средним значением.

2.3.2. Функция правдоподобия

Пусть задан набор наблюдений в виде скалярного вектора-строки $\mathbf{x} = (x_1, \dots, x_N)$, представляющего N наблюдений скалярной переменной x . Обратите внимание, что мы используем другое начертание \mathbf{x} , чтобы отличать его от отдельного наблюдения D -мерной векторнозначной переменной, представленной вектором-столбцом $\mathbf{x} = (x_1, \dots, x_D)^T$. Предположим, что наблюдения взяты независимой выборкой из гауссова распределения, среднее значение μ и дисперсия σ^2 которого неизвестны, и нам хотелось бы определить эти параметры из набора данных. Задача оценки распределения, заданного конечным набором наблюдений, называется оценкой плотности распределения (*density estimation*). Следует подчеркнуть, что задача оценки плотности является фундаментально некорректной, поскольку существует бесконечно много распределений возможных вероятностей, которые могли бы давать наблюдаемый конечный набор данных. Действительно, любое распределение $p(\mathbf{x})$, которое ненулевое в каждой из точек данных $\mathbf{x}_1, \dots, \mathbf{x}_N$, является одним из потенциальных вариантов. В этом случае необходимо ограничить пространство распределений гауссовыми распределениями, что позволяет получить вполне определимое решение.

Точки данных, взятые независимо из одного и того же распределения, называются *независимыми и одинаково распределенными* (*independent and identically distributed*), что зачастую сокращается до i.i.d. или IID. Ранее было показано, что совместная вероятность двух независимых событий определяется произведением предельных вероятностей для каждого события в отдельности. Поскольку наш набор данных \mathbf{x} является i.i.d., вероятность набора данных, заданного μ и σ^2 , можно записать как

$$p(\mathbf{x} | \mu, \sigma^2) = \prod_{n=1}^N \mathcal{N}(x_n | \mu, \sigma^2). \quad (2.55)$$

Если рассматривать эту функцию в качестве функции от μ и σ^2 , ее можно назвать *функцией правдоподобия* (*likelihood function*) для гауссова распределения и представить в виде диаграммы, показанной на рис. 2.9. Максимизация правдоподобия предполагает корректировку среднего и дисперсию гауссова распределения таким образом, чтобы максимизировать это произведение.

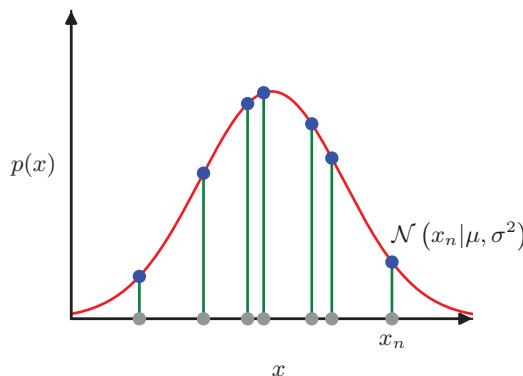


РИС. 2.9 Иллюстрация функции правдоподобия для гауссова распределения (показано красной кривой). Серые точки обозначают набор значений $\{x_n\}$, а функция правдоподобия (2.55) дается произведением соответствующих значений $p(x)$, обозначенных синими точками

Одним из распространенных подходов к определению параметров распределения вероятностей по наблюдаемому набору данных, называемым *методом максимального правдоподобия* (*maximum likelihood*), является поиск значений параметров, при которых функция правдоподобия максимизируется. Это может показаться странным критерием, потому что, исходя из нашего предыдущего обсуждения теории вероятностей, более естественным представляется максимизация вероятности параметров с учетом данных, а не вероятности данных с учетом параметров (см. раздел 2.6.2). На самом деле эти два критерия являются взаимосвязанными.

Однако для начала определим значения неизвестных параметров μ и σ^2 в гауссовом распределении, максимизируя функцию правдоподобия (2.55). На практике удобнее максимизировать логарифм функции правдоподобия.

Поскольку логарифм является монотонно возрастающей функцией своего аргумента, максимизация логарифма функции эквивалентна максимизации самой функции. Взятие логарифма не только упрощает последующий математический анализ, но также помогает в численном выражении, поскольку произведение большого числа малых вероятностей может легко превысить точность компьютера, а это можно исправить путем вычисления суммы логарифмов вероятностей. Исходя из (2.49) и (2.55), функция логарифмического правдоподобия может быть записана в виде

$$\ln p(\mathbf{x} | \mu, \sigma^2) = -\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi). \quad (2.56)$$

При максимизации соотношения (2.56) относительно μ получаем решение с максимальным правдоподобием (см. упражнение 2.15), которое выглядит как

$$\mu_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N x_n, \quad (2.57)$$

что является *выборочным средним значением* (*sample mean*), т. е. средним значением наблюдаемых величин $\{x_n\}$. Аналогично при максимизации (2.56) относительно σ^2 получаем решение максимального правдоподобия для дисперсии в виде

$$\sigma_{\text{ML}}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu_{\text{ML}})^2, \quad (2.58)$$

которая представляет собой *дисперсию выборки* (*sample variance*), измеренную относительно выборочного среднего значения μ_{ML} . Обратите внимание, что здесь выполняется совместная максимизация (2.56) относительно μ и σ^2 , но для гауссова распределения решение для μ отделяется от решения для σ^2 , что позволяет сначала определить (2.57), а затем использовать этот результат для оценки (2.58).

2.3.3. Ошибка максимального правдоподобия

Метод максимального правдоподобия широко используется в глубоком обучении и лежит в основе большинства алгоритмов машинного обучения. Однако у него есть ряд ограничений, которые можно проиллюстрировать на примере одномерного гауссова распределения.

Прежде всего следует отметить, что решения максимального правдоподобия μ_{ML} и σ_{ML}^2 являются функциями значений набора данных x_1, \dots, x_N . Предположим, что каждое из этих значений было независимо сгенерировано из гауссова распределения, истинные параметры которого равны μ и σ^2 . Теперь рассмотрим ожидания μ_{ML} и σ_{ML}^2 относительно этих значений набора данных (см. упражнение 2.16). Достаточно просто продемонстрировать, что

$$\mathbb{E}[\mu_{\text{ML}}] = \mu, \quad (2.59)$$

$$\mathbb{E}[\sigma_{\text{ML}}^2] = \left(\frac{N-1}{N} \right) \sigma^2. \quad (2.60)$$

Как видно, при усреднении по наборам данных заданного размера решение максимального правдоподобия для среднего значения будет равно истинному среднему значению. Однако оценка дисперсии с помощью максимального правдоподобия будет занижать истинную дисперсию в $(N-1)/N$ раз. Это пример явления, называемого *смещением (bias)*, при котором оценка случайной величины систематически отличается от истинного значения. Интуитивное видение этого эффекта представлено на рис. 2.10. Красные кривые показывают истинное гауссово распределение, на основе которого генерируются данные, а три синие кривые показывают гауссова распределения, полученные в результате подгонки к трем наборам данных, каждый из которых состоит из двух точек данных, показанных зеленым цветом, с использованием результатов максимального правдоподобия (2.57) и (2.58). Усредненное по трем наборам данных среднее значение верно, но дисперсия систематически недооценивается, поскольку измеряется относительно среднего значения выборки, а не относительно истинного среднего.

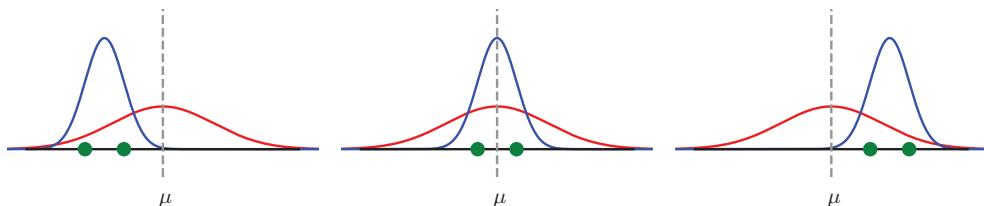


РИС. 2.10 Иллюстрация появления смещения при использовании метода максимального правдоподобия для определения среднего значения и дисперсии гауссова распределения

Обратите внимание, что смещение возникает из-за оценки дисперсии относительно оценки среднего значения по методу максимального правдоподобия, который сам подстраивается под данные. Предположим, что вместо этого у нас есть доступ к истинному среднему μ , и мы используем его для определения дисперсии с помощью оценки

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu)^2. \quad (2.61)$$

Затем мы находим (см. упражнение 2.17), что

$$\mathbb{E}[\hat{\sigma}^2] = \sigma^2, \quad (2.62)$$

которое является несмещенным. Конечно, у нас нет доступа к истинному среднему значению, а есть только наблюдаемые значения данных. Из результата (2.60) следует, что для гауссова распределения несмещенной является следующая оценка параметра дисперсии:

$$\tilde{\sigma}^2 = \frac{N}{N-1} \sigma_{\text{ML}}^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \mu_{\text{ML}})^2. \quad (2.63)$$

Корректировка смещения максимального правдоподобия в сложных моделях, таких как нейронные сети, тем не менее оказывается не такой уж простой.

Отметим, что смещение решения максимального правдоподобия становится менее значительным с увеличением числа N точек данных. В пределе $N \rightarrow \infty$ решение максимального правдоподобия для дисперсии равно истинной дисперсии распределения, сформировавшего данные. В случае гауссова распределения при любом, кроме малого N , это смещение не будет представлять серьезной проблемы. Однако на протяжении всей этой книги речь будет идти о сложных моделях со многими параметрами, для которых проблемы со смещением, связанные с максимальным правдоподобием, будут гораздо серьезнее. На самом деле проблема смещения в методе максимального правдоподобия тесно связана с проблемой чрезмерной подгонки (*over-fitting*) (см. раздел 2.6.3).

2.3.4. Линейная регрессия

Ранее (см. раздел 1.2) нами рассматривалась задача линейной регрессии в контексте минимизации ошибок. Теперь мы вернемся к этому примеру и рассмотрим его с позиции вероятностного подхода, тем самым получив некоторое представление о функциях ошибок и регуляризации.

Цель решения задачи регрессии – получить возможность строить прогнозы для целевой переменной t при некотором новом значении входной переменной x с использованием набора обучающих данных, состоящего из N входных значений $\mathbf{x} = (x_1, \dots, x_N)$ и соответствующих им целевых значений $\mathbf{t} = (t_1, \dots, t_N)$. Неопределенность в отношении значения целевой переменной можно выразить с помощью распределения вероятностей. Для этого будем считать, что при заданном значении x соответствующее значение t имеет гауссово распределение со средним значением, равным значению $y(x, \mathbf{w})$ полиномиальной кривой, заданной в (1.1), где \mathbf{w} – это коэффициенты полинома, и дисперсией σ^2 . Таким образом, получаем

$$p(t|x, \mathbf{w}, \sigma^2) = \mathcal{N}(t|y(x, \mathbf{w}), \sigma^2). \quad (2.64)$$

Схематично это показано на рис. 2.11.

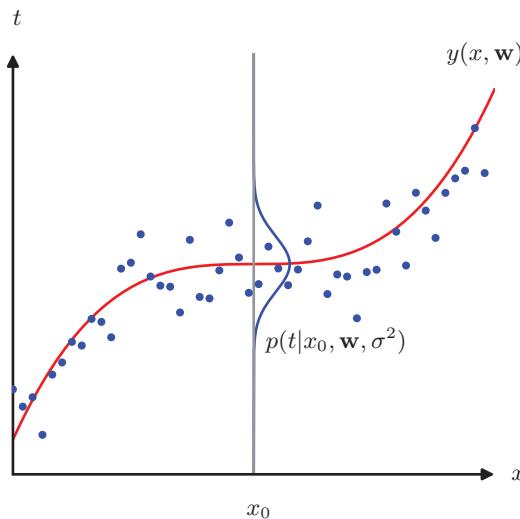


РИС. 2.11 Схематическое изображение гауссова условного распределения для t при x , определяемого (2.64), в котором среднее значение задается полиномиальной функцией $y(x, \mathbf{w})$, а дисперсия — параметром σ^2

Теперь воспользуемся обучающими данными $\{\mathbf{x}, \mathbf{t}\}$ для определения значений неизвестных параметров \mathbf{w} и σ^2 методом максимального правдоподобия. Если предполагается, что данные берутся независимо из распределения (2.64), то функция правдоподобия имеет вид:

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(t_n | y(x_n, \mathbf{w}), \sigma^2). \quad (2.65)$$

Как и в случае простого гауссова распределения, удобнее всего максимизировать логарифм функции правдоподобия. Подставляя в (2.49) гауссово распределение, получаем логарифмическую функцию правдоподобия в виде

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \sigma^2) = -\frac{1}{2\sigma^2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi). \quad (2.66)$$

Сначала рассмотрим оценку решения максимального правдоподобия для коэффициентов полинома, которые будем обозначать \mathbf{w}_{ML} . Они определяются путем максимизации (2.66) относительно \mathbf{w} . Для этого можно опустить два последних члена в правой части (2.66), поскольку они не зависят от \mathbf{w} . Также обратите внимание, что масштабирование логарифмического правдоподобия на положительный постоянный коэффициент не меняет местоположение максимума относительно \mathbf{w} , и поэтому коэффициент $1/2\sigma^2$ можно заменить на $1/2$.

Наконец, вместо максимизации логарифмического правдоподобия можно эквивалентно минимизировать отрицательное логарифмическое правдоподобие. Таким образом, максимизация правдоподобия равнозначна ми-

нимизации функции ошибок суммы квадратов (*sum-of-squares error function*), определяемой как

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2. \quad (2.67)$$

Следовательно, функция погрешности суммы квадратов является следствием максимизации правдоподобия в предположении о гауссовом распределении шума.

Для определения параметра дисперсии σ^2 также можно использовать максимальное правдоподобие. Максимизация (2.66) относительно σ^2 (см. упражнение 2.18) дает

$$\sigma_{ML}^2 = \frac{1}{N} \sum_{n=1}^N \{y(x_n, \mathbf{w}_{ML}) - t_n\}^2. \quad (2.68)$$

Обратите внимание, что для начала нужно рассчитать вектор параметров \mathbf{w}_{ML} , определяющий среднее значение, а затем использовать его для нахождения дисперсии σ_{ML}^2 , как это было в случае простого гауссова распределения.

Определив параметры \mathbf{w} и σ^2 , можно строить прогнозы для новых значений x . Поскольку теперь мы имеем вероятностную модель, они выражаются в виде *прогнозируемого распределения* (*predictive distribution*), которое дает распределение вероятностей по t , а не просто точечную оценку, и получается путем подстановки параметров максимального правдоподобия в (2.64), что дает

$$p(t|x, \mathbf{w}_{ML}, \sigma_{ML}^2) = \mathcal{N}(t|y(x, \mathbf{w}_{ML}), \sigma_{ML}^2). \quad (2.69)$$

2.4. Преобразование плотностей

Перейдем к изучению процесса преобразования плотности распределения вероятностей при нелинейном изменении переменной. Это свойство будет играть важную роль в ходе изучения класса генеративных моделей (см. главу 18), называемых *нормализующими потоками* (*normalizing flows*). Это свойство также указывает на то, что плотность распределения вероятностей при таких преобразованиях проявляет себя иначе, чем просто функция.

Рассмотрим отдельную переменную x и, допустим, выполним замену переменных $x = g(y)$, после чего функция $f(x)$ преобразуется в новую функцию $\tilde{f}(y)$, определяемую как

$$\tilde{f}(y) = f(g(y)). \quad (2.70)$$

Теперь рассмотрим плотность распределения вероятностей $p_x(x)$ и снова поменяем переменные местами посредством $x = g(y)$, что приведет к плотности $p_y(y)$ относительно новой переменной y , где суффиксы указывают на

то, что $p_x(x)$ и $p_y(y)$ – это разные плотности. Наблюдения, попадающие в диапазон $(x, x + \delta x)$, при малых значениях δx будут преобразованы в диапазон $(y, y + \delta y)$, где $x = g(y)$, а $p_x(x)\delta x \simeq p_y(y)\delta y$. Следовательно, если взять предел $\delta x \rightarrow 0$, то получится

$$\begin{aligned} p_y(y) &= p_x(x) \left| \frac{dx}{dy} \right| \\ &= p_x(g(y)) \left| \frac{dg}{dy} \right|. \end{aligned} \quad (2.71)$$

Здесь модуль $|\cdot|$ возникает в связи с тем, что производная dy/dx может быть отрицательной, тогда как плотность определяется отношением длин, которое всегда положительно.

Эта операция преобразования плотностей может быть весьма эффективной. Любая плотность $p(y)$ может быть получена из фиксированной плотности $q(x)$, имеющей везде ненулевое значение, путем нелинейной замены переменной $y = f(x)$, где $f(x)$ – это монотонная функция, так что $0 \leq f'(x) < \infty$ (см. упражнение 2.19).

Следствием свойства преобразования (2.71) является то, что определение максимума для плотности вероятности зависит от выбора переменной. Предположим, что $f(x)$ имеет моду (т. е. максимум) в точке \hat{x} , так что $f'(\hat{x}) = 0$. Соответствующая мода $\hat{f}(y)$ будет иметь место для значения \hat{y} , полученного дифференцированием обеих сторон (2.70) относительно y :

$$\hat{f}'(\hat{y}) = f'(g(\hat{y}))g'(\hat{y}) = 0. \quad (2.72)$$

Если предположить, что $g'(\hat{y}) \neq 0$ в моде, то $f'(g(\hat{y})) = 0$. Однако уже известно, что $f'(\hat{x}) = 0$, и поэтому видно, что местоположение моды, выраженное в виде каждой из переменных x и y , связано через $\hat{x} = g(\hat{y})$, как и следовало ожидать. Таким образом, нахождение моды по переменной x эквивалентно преобразованию к переменной y , последующему нахождению моды по y и дальнейшему обратному преобразованию к x .

Теперь рассмотрим поведение плотности вероятности $p_x(x)$ при замене переменных $x = g(y)$, где плотность относительно новой переменной равна $p_y(y)$ и задается (2.71). Чтобы разобраться с модулем в (2.71), мы можем записать $g'(y) = s|g'(y)|$, где $s \in \{-1, +1\}$. Тогда (2.71) можно записать в виде

$$p_y(y) = p_x(g(y))sg'(y),$$

где мы использовали $1/s = s$. Дифференцирование обеих сторон относительно y даст

$$p'_y(y) = sp'_x(g(y))\{g'(y)\}^2 + sp_x(g(y))g''(y). \quad (2.73)$$

Поскольку в правой части (2.73) присутствует второй член, соотношение $\hat{x} = g(\hat{y})$ больше не выполняется. Таким образом, значение x , полученное при

максимизации $p_x(x)$, не будет равно значению, полученному при преобразовании к $p_y(y)$ с последующей максимизацией по y и обратным преобразованием к x . Это приводит к зависимости мод плотностей от выбора переменных. Однако при линейном преобразовании второй член в правой части (2.73) исчезает, и тогда местоположение максимума преобразуется в соответствии с $\hat{x} = g(\hat{y})$.

Этот эффект можно проиллюстрировать на простом примере, как показано на рис. 2.12. Для начала рассмотрим гауссово распределение $p_x(x)$ по x , показанное красной кривой на рис. 2.12. Затем возьмем выборку из $N = 50\,000$ точек этого распределения и построим гистограмму их значений, которая, как и ожидалось, согласуется с распределением $p_x(x)$. Теперь рассмотрим нелинейный переход переменных от x к y , заданный выражением

$$x = g(y) = \ln(y) - \ln(1 - y) + 5. \quad (2.74)$$

Обратной функцией этой функции является

$$y = g^{-1}(x) = \frac{1}{1 + \exp(-x + 5)}, \quad (2.75)$$

которая представляет собой *логистическую сигмоидную (logistic sigmoid)* функцию и показана на рис. 2.12 синей кривой.

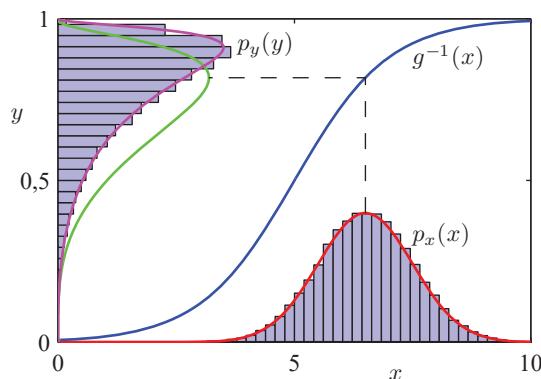


РИС. 2.12 Пример преобразования моды плотности при нелинейной смене переменных, показывающий иное поведение по сравнению с простой функцией

Если просто преобразовать $p_x(x)$ как функцию от x , то получится зеленая кривая $p_x(g(y))$, показанная на рис. 2.12, и можно увидеть, что мода плотности $p_x(x)$ преобразуется через сигмоидную функцию в моду этой кривой. Однако плотность по y преобразуется в соответствии с (2.71) и показана пурпурной кривой в левой части диаграммы. Обратите внимание, что ее мода смешена относительно моды зеленой кривой.

Чтобы подтвердить этот результат, возьмем нашу выборку из 50 000 значений x , оценим соответствующие значения y с помощью (2.75), а затем постро-

им гистограмму их значений. В результате окажется, что эта гистограмма соответствует пурпурной кривой на рис. 2.12, а не зеленой.

2.4.1. Многомерное распределение

Результат (2.71) можно применить к плотностям, определяемым над несколькими переменными. Рассмотрим плотность $p(\mathbf{x})$ по D -мерной переменной $\mathbf{x} = (x_1, \dots, x_D)^T$, и представим, что мы преобразуем ее к новой переменной $\mathbf{y} = (y_1, \dots, y_D)^T$, где $\mathbf{x} = \mathbf{g}(\mathbf{y})$. Здесь мы ограничимся случаем, когда \mathbf{x} и \mathbf{y} имеют одинаковую размерность. Тогда преобразованная плотность определяется обобщением (2.71) вида

$$p_y(\mathbf{y}) = p_x(\mathbf{x}) |\det \mathbf{J}|, \quad (2.76)$$

где \mathbf{J} – это *матрица Якоби (Jacobian matrix)*, элементы которой задаются частными производными $J_{ij} = \partial g_i / \partial y_j$, поэтому

$$\mathbf{J} = \begin{bmatrix} \frac{\partial g_1}{\partial y_1} & \dots & \frac{\partial g_1}{\partial y_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_D}{\partial y_1} & \dots & \frac{\partial g_D}{\partial y_D} \end{bmatrix}. \quad (2.77)$$

Фактически замену переменных можно рассматривать как расширение одних областей пространства и сужение других, при этом бесконечно малая область $\Delta \mathbf{x}$ вокруг точки \mathbf{x} преобразуется в область $\Delta \mathbf{y}$ вокруг точки $\mathbf{y} = \mathbf{g}(\mathbf{x})$. Абсолютное значение *определителя матрицы Якоби (якобиана)* представляет собой отношение этих объемов и является тем же самым коэффициентом, который возникает при смене переменных в интеграле. Формула (2.77) следует из того факта, что вероятностная мера в области $\Delta \mathbf{x}$ такая же, как и вероятностная мера в области $\Delta \mathbf{y}$. И снова нужно брать модуль, чтобы гарантировать, что плотность неотрицательна.

Это можно проиллюстрировать заменой переменных для гауссова распределения в двух измерениях, как показано в верхней строке на рис. 2.13. Здесь преобразование (см. упражнение 2.20) из \mathbf{x} в \mathbf{y} выполняется следующим образом:

$$y_1 = x_1 + \tanh(5x_1), \quad (2.78)$$

$$y_2 = x_2 + \tanh(5x_2) + \frac{x_1^3}{3}. \quad (2.79)$$

В нижней строке также показаны выборки из гауссова распределения в пространстве \mathbf{x} и соответствующие преобразованные выборки в пространстве \mathbf{y} .

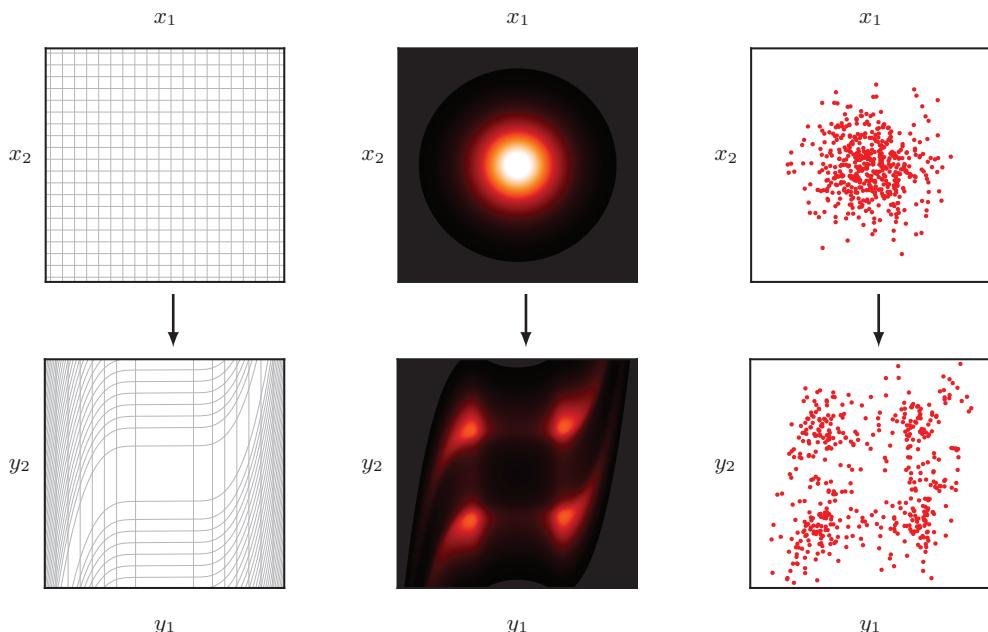


РИС. 2.13 Иллюстрация влияния изменения переменных на распределение вероятностей в двух измерениях. В левом столбце показано преобразование переменных, а в среднем и правом столбцах – соответствующие эффекты для гауссова распределения и выборок из него соответственно

2.5. Теория информации

Теория вероятностей служит основой для другой важной концепции, называемой теорией информации (information theory), которая дает возможность определить количество информации, содержащейся в наборе данных, и играет важную роль в машинном обучении. В этом разделе представлено краткое введение в некоторые ключевые аспекты теории информации, которые понадобятся нам далее в книге, включая такую важную концепцию, как энтропия, в ее различных проявлениях. Более полное введение в теорию информации с привязкой к машинному обучению см. в (MacKay, 2003).

2.5.1. Энтропия

Начнем с рассмотрения дискретной случайной переменной x и зададимся вопросом, какое количество информации будет получено при наблюдении определенного значения этой переменной. Количество информации можно рассматривать как «степень неожиданности» при получении значения x . Даже если нам скажут, что только что произошло крайне маловероятное событие, мы получим больше информации, чем если бы нам сказали, что только что

произошло какое-то очень вероятное событие. А если мы знали, что событие обязательно произойдет, мы не получили бы никакой информации. Поэтому наша мера информативности будет зависеть от распределения вероятностей $p(x)$, и потому мы ищем величину $h(x)$, которая является монотонной функцией вероятности $p(x)$ и выражает количество информации. Форму $h(\cdot)$ можно определить исходя из того, что если у нас есть два несвязанных события x и y , то информация, полученная от наблюдения за ними обоими, должна быть равна сумме информации, полученной от каждого из них в отдельности, т. е. $h(x, y) = h(x) + h(y)$. Два несвязанных события статистически независимы, поэтому $p(x, y) = p(x)p(y)$. Из этих двух соотношений нетрудно вывести (см. упражнение 2.21), что $h(x)$ должен быть равен логарифму $p(x)$, поэтому имеем

$$h(x) = -\log_2 p(x), \quad (2.80)$$

где знак минус означает, что информация имеет положительное или нулевое значение. Заметим, что события с низкой вероятностью x соответствуют большому количеству информации. Выбор основания для логарифма произведен, но сейчас будем придерживаться распространенной в теории информации традиции использования логарифмов с основанием 2. В этом случае, как вскоре будет показано, единицами измерения $h(x)$ являются биты («двоичные разряды»).

Теперь предположим, что отправитель хочет передать получателю значение случайной переменной. Передаваемый при этом средний объем информации определяется ожиданием (2.80) относительно распределения $p(x)$ и задается как

$$H[x] = -\sum_x p(x) \log_2 p(x). \quad (2.81)$$

Эта важная величина называется *энтропией* (*entropy*) произвольной переменной x . Заметим, что $\lim_{\epsilon \rightarrow 0} (\epsilon \ln \epsilon) = 0$, и поэтому будем принимать $p(x) \ln p(x) = 0$ всякий раз, когда встретим значение x , при котором $p(x) = 0$.

До сих пор для определения информации (2.80) и соответствующей энтропии (2.81) использовалась довольно эвристическая формулировка. Далее будет показано, что эти определения действительно обладают полезными свойствами. Рассмотрим случайную переменную x с восемью возможными состояниями, каждое из которых равновероятно. Чтобы передать значение x получателю, необходимо передать сообщение длиной в 3 бита. Обратите внимание, что энтропия этой переменной определяется как

$$H[x] = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ бита.}$$

Теперь рассмотрим пример (Cover and Thomas, 1991) переменной с восемью возможными состояниями $\{a, b, c, d, e, f, g, h\}$, для которых соответствующие вероятности заданы $(1/2, 1/4, 1/8, 1/16, 1/64, 1/64, 1/64, 1/64)$. Энтропия в этом случае определяется как

$$H[x] = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{1}{8} \log_2 \frac{1}{8} - \frac{1}{16} \log_2 \frac{1}{16} - \frac{4}{64} \log_2 \frac{4}{64} = 2 \text{ бита.}$$

Как видно, неравномерное распределение имеет меньшую энтропию, чем равномерное, и скоро это станет понятным, поскольку мы будем рассматривать интерпретацию энтропии в контексте неупорядоченности. Пока же рассмотрим способы передачи информации о состоянии переменной приемнику. Как и прежде, это можно сделать с помощью 3-битного числа. Однако при этом можно воспользоваться преимуществами неравномерного распределения и передавать более короткие коды для более вероятных событий за счет более длинных кодов для менее вероятных событий в надежде получить меньшую среднюю длину кода. Это можно сделать, представляя состояния $\{a, b, c, d, e, f, g, h\}$ с помощью, например, следующего набора кодовых строк: 0, 10, 110, 1110, 111100, 111101, 111110 и 111111. Тогда средняя длина кода, который необходимо передать, равна:

$$\text{средняя длина кода} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + 4 \times \frac{1}{64} \times 6 = 2 \text{ бита},$$

что опять же равно энтропии случайной величины. Заметим, что более короткие кодовые строки использовать нельзя, поскольку конкатенация таких строк должна быть разборчивой на составные части. Например, 11001110 однозначно декодируется в последовательность состояний c, a, d . Эта связь между энтропией и наименьшей длиной кодирования имеет общий характер. *Теорема о бесшумном кодировании (noiseless coding theorem)* (Shannon, 1948) определяет, что энтропия – это нижний предел для количества битов, необходимых для передачи состояния случайной переменной.

С этого момента для определения энтропии будут использоваться натуральные логарифмы, так как это обеспечит более удобную связь с идеями, изложенными в других разделах этой книги. В этом случае энтропия измеряется в единицах *натов* (*nats*, от «натуральный логарифм» – *natural logarithm*), а не в битах, которые отличаются всего лишь в $\ln 2$ раза.

2.5.2. Физическая перспектива

Определение энтропии было представлено в виде среднего количества информации, необходимого для определения состояния случайной переменной. В действительности концепция энтропии гораздо раньше возникла в физике, где она была введена при изучении равновесной термодинамики, а затем получила более глубокую интерпретацию как мера разупорядоченности (*disorder*) в результате развития статистической механики. Понять этот альтернативный подход к определению энтропии поможет изучение набора из N одинаковых предметов, которые должны быть разделены между множеством ячеек таким образом, чтобы в i -й ячейке находилось n_i предметов. Рассмотрим количество различных способов распределения объектов по ячейкам. Существует N способов выбрать первый объект, $(N-1)$ способов вы-

брать второй объект и т. д., что приводит к общему числу $N!$ способов распределить все N объектов по ячейкам, где $N!$ (произносится как «факториал N ») обозначает произведение $N \times (N - 1) \times \dots \times 2 \times 1$. Однако при этом не требуется различать перестановки объектов внутри каждой ячейки. В i -й ячейке существует $n!$ способов переупорядочить объекты, поэтому общее число способов распределения N объектов по ячейкам определяется как

$$W = \frac{N!}{\prod_i n_i!}, \quad (2.82)$$

что называют *множественностью* (*multiplicity*). Энтропия определяется как логарифм этой множественности, выраженный с помощью постоянного множителя $1/N$, так что

$$H = \frac{1}{N} \ln W = \frac{1}{N} \ln N! - \frac{1}{N} \sum_i \ln n_i!. \quad (2.83)$$

Теперь рассмотрим предел $N \rightarrow \infty$, в котором доли n_i/N фиксированы, и применим *приближение Стирлинга*:

$$\ln N! \simeq N \ln N - N, \quad (2.84)$$

что дает

$$H = - \lim_{N \rightarrow \infty} \sum_i \left(\frac{n_i}{N} \right) \ln \left(\frac{n_i}{N} \right) = - \sum_i p_i \ln p_i, \quad (2.85)$$

где используется $\sum_i n_i = N$. Здесь $p_i = \lim_{N \rightarrow \infty} (n_i/N)$ – это вероятность того, что объект будет отнесен к i -й корзине. В физике конкретное распределение объектов по ячейкам называется *микросостоянием* (*microstate*), а общее распределение чисел заполнения, выраженное через соотношения n_i/N , называется *макросостоянием* (*macrostate*). Множество W , выражающее количество микросостояний в определенном макросостоянии, называют также *весом* (*weight*) этого макросостояния.

Можно интерпретировать ячейки как состояния x_i дискретной случайной величины X , где $p(X = x_i) = p_i$. Тогда энтропия случайной переменной X равна

$$H[p] = - \sum_i p(x_i) \ln p(x_i). \quad (2.86)$$

Распределения $p(x_i)$, имеющие резкие пики вблизи нескольких значений, будут иметь относительно низкую энтропию, в то время как распределения, более равномерные по многим значениям, будут иметь более высокую энтропию, как показано на рис. 2.14.

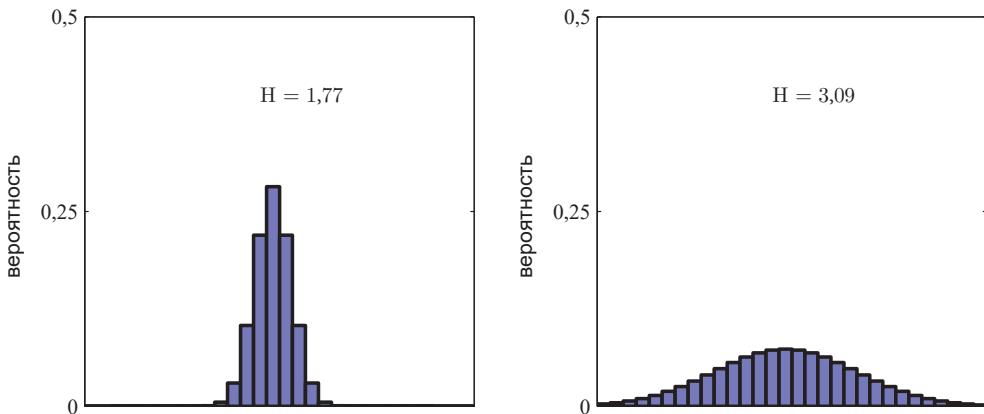


РИС. 2.14 Гистограммы двух распределений вероятностей по 30 ячейкам, иллюстрирующие более высокое значение энтропии H для более широкого распределения. Наибольшая энтропия возникает при равномерном распределении: $H = -\ln(1/30) = 3,40$

Поскольку $0 \leq p_i \leq 1$, энтропия неотрицательна и будет равна своему минимальному значению 0 в случае, когда один из $p_i = 1$, а все остальные $p_{j \neq i} = 0$. Конфигурация с максимальной энтропией может быть найдена путем максимизации H с использованием множителя Лагранжа для обеспечения ограничения нормализации вероятностей (см. приложение С). Таким образом, мы максимизируем

$$\tilde{H} = -\sum_i p(x_i) \ln p(x_i) + \lambda \left(\sum_i p(x_i) - 1 \right), \quad (2.87)$$

из чего следует, что все $p(x_i)$ равны и задаются как $p(x_i) = 1/M$, где M – это общее число состояний x_i . Тогда соответствующее значение энтропии равно $H = \ln M$ (см. упражнение 2.22). Этот результат также может быть получен из неравенства Йенсена (которое будет рассмотрено ниже) (см. упражнение 2.23). Чтобы убедиться, что стационарная точка действительно является максимумом, можно оценить вторую производную энтропии, что дает

$$\frac{\partial \tilde{H}}{\partial p(x_i) \partial p(x_j)} = -I_{ij} \frac{1}{p_i}, \quad (2.88)$$

где I_{ij} – это элементы матрицы тождества. Как видно, все эти значения отрицательны, и, следовательно, стационарная точка действительно является максимумом.

2.5.3. Дифференциальная энтропия

Определение энтропии можно расширить до распределений $p(x)$ по непрерывным переменным x следующим образом. Сначала разделим x на ячейки шириной Δ . Далее, исходя из предположения, что $p(x)$ непрерывно, теоре-

ма о среднем значении (*mean value theorem*) (Weisstein, 1999) подсказывает, что для каждой такой ячейки должно существовать значение x_i в диапазоне $i\Delta \leq x_i \leq (i+1)\Delta$ такое, что

$$\int_{i\Delta}^{(i+1)\Delta} p(x)dx = p(x_i)\Delta. \quad (2.89)$$

Теперь можно квантовать непрерывную переменную x , присваивая любому значению x значение x_i всякий раз, когда x попадает в i -ю ячейку. Тогда вероятность наблюдения значения x_i равна $p(x_i)\Delta$. Это дает дискретное распределение, для которого энтропия имеет вид

$$H_\Delta = -\sum_i p(x_i)\Delta \ln(p(x_i)\Delta) = -\sum_i p(x_i)\Delta \ln p(x_i) - \ln \Delta, \quad (2.90)$$

где мы использовали $\sum_i p(x_i)\Delta = 1$, что следует из (2.89) и (2.25). Теперь опустим второй член $-\ln \Delta$ в правой части (2.90), поскольку он не зависит от $p(x)$, и рассмотрим предел $\Delta \rightarrow 0$. Первый член в правой части (2.90) будет приближаться к интегралу от $p(x) \ln p(x)$ в этом пределе так, что

$$\lim_{\Delta \rightarrow 0} \left\{ -\sum_i p(x_i)\Delta \ln p(x_i) \right\} = -\int p(x) \ln p(x) dx, \quad (2.91)$$

где значение в правой части называется *дифференциальной энтропией* (*differential entropy*). Как видно, дискретная и непрерывная формы энтропии отличаются на величину $\ln \Delta$, которая расходится в пределе $\Delta \rightarrow 0$. Это говорит о том, что для точного определения непрерывной переменной требуется большое количество битов. Для плотности, определенной по нескольким непрерывным переменным, обозначаемым в совокупности вектором \mathbf{x} , дифференциальная энтропия задается как

$$H[\mathbf{x}] = -\int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x}. \quad (2.92)$$

2.5.4. Максимальная энтропия

Для дискретного распределения, как было показано, максимальная конфигурация энтропии соответствует равномерному распределению вероятностей по возможным состояниям переменной. Теперь рассмотрим соответствующий результат для непрерывной переменной. Для точного определения этого максимума необходимо ограничить первое и второе значения $p(x)$, а также сохранить ограничение на нормализацию. В результате достигается максимизация дифференциальной энтропии с тремя ограничениями:

$$\int_{-\infty}^{\infty} p(x)dx = 1, \quad (2.93)$$

$$\int_{-\infty}^{\infty} xp(x)dx = \mu, \quad (2.94)$$

$$\int_{-\infty}^{\infty} (x - \mu)^2 p(x)dx = \sigma^2. \quad (2.95)$$

Ограниченнная максимизация выполняется с помощью множителей Лагранжа таким образом, чтобы максимизировать следующий функционал относительно $p(x)$ (см. приложение С):

$$\begin{aligned} & -\int_{-\infty}^{\infty} p(x) \ln p(x) dx + \lambda_1 \left(\int_{-\infty}^{\infty} p(x) dx - 1 \right) \\ & + \lambda_2 \left(\int_{-\infty}^{\infty} x p(x) dx - \mu \right) + \lambda_3 \left(\int_{-\infty}^{\infty} (x - \mu)^2 p(x) dx - \sigma^2 \right). \end{aligned} \quad (2.96)$$

Используя вариационное исчисление (см. приложение В), сведем производную этого функционала к нулю, что дает

$$p(x) = \exp\{-1 + \lambda_1 + \lambda_2 x + \lambda_3(x - \mu)^2\}. \quad (2.97)$$

Множители Лагранжа можно найти путем обратной подстановки этого результата в три ограничительных уравнения (см. упражнение 2.24), что в итоге приводит к результату:

$$p(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{(x - \mu)^2}{2\sigma^2}\right\}, \quad (2.98)$$

и потому распределение, максимизирующее дифференциальную энтропию, является гауссовым. Обратите внимание, что при максимизации энтропии мы не ограничивали распределение неотрицательностью. Тем не менее, поскольку полученное распределение действительно неотрицательно, с учетом нашего опыта можно понять, что в таком ограничении нет необходимости.

Если оценивать дифференциальную энтропию гауссова распределения (см. упражнение 2.25), получится

$$H(x) = \frac{1}{2} \{1 + \ln(2\pi\sigma^2)\}. \quad (2.99)$$

Таким образом, вновь становится видно, что энтропия возрастает по мере расширения распределения, т. е. по мере увеличения σ^2 . Этот результат также показывает, что дифференциальная энтропия, в отличие от дискретной, может быть отрицательной, поскольку $H(x) < 0$ в (2.99) для $\sigma^2 < 1/(2\pi e)$.

2.5.5. Дивергенция Кульбака–Лейблера

Ранее в этой главе был представлен ряд концепций из теории информации, включая ключевое определение энтропии. Теперь перейдем к изучению этих идей в сфере машинного обучения. Рассмотрим некоторое неизвестное распределение $p(\mathbf{x})$ и предположим, что мы смоделировали его с помощью приближенного распределения $q(\mathbf{x})$. Если использовать $q(\mathbf{x})$ при построении алгоритма кодирования для передачи значений \mathbf{x} приемнику, то средний дополнительный объем информации (в натах), необходимый для определения значения \mathbf{x} (при условии выбора эффективной схемы кодирования) в результате использования $q(\mathbf{x})$ вместо истинного распределения $p(\mathbf{x})$, определяется как

$$\begin{aligned} \text{KL}(p\|q) &= -\int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(-\int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right) \\ &= -\int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x}. \end{aligned} \quad (2.100)$$

Это выражение называют *относительной энтропией* (*relative entropy*), или *дивергенцией Кульбака–Лейблера* (*Kullback–Leibler divergence*, или *KL divergence*) (Kullback and Leibler, 1951), между распределениями $p(\mathbf{x})$ и $q(\mathbf{x})$. Обратите внимание, что эта величина не является симметричной, т. е. $\text{KL}(p\|q) \neq \text{KL}(q\|p)$.

Теперь приведем доказательство того, что дивергенция Кульбака–Лейблера удовлетворяет условию $\text{KL}(p\|q) \geq 0$ при условии равенства тогда и только тогда, когда $p(\mathbf{x}) = q(\mathbf{x})$. Для этого вначале необходимо ввести определение *выпуклых* (*convex*) функций. Функция $f(x)$ называется выпуклой в том случае, если каждая ее хорда лежит на ней или над ней, как показано на рис. 2.15.

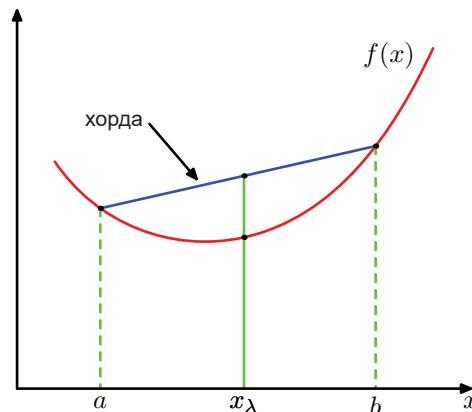


РИС. 2.15 Выпуклой функцией $f(x)$ называется такая, для которой каждая хорда (синий цвет) лежит на функции или над ней (красный цвет)

Любое значение x на интервале от $x = a$ до $x = b$ можно записать в виде $\lambda a + (1 - \lambda)b$, где $0 \leq \lambda \leq 1$. Соответствующая точка на хорде задается выражением $\lambda f(a) + (1 - \lambda)f(b)$, а соответствующее значение функции имеет вид $f(\lambda a + (1 - \lambda)b)$. Отсюда следует, что выпуклость подразумевает

$$f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b). \quad (2.101)$$

Это эквивалентно условию, что вторая производная функции должна быть всюду положительной. Примерами выпуклых функций являются $x \ln x$ (для $x > 0$) и x^2 (см. упражнение 2.32). Функция называется *строго выпуклой* (*strictly convex*), если равенство выполняется только при $\lambda = 0$ и $\lambda = 1$. Если функция обладает противоположным свойством, а именно каждая хорда лежит на или под функцией, то она называется *вогнутой* (*concave*) с соответствующим

определенiem строго вогнутой (*strictly concave*). Если функция $f(x)$ выпуклая, то $-f(x)$ будет вогнутой.

Используя технику доказательства по индукции (см. упражнение 2.33), можно показать из (2.101), что выпуклая функция $f(x)$ удовлетворяет условию

$$f\left(\sum_{i=1}^M \lambda_i x_i\right) \leq \sum_{i=1}^M \lambda_i f(x_i), \quad (2.102)$$

где $\lambda_i \geq 0$ и $\sum_i \lambda_i = 1$, для любого множества точек $\{x_i\}$. Результат (2.102) известен как неравенство Йенсена (*Jensen's inequality*). Если интерпретировать λ_i как распределение вероятностей по дискретной переменной x , принимающей значения $\{x_i\}$, то (2.102) можно записать как

$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)], \quad (2.103)$$

где $\mathbb{E}[\cdot]$ обозначает математическое ожидание. Для непрерывных переменных неравенство Йенсена имеет вид:

$$f\left(\int x p(x) dx\right) \leq \int f(x) p(x) dx. \quad (2.104)$$

Применив неравенство Йенсена в форме (2.104) к дивергенции Кульбака–Лейблера (2.100), можно получить

$$\text{KL}(p \| q) = -\int p(x) \ln \left\{ \frac{q(x)}{p(x)} \right\} dx \geq -\ln \int q(x) dx = 0, \quad (2.105)$$

где используется выпуклая функция $-\ln x$, а также условие нормализации $\int q(x) dx = 1$. На самом деле $-\ln x$ является строго выпуклой функцией, поэтому равенство выполняется тогда и только тогда, когда $q(x) = p(x)$ для всех x . Таким образом, дивергенцию Кульбака–Лейблера можно интерпретировать как меру неподобия двух распределений $p(x)$ и $q(x)$.

Как видно, между сжатием данных и оценкой плотности (т. е. проблемой моделирования неизвестного распределения вероятностей) существует тесная взаимосвязь, поскольку наиболее эффективное сжатие достигается при известном истинном распределении. Если использовать распределение, отличное от истинного, то обязательно получится менее эффективное кодирование, и при этом дополнительная информация, которую необходимо передать, в среднем будет (как минимум) равна расхождению Кульбака–Лейблера между двумя распределениями.

Предположим, что данные генерируются на основе неизвестного распределения $p(x)$, которое необходимо смоделировать. Можно попытаться аппроксимировать это распределение с помощью некоторого параметрического распределения $q(x | \theta)$, управляемого набором настраиваемых параметров θ . Один из способов определить θ – минимизировать расхождение Кульбака–Лейблера между $p(x)$ и $q(x | \theta)$ относительно θ . Напрямую это сделать нельзя, поскольку нам неизвестно $p(x)$. Тем не менее предположим, что мы наблюдаем конечное множество обучающих точек x_n для $n = 1, \dots, N$, взятых из

$p(\mathbf{x})$. Тогда ожидание относительно $p(\mathbf{x})$ можно аппроксимировать конечной суммой по этим точкам с помощью (2.40), так что

$$\text{KL}(p\|q) = \frac{1}{N} \sum_{n=1}^N \{-\ln q(\mathbf{x}_n|\boldsymbol{\theta}) + \ln p(\mathbf{x}_n)\}. \quad (2.106)$$

Второй член в правой части (2.106) не зависит от $\boldsymbol{\theta}$, а первый член – это отрицательная функция логарифмического правдоподобия для $\boldsymbol{\theta}$ при распределении $q(\mathbf{x}|\boldsymbol{\theta})$, оцененном по обучающему набору. Таким образом, минимизация этого расхождения Кульбака–Лейблера эквивалентна максимизации функции логарифмического правдоподобия (см. упражнение 2.34).

2.5.6. Условная энтропия

Теперь рассмотрим совместное распределение между двумя наборами переменных \mathbf{x} и \mathbf{y} , заданное $p(\mathbf{x}, \mathbf{y})$, из которого выбираем пары значений \mathbf{x} и \mathbf{y} . Если значение \mathbf{x} уже известно, то дополнительная информация, необходимая для определения соответствующего значения \mathbf{y} , составляет $-\ln p(\mathbf{y}|\mathbf{x})$. Таким образом, средняя дополнительная информация, необходимая для определения \mathbf{y} , может быть записана в виде

$$H[\mathbf{y}|\mathbf{x}] = -\iint p(\mathbf{y}, \mathbf{x}) \ln p(\mathbf{y}|\mathbf{x}) d\mathbf{y} d\mathbf{x}, \quad (2.107)$$

что называют *условной энтропией* (*conditional entropy*) для \mathbf{y} относительно \mathbf{x} . Из правила произведения (см. упражнение 2.35) можно сделать очевидный вывод, что условная энтропия удовлетворяет соотношению

$$H[\mathbf{x}, \mathbf{y}] = H[\mathbf{y}|\mathbf{x}] + H[\mathbf{x}], \quad (2.108)$$

где $H[\mathbf{x}, \mathbf{y}]$ – это дифференциальная энтропия $p(\mathbf{x}, \mathbf{y})$, а $H[\mathbf{x}]$ – это дифференциальная энтропия маргинального распределения $p(\mathbf{x})$. Таким образом, информация, необходимая для описания \mathbf{x} и \mathbf{y} , определяется суммой информации, необходимой только для описания \mathbf{x} , плюс дополнительная информация, необходимая для определения \mathbf{y} относительно \mathbf{x} .

2.5.7. Взаимная информация

Если две переменные \mathbf{x} и \mathbf{y} независимы, их совместное распределение факторизуется произведением их маргиналов $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$. Если переменные не являются независимыми, можно получить некоторое представление о степени их «близости» к независимости с помощью расхождения Кульбака–Лейблера между совместным распределением и произведением маргиналов, которое определяется формулой

$$\begin{aligned} I[\mathbf{x}, \mathbf{y}] &\equiv \text{KL}(p(\mathbf{x}, \mathbf{y}) \| p(\mathbf{x})p(\mathbf{y})) \\ &= -\iint p(\mathbf{x}, \mathbf{y}) \ln \left(\frac{p(\mathbf{x})p(\mathbf{y})}{p(\mathbf{x}, \mathbf{y})} \right) d\mathbf{x} d\mathbf{y}, \end{aligned} \quad (2.109)$$

что называют *взаимной информацией* (*mutual information*) между переменными x и y . Из свойств дивергенции Кульбака–Лейблера следует, что $I[x, y] > 0$ справедливо тогда и только тогда, когда x и y независимы. Используя правила суммы и произведения вероятностей, можно убедиться, что взаимная информация связана с условной энтропией (см. упражнение 2.38) через

$$I[x, y] = H[x] - H[x|y] = H[y] - H[y|x]. \quad (2.110)$$

Таким образом, взаимная информация представляет собой уменьшение неопределенности относительно x в результате получения информации о значении y (или наоборот). С позиции теории Байеса можно рассматривать $p(x)$ как апостериорное распределение для x , а $p(x|y)$ – как апостериорное распределение после наблюдения новых данных y . Получается, что взаимная информация представляет собой уменьшение неопределенности в отношении x в результате нового наблюдения y .

2.6. Байесовские вероятности

Когда мы изучали согнутую монету на рис. 2.2, было введено определение вероятности в контексте периодичности наступления случайных, повторяющихся событий, таких как вероятность выпадения монеты вогнутой стороной вверх. Мы будем называть это классической (classical) или частотной (frequentist) интерпретацией вероятности. Кроме того, была введена более общая байесовская точка зрения, в которой вероятности представляют собой количественную оценку неопределенности. В данном случае эта неопределенность заключается в оценке того, является ли вогнутая сторона монеты орлом или решкой.

Использование вероятности для представления неопределенности – не просто случайный выбор, а неизбежность, позволяющая нам руководствоваться правилами здравого смысла и делать рациональные, последовательные умозаключения. Например, в работе (Cox, 1946) было показано, что если числовые значения используются для представления степеней вероятности, то простой набор аксиом, кодирующих свойства практического смысла таких убеждений, однозначно приводит к набору правил для манипулирования степенями вероятности, которые эквивалентны правилам суммы и произведения вероятностей. Поэтому естественно называть эти величины (байесовскими) вероятностями.

В случае с изогнутой монетой мы предположили, что вероятность выпадения вогнутой стороны монеты равна 0,5. Теперь предположим, что нам сообщили результаты нескольких подбрасываний монеты. Интуитивно кажется, что это должно дать нам некоторую информацию о том, выпадет ли вогнутая сторона монеты вверху. Например, предположим, что мы видим гораздо больше случаев, когда выпадает решка, нежели орел. Если учесть, что монета с большей вероятностью упадет вогнутой стороной вверх, это дает

основания предположить, что вогнутая сторона с большей вероятностью окажется решкой. На самом деле эта интуиция верна, и более того (см. упражнение 2.40), мы можем количественно оценить это с помощью правил вероятности. Теперь теорема Байеса приобретает новое значение, поскольку она позволяет нам преобразовать априорную вероятность того, что вогнутая сторона окажется орлом, в последующую вероятность, учитывая данные, полученные в результате подбрасывания монет. Более того, этот процесс является итеративным, т. е. апостериорная вероятность становится априорной для включения данных, полученных в ходе дальнейших подбрасываний монет.

Один из аспектов байесовского подхода заключается в том, что включение предварительных знаний происходит естественным образом. Предположим, например, что обычная монета подбрасывается три раза и каждый раз выпадает орел. Оценка вероятности выпадения орлов с максимальным правдоподобием будет равна 1 (см. раздел 3.1.2), а это означает, что все последующие подбрасывания будут заканчиваться выпадением орлов! В отличие от этого байесовский подход с любым разумным априорным значением приведет к менее экстремальному выводу.

2.6.1. Параметры модели

Байесовская перспектива помогает лучше понять некоторые аспекты машинного обучения (см. раздел 1.2), что можно проиллюстрировать на примере регрессии синусоидальной кривой. В данном случае обучающий набор данных обозначен как \mathcal{D} . На примере линейной регрессии уже было показано, что параметры могут быть выбраны с помощью метода максимального правдоподобия, при котором w принимает значение, максимизирующее функцию правдоподобия $p(\mathcal{D} | w)$. Это соответствует выбору значения w , для которого максимально вероятен наблюдаемый набор данных. В литературе по машинному обучению отрицательный логарифм функции правдоподобия называется функцией ошибки (*error function*). Поскольку отрицательный логарифм является монотонно убывающей функцией, максимизация правдоподобия эквивалентна минимизации ошибки. Это приводит к выбору конкретных значений параметров, обозначаемых w_{ML} , которые затем используются для прогнозирования новых данных.

Выше было показано, что различные варианты набора обучающих данных, например содержащие разное количество точек данных, приводят к различным результатам для w_{ML} . Исходя из байесовской перспективы, для описания этой неопределенности в параметрах модели также можно использовать механизмы теории вероятностей. Предположения о w , сделанные до наблюдения за данными, можно выразить в виде априорного распределения вероятности $p(w)$. Влияние наблюдаемых данных \mathcal{D} выражается через функцию правдоподобия $p(\mathcal{D} | w)$, и теорема Байеса теперь принимает вид

$$p(w | \mathcal{D}) = \frac{p(\mathcal{D} | w)p(w)}{p(\mathcal{D})}, \quad (2.111)$$

что позволяет оценить неопределенность в w после наблюдения \mathcal{D} в виде апостериорной вероятности $p(w|\mathcal{D})$.

Важно подчеркнуть, что величина $p(\mathcal{D}|w)$ называется функцией правдоподобия, когда она рассматривается как функция вектора параметров w , и это выражение показывает, насколько вероятен наблюдаемый набор данных для различных значений w . Обратите внимание, что вероятность $p(\mathcal{D}|w)$ не является распределением вероятностей по w , и ее интеграл по w не (обязательно) равен единице.

С учетом этого определения правдоподобия можно сформулировать теорему Байеса следующим образом:

$$\begin{aligned} \text{posterior (апостериорное)} &\propto \text{likelihood (вероятность)} \\ &\times \text{prior (априорное),} \end{aligned} \quad (2.112)$$

где все эти величины рассматриваются как функции от w . Знаменатель в (2.111) – это константа нормализации, которая гарантирует, что апостериорное распределение в левой части является действительной плотностью распределения вероятностей и интегрируется в единицу. Действительно, интегрируя обе стороны (2.111) по w , можно выразить знаменатель в теореме Байеса в виде предшествующего распределения и функции правдоподобия:

$$p(\mathcal{D}) = \int p(\mathcal{D}|w)p(w)dw. \quad (2.113)$$

Как в байесовской, так и в частотной модели центральную роль играет функция правдоподобия $p(\mathcal{D}|w)$. Однако способы ее использования в этих двух парадигмах принципиально отличаются. В случае частотного метода w рассматривается как фиксированный параметр, значение которого определяется некоторой «оценкой», а границы погрешности этой оценки определяются (по крайней мере, концептуально) путем оценки распределения возможных наборов данных \mathcal{D} . В отличие от этого в байесовской концепции существует только один набор данных \mathcal{D} (т. е. реально наблюдаемый), а неопределенность параметров выражается через распределение вероятности по w .

2.6.2. Регуляризация

С помощью этой байесовской перспективы можно получить представление о технике регуляризации (см. раздел 1.2.5), которая использовалась в примере с регрессией синусоидальной кривой для уменьшения избыточной подгонки. Вместо вычисления параметров модели путем максимизации функции правдоподобия по отношению к w можно максимизировать апостериорную вероятность (2.111). Эта техника называется *максимальной апостериорной* (*maximum a posteriori*) оценкой, или просто МАР-оценкой. Аналогичным образом можно минимизировать отрицательный логарифм апостериорной вероятности. Взяв отрицательные логарифмы обеих сторон (2.111), получаем:

$$-\ln p(w|\mathcal{D}) = -\ln p(\mathcal{D}|w) - \ln p(w) + \ln p(\mathcal{D}). \quad (2.114)$$

Первый член в правой части (2.114) – это обычное логарифмическое правдоподобие. Третий член можно опустить, так как он не зависит от w . Второй член имеет вид функции от w , которая добавляется к логарифмическому правдоподобию, и его можно рассматривать как форму регуляризации. Чтобы сделать это более явным, выберем априорное распределение $p(w)$ как произведение независимых нулевых средних гауссовых распределений для всех элементов w , каждое из которых имеет одинаковую дисперсию s^2 , так что

$$p(w|s) = \prod_{i=0}^M \mathcal{N}(w_i|0, s^2) = \prod_{i=0}^M \left(\frac{1}{2\pi s^2} \right)^{1/2} \exp \left\{ -\frac{w_i^2}{2s^2} \right\}. \quad (2.115)$$

Подставляя в (2.114), получаем:

$$-\ln p(w|\mathcal{D}) = -\ln p(\mathcal{D}|w) + \frac{1}{2s^2} \sum_{i=0}^M w_i^2 + \text{константа}. \quad (2.116)$$

Если рассматривать частный случай линейной регрессионной модели, логарифмическое правдоподобие которой задается в (2.66), то окажется, что максимизация апостериорного распределения эквивалентна (см. упражнение 2.41) минимизации функции:

$$E(w) = \frac{1}{2\sigma^2} \sum_{n=1}^N \{y(x_n, w) - t_n\}^2 + \frac{1}{2s^2} w^\top w. \quad (2.117)$$

Как видно, это имеет вид регуляризованной функции ошибки суммы квадратов, встречавшейся ранее в форме (1.4).

2.6.3. Байесовское машинное обучение

Байесовская методика позволяет обосновать использование регуляризации и вывести конкретную форму регуляризационного члена. Однако использование одной лишь теоремы Байеса не является по-настоящему байесовским подходом к машинному обучению, поскольку в этом случае все равно находится единственное решение для w и не учитывается неопределенность в значении w . Предположим, у нас есть обучающий набор данных \mathcal{D} и наша цель – предсказать некоторую целевую переменную t при новом входном значении x . Таким образом, нас интересует распределение t относительно как x , так и \mathcal{D} . Исходя из правил суммы и произведения вероятностей, получаем:

$$p(t|x, \mathcal{D}) = \int p(t|x, w)p(w|\mathcal{D})dw. \quad (2.118)$$

Как видно, прогноз получается путем взятия средневзвешенного значения $p(t|x, w)$ по всем возможным значениям w , где весовая функция задается апостериорным распределением вероятности $p(w|\mathcal{D})$. Ключевым отличием байесовских методов является интегрирование по пространству параметров. В отличие от этого обычные частотные методы используют точечные оценки

параметров, полученные путем оптимизации функции потерь, например регуляризованной суммы квадратов.

Такая полностью байесовская интерпретация машинного обучения позволяет сделать ряд важных выводов. Например, проблема чрезмерной подгонки, с которой пришлось столкнуться ранее в контексте полиномиальной регрессии (см. раздел 1.2), является примером патологии, возникающей при использовании максимального правдоподобия, и не возникает при маргинализации параметров с помощью байесовского подхода. Аналогично можно иметь несколько потенциальных моделей, которые могут быть использованы для решения данной задачи, например полиномы разных порядков в примере с регрессией. При подходе с максимальным правдоподобием просто выбирается модель, которая дает наибольшую вероятность для имеющихся данных, но при этом предпочтение отдается более сложным моделям, что приводит к чрезмерной подгонке. Полностью байесовский метод предполагает усреднение по всем потенциальным моделям (см. раздел 9.6), при этом влияние каждой модели оценивается по ее апостериорной вероятности. Более того, эта вероятность обычно максимальна для моделей промежуточной сложности. Очень простые модели (например, полиномы невысокого порядка) имеют низкую вероятность, поскольку они неспособны эффективно соответствовать данным, в то время как очень сложные модели (например, полиномы очень высокого порядка) также имеют низкую вероятность, поскольку байесовское интегрирование по параметрам автоматически и elegantly влечет компенсацию за сложность. Полный обзор байесовских методов, применяемых в машинном обучении, включая нейронные сети, см. в книге (Bishop, 2006).

К сожалению, у байесовской системы есть существенный недостаток, который проявляется в (2.118), где требуется интегрирование по пространству параметров. Современные модели глубокого обучения способны насчитывать миллионы или миллиарды параметров, и даже простые приближения для подобных интегралов, как правило, невыполнимы. На самом деле при ограниченном вычислительном бюджете и наличии достаточного количества обучающих данных зачастую лучше применить методы максимального правдоподобия, обычно дополненные одной или несколькими формами регуляризации, к большой нейронной сети, чем применять байесовскую обработку для значительно меньшей модели.

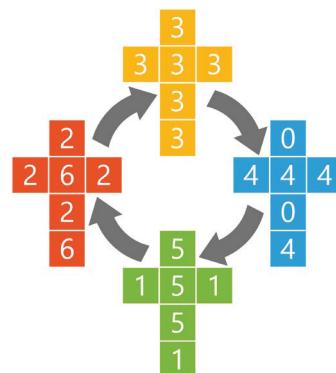
Упражнения

- 2.1** (*) В примере с диагностикой онкологических заболеваний использовалась априорная вероятность возникновения рака, равная $p(C = 1) = 0,01$. В реальности распространенность онкологии обычно намного ниже. Рассмотрим ситуацию, в которой $p(C = 1) = 0,001$, и пересчитаем вероятность наличия рака при положительном тесте $p(C = 1 | T = 1)$. Результаты могут показаться многим людям удивительными, поскольку тест, каза-

лось бы, обладает высокой точностью, но положительный результат все равно приводит к небольшой вероятности заболеть раком.

- 2.2** (***) Детерминированные числа удовлетворяют свойству *транзитивности* (*transitivity*), так что если $x > y$ и $y > z$, то из этого следует, что $x > z$. Однако при обращении к случайным числам это свойство больше не требуется. На рис. 2.16 показан набор из четырех кубических игральных костей, расположенных в циклическом порядке. Докажите, что вероятность выпадения на каждой из четырех костей на $2/3$ выше, чем на предыдущей в цикле. Такие кости называются *нетранзитивными* (*non-transitive*), а приведенные здесь в конкретном примере – *костями Эфрана* (*Efron dice*).

РИС. 2.16 Пример непереходных кубических костей, где каждая из них «развернута», чтобы показать число на каждой из граней. Кости расположены по циклу таким образом, что вероятность того, что на каждой кости выпадет большее число, чем на предыдущей в цикле, составляет $2/3$



- 2.3** (*) Пусть переменная y задана суммой двух независимых случайных величин $y = u + v$, где $u \sim p_u(u)$, а $v \sim p_v(v)$. Докажите, что распределение $p_y(y)$ задается как

$$p(y) = \int p_u(u)p_v(y - u)du. \quad (2.119)$$

Это называется *сверткой* (*convolution*) величин $p_u(u)$ и $p_v(v)$.

- 2.4** (***) Подтвердите, что равномерное распределение (2.33) правильно нормализовано, и найдите выражения для его среднего значения и дисперсии.
- 2.5** (***) Убедитесь, что экспоненциальное распределение (2.34) и распределение Лапласа (2.35) нормализованы правильно.
- 2.6** (*) Используя свойства дельта-функции Дирака, докажите, что эмпирическая плотность (2.37) нормализована правильно.
- 2.7** (*) Используя эмпирическую плотность (2.37), докажите, что математическое ожидание, заданное в (2.39), может быть аппроксимировано суммой по конечному набору выборок, взятых из плотности в результате (2.40).

- 2.8** (*) Используя определение (2.44), докажите, что $\text{var}[f(x)]$ удовлетворяет (2.45).
- 2.9** (*) Докажите, что если две переменные x и y независимы, то их ковариация равна нулю.
- 2.10** (*) Предположим, что две переменные x и z статистически независимы. Докажите, что среднее значение и дисперсия их суммы удовлетворяют условиям:

$$\mathbb{E}[x + z] = \mathbb{E}[x] + \mathbb{E}[z], \quad (2.120)$$

$$\text{var}[x + z] = \text{var}[x] + \text{var}[z]. \quad (2.121)$$

- 2.11** (*) Пусть две переменные x и y имеют совместное распределение $p(x, y)$. Докажите следующие два следствия:

$$\mathbb{E}[x] = \mathbb{E}_y[\mathbb{E}_x[x|y]], \quad (2.122)$$

$$\text{var}[x] = \mathbb{E}_y[\text{var}_x[x|y]] + \text{var}_y[\mathbb{E}_x[x|y]]. \quad (2.123)$$

Здесь $\mathbb{E}_x[x|y]$ обозначает ожидание x при условном распределении $p(x|y)$ с аналогичным обозначением для условной дисперсии.

- 2.12** (***) В этом упражнении доказывается условие нормализации (2.51) для одномерных гауссиан. Для этого рассмотрим интеграл

$$I = \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2\sigma^2}x^2\right) dx, \quad (2.124)$$

который можно вычислить, записав для начала его возведение в квадрат в виде

$$I^2 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}y^2\right) dx dy. \quad (2.125)$$

Теперь выполним преобразование из декартовых координат (x, y) в полярные (r, θ) , а затем подставим $u = r^2$. Докажите, что, если взять интегралы по θ и u , а затем извлечь квадратный корень из обеих сторон, то получится

$$I = (2\pi\sigma^2)^{1/2}. \quad (2.126)$$

Наконец, используйте этот результат, чтобы показать, что гауссово распределение $\mathcal{N}(x|\mu, \sigma^2)$ нормализовано.

- 2.13** (**) С помощью замены переменных определите, что одномерное гауссово распределение, заданное в (2.49), удовлетворяет условиям (2.52). Далее, продифференцировав обе стороны условия нормализации

$$\int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) dx = 1 \quad (2.127)$$

по отношению к σ^2 , убедитесь, что гауссово распределение удовлетворяет условиям (2.53). Наконец, докажите, что (2.54) выполняется.

- 2.14** (*) Докажите, что мода (т. е. максимум) гауссова распределения (2.49) задается значением μ .
- 2.15** (*) Задав производные функции логарифмического правдоподобия в (2.56) по μ и σ^2 равными нулю, проверьте результаты (2.57) и (2.58).
- 2.16** (**) Используя результаты (2.52) и (2.53), покажите, что

$$\mathbb{E}[x_n x_m] = \mu^2 + I_{nm} \sigma^2, \quad (2.128)$$

где x_n и x_m обозначают точки данных, выбранные из гауссова распределения со средним значением μ и дисперсией σ^2 , а I_{nm} удовлетворяет условию $I_{nn} = 1$, если $n = m$, и в противном случае $I_{nm} = 0$. Отсюда вытекают доказательства результатов (2.59) и (2.60).

- 2.17** (**) Используя определение (2.61), докажите результат (2.62), который показывает, что ожидание оценки дисперсии для гауссова распределения, основанной на истинном среднем значении, определяется истинной дисперсией σ^2 .
- 2.18** (*) Докажите, что максимизация (2.66) по отношению к σ^2 приводит к результату (2.68).
- 2.19** (**) Используйте свойство преобразования (2.71) плотности распределения вероятностей при изменении переменной, чтобы доказать, что любая плотность $p(y)$ может быть получена из фиксированной и везде ненулевой плотности $q(x)$ путем нелинейной замены переменной $y = f(x)$, в которой $f(x)$ – это монотонная функция, удовлетворяющая условию $0 \leq f'(x) < \infty$. Запишите дифференциальное уравнение, которому удовлетворяет $f(x)$, и постройте диаграмму, иллюстрирующую преобразование плотности.
- 2.20** (*) Вычислите элементы матрицы Якоби для преобразования, определяемого (2.78) и (2.79).
- 2.21** (*) В разделе 2.5 мы ввели понятие энтропии $h(x)$ как информации, получаемой при наблюдении значения случайной величины x с распределением $p(x)$. Было показано, что для независимых переменных x и y , для которых $p(x, y) = p(x)p(y)$, функции энтропии аддитивны, так что $h(x, y) = h(x) + h(y)$. В этом упражнении необходимо определить связь между h и p в виде функции $h(p)$. Сначала докажите, что $h(p^2) = 2h(p)$ и, следовательно, по индукции что $h(p^n) = nh(p)$, где n – целое положительное число. Отсюда следует, что $h(p^{n/m}) = (n/m)h(p)$, где m – тоже целое положительное число. Это значит, что $h(p^x) = xh(p)$, где x – положительное рациональное число и, следовательно, в силу непрерывности, положительное действительное число. Наконец, докажите, что это предполагает, что $h(p)$ должно иметь вид $h(p) \propto \ln p$.

- 2.22** (*) С помощью множителя Лагранжа докажите, что максимизация энтропии (2.86) для дискретной переменной дает распределение, в котором все вероятности $p(x_i)$ равны, и что соответствующее значение энтропии равно $\ln M$.
- 2.23** (*) Возьмите дискретную случайную переменную x с M состояниями и используйте неравенство Йенсена в форме (2.102) для доказательства того, что энтропия ее распределения $p(x)$ удовлетворяет условию $H[x] \leq \ln M$.
- 2.24** (**) Используйте вариационное исчисление для доказательства того, что стационарная точка функционала (2.96) задается в (2.97). Затем используйте ограничения (2.93), (2.94) и (2.95) для устранения множителей Лагранжа и, таким образом, докажите, что решение с максимальной энтропией задается гауссовым распределением (2.98).
- 2.25** (*) Используйте результаты (2.94) и (2.95), чтобы доказать, что энтропия одномерного гауссова распределения (2.98) задается выражением (2.99).
- 2.26** (**) Предположим, что $p(\mathbf{x})$ – некоторое фиксированное распределение, и его необходимо аппроксимировать с помощью гауссова распределения $q(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Записав форму расхождения Кульбака–Лейблера $KL(p || q)$ для гауссова распределения $q(\mathbf{x})$ и далее дифференцируя, докажите, что минимизация $KL(p || q)$ относительно $\boldsymbol{\mu}$ и $\boldsymbol{\Sigma}$ приводит к тому, что $\boldsymbol{\mu}$ задается ожиданием \mathbf{x} при $p(\mathbf{x})$, а $\boldsymbol{\Sigma}$ – ковариацией.
- 2.27** (**) Вычислите расхождение Кульбака–Лейблера (2.100) между двумя гауссовыми распределениями $p(x) = \mathcal{N}(x | \mu, \sigma^2)$ и $q(x) = \mathcal{N}(x | m, s^2)$.
- 2.28** (*) Семейство альфа-дивергенций определяется как
- $$D_\alpha(p || q) = \frac{4}{1 - \alpha^2} \left(1 - \int p(x)^{(1+\alpha)/2} q(x)^{(1-\alpha)/2} dx \right), \quad (2.129)$$
- где $-\infty < \alpha < \infty$ – непрерывный параметр. Докажите, что расходимость Кульбака–Лейблера $KL(p || q)$ соответствует $\alpha \rightarrow 1$. Это можно сделать, записав $p^\epsilon = \exp(\epsilon \ln p) = 1 + \ln p + O(\epsilon^2)$ и затем приняв $\epsilon \rightarrow 0$. Аналогичным образом докажите, что $KL(q || p)$ удовлетворяет $\alpha \rightarrow -1$.
- 2.29** (**) Рассмотрим две переменные \mathbf{x} и \mathbf{y} , имеющие совместное распределение $p(\mathbf{x}, \mathbf{y})$. Докажите, что дифференциальная энтропия этой пары переменных удовлетворяет условию
- $$H[\mathbf{x}, \mathbf{y}] \leq H[\mathbf{x}] + H[\mathbf{y}] \quad (2.130)$$
- при условии равенства тогда и только тогда, когда \mathbf{x} и \mathbf{y} статистически независимы.
- 2.30** (*) Рассмотрим вектор \mathbf{x} непрерывных переменных с распределением $p(\mathbf{x})$ и соответствующей энтропией $H[\mathbf{x}]$. Допустим, что мы производим

несингулярное линейное преобразование x с целью получения новой переменной $y = Ax$. Докажите, что соответствующая энтропия задается $H[y] = H[x] + \ln \det A$, где $\det A$ обозначает детерминант A .

- 2.31** (***) Предположим, что условная энтропия $H[y|x]$ между двумя дискретными случайными величинами x и y равна нулю. Докажите, что для всех значений x , таких что $p(x) > 0$, переменная y должна быть функцией x . Другими словами, для каждого x существует только одно значение y , такое что $p(y|x) \neq 0$.
- 2.32** (*) Строго выпуклой считается такая функция, для которой каждая хорда лежит выше функции. Покажите, что это эквивалентно условию, при котором вторая производная функции положительна.
- 2.33** (**) С помощью доказательства методом индукции докажите, что из неравенства (2.101) для выпуклых функций вытекает результат (2.102).
- 2.34** (*) Докажите, что расхождение Кульбака–Лейблера (2.100) между эмпирическим распределением (2.37) и модельным распределением $q(x|\theta)$ равно отрицательной функции логарифмического правдоподобия с точностью до аддитивной константы.
- 2.35** (*) Используя определение (2.107) и правило произведения вероятностей, докажите результат (2.108).
- 2.36** (****) Рассмотрим две бинарные переменные x и y с совместным распределением, заданным

		y
	0	0 1
x	0	1/3 1/3
	1	0 1/3

Оцените следующие величины:

- (a) $H[x]$; (c) $H[y|x]$; (e) $H[x, y]$;
 (b) $H[y]$; (d) $H[x|y]$; (f) $I[x, y]$.

Нарисуйте диаграмму Венна, чтобы показать связь между этими различными величинами.

- 2.37** (*) Применяя неравенство Йенсена (2.102) с $f(x) = \ln x$, докажите, что среднее арифметическое значение набора действительных чисел никогда не бывает меньше их среднего геометрического.
- 2.38** (*) Используя правила суммы и произведения вероятностей, докажите, что взаимная информация $I(x, y)$ будет удовлетворять соотношению (2.110).
- 2.39** (**) Предположим, что две переменные z_1 и z_2 независимы, так что $p(z_1, z_2) = p(z_1)p(z_2)$. Докажите, что ковариационная матрица между этими переменными диагональная. Из этого следует, что независимость

является достаточным условием для того, чтобы две переменные были некоррелированными. Теперь рассмотрим две переменные y_1 и y_2 , где y_1 симметрично распределена относительно 0, а $y_2 = y_1^2$. Запишите условное распределение $p(y_2|y_1)$ и убедитесь, что оно зависит от y_1 , тем самым доказывая, что эти две переменные не являются независимыми. Теперь покажите, что ковариационная матрица между этими двумя переменными снова диагональная. Для этого используйте соотношение $p(y_1, y_2) = p(y_1)p(y_2|y_1)$, чтобы убедиться, что недиагональные члены равны нулю. Этот контрпример показывает, что нулевая корреляция не является достаточным условием независимости.

- 2.40** (*) Рассмотрим изогнутую монету на рис. 2.2. Предположим, что априорная вероятность того, что выпуклая сторона окажется орлом, равна 0,1. Теперь предположим, что монету подбросили 10 раз и сказали, что восемь из них выпали орлом вверх, а два – решкой вверх. Используйте теорему Байеса для определения апостериорной вероятности того, что вогнутая сторона будет орлом. Вычислите вероятность того, что при следующем подбрасывании выпадет орел.
- 2.41** (*) Подставив (2.115) в (2.114) и воспользовавшись результатом (2.66) для логарифмического правдоподобия модели линейной регрессии, выведите результат (2.117) для регуляризованной функции ошибки.

Глава 3

Стандартные распределения

В этой главе рассматриваются различные примеры распределений вероятностей и их свойства. Помимо того, что эти распределения сами по себе представляют определенный интерес, они также могут служить структурными элементами для более сложных моделей и будут часто встречаться в этой книге.

Одним из способов применения распределений, рассматриваемых в этой главе, является моделирование распределения вероятностей $p(\mathbf{x})$ случайной величины \mathbf{x} , заданной конечным набором наблюдений $\mathbf{x}_1, \dots, \mathbf{x}_N$. Эта задача известна как *оценка плотности распределения* (*density estimation*). Следует подчеркнуть, что проблема оценки плотности является принципиально неразрешимой, поскольку существует бесконечно много распределений вероятностей, которые могли бы привести к наблюдаемому конечному набору данных. Действительно, возможным вариантом является любое ненулевое распределение $p(\mathbf{x})$ в каждой из точек данных $\mathbf{x}_1, \dots, \mathbf{x}_N$. Вопрос выбора подходящего распределения связан с задачей выбора модели, которая уже упоминалась ранее в контексте подгонки полиномиальных кривых. Эта задача является одной из ключевых в машинном обучении (см. раздел 1.2).

Для начала рассмотрим распределения для дискретных переменных, после чего познакомимся с распределением Гаусса для непрерывных переменных. Они представляют собой конкретные примеры *параметрических* (*parametric*) распределений, названных так потому, что они задаются относительно небольшим числом настраиваемых параметров, таких как среднее значение и дисперсия гауссова распределения. Чтобы применять такие модели для решения задачи оценки плотности, понадобится процедура определения подходящих значений параметров при наличии наблюдаемого набора данных, и основное внимание будет уделено максимизации функции правдоподобия. В этой главе подразумевается, что данные наблюдений являются независимыми и одинаково распределенными (*independent and identically distributed*, *i.i.d.*), в то время как в последующих главах рассматриваются более сложные случаи со *структурированными данными* (*structured data*), где это условие уже не выполняется.

Одним из ограничений параметрического метода является допущение определенной функциональной формы распределения, которая может оказаться неприемлемой для конкретных задач. Альтернативным подходом являются *непараметрические* (*nonparametric*) методы оценки плотности распределения, когда форма распределения в основном определяется размером набора данных. Такие модели все равно содержат параметры, но они скорее управляют сложностью модели, чем формой распределения. В конце этой главы приведены три непараметрических метода, основанных, соответственно, на гистограммах, близлежащих соседних элементах и ядрах. Основное ограничение подобных непараметрических методов заключается в необходимости хранения всех обучающих данных. Другими словами, количество параметров растет с увеличением размера набора данных, поэтому для больших наборов данных этот метод становится крайне непригодным. Глубокое обучение сочетает в себе высокую эффективность параметрических моделей и общность непараметрических методов, поскольку рассматривает гибкие распределения на основе нейронных сетей с большим, но фиксированным числом параметров.

3.1. Дискретные переменные

Для начала рассмотрим простые распределения для дискретных переменных, от бинарных до переменных с несколькими состояниями.

3.1.1. Распределение Бернулли

Рассмотрим отдельную бинарную случайную величину $x \in \{0, 1\}$. К примеру, x может описывать результат подбрасывания монеты, при этом $x = 1$ означает «орел», а $x = 0$ означает «решка». Если монета повреждена, как показано на рис. 2.2, вероятность выпадения орла не всегда будет совпадать с вероятностью выпадения решки.

Вероятность выпадения $x = 1$ будет обозначаться параметром μ , так что

$$p(x = 1 | \mu) = \mu, \quad (3.1)$$

где $0 \leq \mu \leq 1$, откуда следует, что $p(x = 0 | \mu) = 1 - \mu$. Поэтому распределение вероятностей по x можно записать выражением

$$\text{Bern}(x | \mu) = \mu^x(1 - \mu)^{1-x}, \quad (3.2)$$

которое известно как *распределение Бернулли* (*Bernoulli distribution*) (см. упражнение 3.1). Несложно убедиться, что это распределение нормализовано, а его среднее значение и дисперсия определяются как

$$\mathbb{E}[x] = \mu, \quad (3.3)$$

$$\text{var}[x] = \mu(1 - \mu). \quad (3.4)$$

Теперь предположим, что у нас есть набор данных $\mathcal{D} = \{x_1, \dots, x_N\}$ из наблюдавшихся значений x . Можно построить функцию правдоподобия, которая является функцией μ , в предположении, что наблюдения взяты независимо от $p(x|\mu)$, так что

$$p(\mathcal{D}|\mu) = \prod_{n=1}^N p(x_n|\mu) = \prod_{n=1}^N \mu^{x_n} (1-\mu)^{1-x_n}. \quad (3.5)$$

Значение μ можно оценить путем максимизации функции правдоподобия или, что равнозначно, путем максимизации логарифма правдоподобия, поскольку логарифм является монотонной функцией. Логарифмическая функция правдоподобия для распределения Бернулли имеет вид:

$$\ln p(\mathcal{D}|\mu) = \sum_{n=1}^N \ln p(x_n|\mu) = \sum_{n=1}^N \{x_n \ln \mu + (1-x_n) \ln(1-\mu)\}. \quad (3.6)$$

Обратите внимание, что функция логарифмического правдоподобия зависит от N наблюдений x_n только через их сумму $\sum_n x_n$. Эта сумма является примером *достаточной статистики* (*sufficient statistic*) для данных при таком распределении (см. раздел 3.4). Если определить производную $\ln p(\mathcal{D}|\mu)$ по μ равной нулю, получится оценка максимального правдоподобия:

$$\mu_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N x_n, \quad (3.7)$$

которая также известна как *выборочное среднее значение* (*sample mean*). Обозначив число наблюдений $x=1$ (орлов) в этом наборе данных через m , можно записать (3.7) в виде

$$\mu_{\text{ML}} = \frac{m}{N}. \quad (3.8)$$

Таким образом, в этой системе максимального правдоподобия вероятность выпадения орлов определяется количеством наблюдений орлов в наборе данных.

3.1.2. Биноминальное распределение

Также можно вычислить распределение для бинарной переменной x по числу m наблюдений $x=1$, с учетом того что размер набора данных составляет N . Это называется биномиальным распределением (*binomial distribution*), и, как следует из (3.5), оно прямо пропорционально $\mu^m (1-\mu)^{N-m}$. При получении коэффициента нормализации необходимо учесть, что из N подбрасываний монеты необходимо сложить все возможные способы получения m орлов, так что биномиальное распределение можно записать в виде

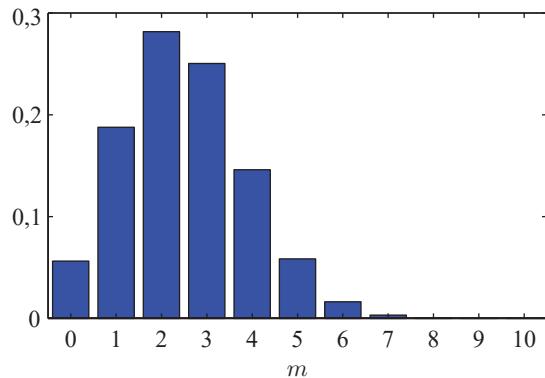
$$\text{Bin}(m|N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m}, \quad (3.9)$$

где

$$\binom{N}{m} \equiv \frac{N!}{(N-m)!m!} \quad (3.10)$$

означает количество способов выбрать m объектов из N одинаковых объектов без замены. На рис. 3.1 показан график биномиального распределения для $N = 10$ и $\mu = 0,25$ (см. упражнение 3.3).

РИС. 3.1 Гистограмма биномиального распределения (3.9) в зависимости от m



Среднее значение и дисперсию биномиального распределения можно определить исходя из того, что для независимых событий среднее значение суммы равно сумме средних значений, а дисперсия суммы равна сумме дисперсий, как показано в упражнении 2.10. Поскольку $m = x_1 + \dots + x_N$ и поскольку для каждого наблюдения среднее значение и дисперсия заданы соответственно (3.3) и (3.4), получаем:

$$\mathbb{E}[m] \equiv \sum_{m=0}^N m \text{Bin}(m|N, \mu) = N\mu, \quad (3.11)$$

$$\text{var}[m] \equiv \sum_{m=0}^N (m - \mathbb{E}[m])^2 \text{Bin}(m|N, \mu) = N\mu(1 - \mu). \quad (3.12)$$

Эти результаты можно также доказать напрямую с помощью вычислений (см. упражнение 3.4).

3.1.3. Полиномиальное распределение

Бинарные переменные служат для описания величин, которые могут принимать одно из двух допустимых значений. Однако часто приходится сталкиваться с дискретными переменными, которые могут принимать одно из K возможных взаимоисключающих состояний. Хотя существуют различные альтернативные способы выражения таких переменных, далее будет показано, что особенно удобным представлением является схема «1 из K », иногда

называемая «одноточечным» (а также «прямым» или «одноразовым») кодированием (one-hot encoding), когда переменная представляется К-мерным вектором \mathbf{x} с одним из элементов x_k , равным 1, а все остальные элементы равны 0. Так, например, если есть переменная, которая может принимать $K = 6$ состояний, и конкретное наблюдение переменной соответствует состоянию, когда $x_3 = 1$, то \mathbf{x} будет представлено в виде

$$\mathbf{x} = (0, 0, 1, 0, 0, 0)^T. \quad (3.13)$$

Обратите внимание, что подобные векторы удовлетворяют условию $\sum_{k=1}^K x_k = 1$. Если обозначить вероятность $x_k = 1$ через параметр μ_k , то распределение \mathbf{x} будет иметь вид

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^K \mu_k^{x_k}, \quad (3.14)$$

где $\boldsymbol{\mu} = (\mu_1, \dots, \mu_K)^T$, а параметры μ_k имеют ограничения, удовлетворяющие условиям $\mu_k > 0$ и $\sum_k \mu_k = 1$, поскольку они представляют собой вероятности. Распределение (3.14) можно рассматривать как обобщение распределения Бернулли на более чем два результата. Несложно заметить, что распределение нормализовано:

$$\sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu}) = \sum_{\mathbf{x}} \prod_{k=1}^K \mu_k^{x_k} = 1, \quad (3.15)$$

и что

$$\mathbb{E}[\mathbf{x}|\boldsymbol{\mu}] = \sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu}) \mathbf{x} = \boldsymbol{\mu}. \quad (3.16)$$

Теперь рассмотрим набор данных \mathcal{D} из N независимых наблюдений x_1, \dots, x_N . Соответствующая функция правдоподобия имеет вид

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{n=1}^N \prod_{k=1}^K \mu_k^{x_{nk}} = \prod_{k=1}^K \mu_k^{\left(\sum_n x_{nk}\right)} = \prod_{k=1}^K \mu_k^{m_k}, \quad (3.17)$$

где видно, что функция правдоподобия определяется N точками данных только посредством K величин:

$$m_k = \sum_{n=1}^N x_{nk}, \quad (3.18)$$

которые представляют собой число наблюдений $x_k = 1$. Эти величины называются *достаточными статистиками* (*sufficient statistics*) для этого распределения (см. раздел 3.4). Следует отметить, что для переменных m_k действует ограничение:

$$\sum_{k=1}^K m_k = N. \quad (3.19)$$

Чтобы найти значение максимального правдоподобия для μ , необходимо максимизировать $\ln p(\mathcal{D} | \mu)$ для μ_k с учетом ограничения (3.15), согласно которому сумма μ_k должна равняться единице. Для этого можно использовать множитель Лагранжа λ (см. приложение С) и получить максимизацию:

$$\sum_{k=1}^K m_k \ln \mu_k + \lambda \left(\sum_{k=1}^K \mu_k - 1 \right). \quad (3.20)$$

Обратив производную (3.20) по μ_k в ноль, получим:

$$\mu_k = -m_k / \lambda. \quad (3.21)$$

Подставляя (3.21) в ограничение $\sum_k \mu_k = 1$, можно решить вопрос о множителе Лагранжа λ , получив $\lambda = -N$. Получается решение максимального правдоподобия для μ_k в виде

$$\mu_k^{\text{ML}} = \frac{m_k}{N}, \quad (3.22)$$

что является той частью N наблюдений, для которых $x_k = 1$.

Можно также рассмотреть совместное распределение величин m_1, \dots, m_K , обусловленное вектором параметров μ и общим числом N наблюдений. Исходя из (3.17), оно принимает вид

$$\text{Mult}(m_1, m_2, \dots, m_K | \mu, N) = \binom{N}{m_1 m_2 \dots m_K} \prod_{k=1}^K \mu_k^{m_k}, \quad (3.23)$$

что известно как *полиномиальное распределение* (*multinomial distribution*). Коэффициент нормализации представляет собой число способов разбиения N объектов на K групп размером m_1, \dots, m_K и определяется как

$$\binom{N}{m_1 m_2 \dots m_K} = \frac{N!}{m_1! m_2! \dots m_K!}. \quad (3.24)$$

Обратите внимание, что величины с двумя состояниями могут быть представлены либо как бинарные переменные с моделированием через биномиальное распределение (3.9), либо как переменные вида «1 из 2» с моделированием через распределение (3.14) с $K = 2$.

3.2. Многомерное гауссово распределение

Гауссово распределение, также называемое нормальным распределением, представляет собой широко распространенную модель распределения непрерывных переменных. Ранее (см. раздел 2.3) уже было показано, что для отдельной переменной x гауссово распределение можно записать в виде

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}, \quad (3.25)$$

где μ – среднее значение, а σ^2 – дисперсия. Для D -мерного вектора \mathbf{x} много-мерное гауссово распределение имеет вид

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}, \quad (3.26)$$

где $\boldsymbol{\mu}$ – D -мерный средний вектор, $\boldsymbol{\Sigma}$ – ковариационная матрица $D \times D$, а $\det \boldsymbol{\Sigma}$ – детерминант $\boldsymbol{\Sigma}$.

Гауссово распределение используется во многих различных ситуациях, и это может быть обосновано различными причинами (см. раздел 2.5). Например, как уже было отмечено, распределение, максимизирующее энтропию для отдельной вещественной переменной, является гауссовым. Это свойство применимо и к многомерному гауссовому распределению (см. упражнение 3.8).

В другой ситуации гауссово распределение используется для определения суммы нескольких случайных величин. Как гласит *центральная предельная теорема* (*central limit theorem*), при определенных мягких условиях сумма набора случайных величин, которая сама по себе является случайной величиной, имеет распределение, которое приобретает все более гауссовский характер по мере увеличения числа членов в сумме (Walker, 1969). Это можно проиллюстрировать, рассмотрев N переменных x_1, \dots, x_N , каждая из которых имеет равномерное распределение в интервале $[0, 1]$, а затем проанализировать распределение среднего значения $(x_1 + \dots + x_N)/N$. При больших N это распределение стремится к гауссову, как показано на рис. 3.2. На практике сближение с гауссовым распределением при увеличении N может быть очень быстрым. Одним из следствий этого является тот факт, что биномиальное распределение (3.9), которое представляет собой распределение по t и определяется суммой N наблюдений случайной двоичной переменной x , будет стремиться к гауссовому при $N \rightarrow \infty$ (см. рис. 3.1 для $N = 10$).

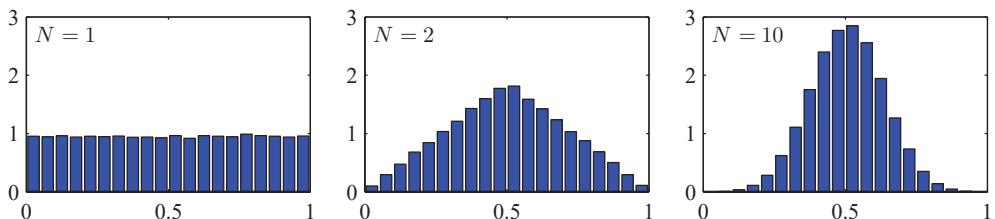


РИС. 3.2 Гистограммы среднего значения N равномерно распределенных чисел для различных значений N

Гауссово распределение обладает многими важными аналитическими свойствами, и некоторые из них будут рассмотрены более подробно. В ре-

зультате этот раздел будет более технически сложным, чем некоторые из предыдущих разделов, и потребует знакомства с различными матричными тождествами (см. приложение А).

3.2.1. Геометрия гауссова распределения

Для начала рассмотрим геометрическую форму гауссова распределения. Функциональная зависимость гауссова распределения от \mathbf{x} имеет квадратичную зависимость:

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}), \quad (3.27)$$

которая проявляется в экспоненте. Величина Δ называется *расстоянием Махаланобиса* (*Mahalanobis distance*), или *обобщенным расстоянием от $\boldsymbol{\mu}$ до \mathbf{x}* . Она сводится к евклидову расстоянию при условии, что $\boldsymbol{\Sigma}$ является единичной матрицей. Гауссово распределение постоянно на таких поверхностях в пространстве \mathbf{x} , для которых эта квадратичная форма постоянна.

Прежде всего отметим, что матрицу $\boldsymbol{\Sigma}$ можно рассматривать как симметричную без потери общности (см. упражнение 3.11), поскольку любая антисимметричная компонента из экспоненты просто исчезнет. Теперь рассмотрим уравнение собственных векторов для ковариационной матрицы:

$$\boldsymbol{\Sigma} \mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad (3.28)$$

где $i = 1, \dots, D$. Поскольку $\boldsymbol{\Sigma}$ представляет собой вещественную симметричную матрицу, ее собственные значения будут вещественными (см. упражнение 3.12), а собственные векторы можно выбрать так, чтобы они образовывали ортонормальное множество, так что

$$\mathbf{u}_i^T \mathbf{u}_j = I_{ij}, \quad (3.29)$$

где I_{ij} – это элемент единичной матрицы на позиции i, j , удовлетворяющий условиям:

$$I_{ij} = \begin{cases} 1, & \text{если } i = j, \\ 0, & \text{в противном случае.} \end{cases} \quad (3.30)$$

Ковариационная матрица $\boldsymbol{\Sigma}$ может быть выражена в виде ее разложения по собственным векторам (см. упражнение 3.13) в форме:

$$\boldsymbol{\Sigma} = \sum_{i=1}^D \lambda_i \mathbf{u}_i \mathbf{u}_i^T, \quad (3.31)$$

и точно так же обратная ковариационная матрица $\boldsymbol{\Sigma}^{-1}$ может быть выражена как

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^D \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T. \quad (3.32)$$

При подстановке (3.32) в (3.27) квадратичная форма приобретает вид

$$\Delta^2 = \sum_{i=1}^D \frac{y_i^2}{\lambda_i}, \quad (3.33)$$

где определено

$$y_i = \mathbf{u}_i^T (\mathbf{x} - \boldsymbol{\mu}). \quad (3.34)$$

Теперь можно интерпретировать $\{y_i\}$ как новую систему координат, определяемую ортонормальными векторами \mathbf{u}_i , которые сдвинуты и повернуты относительно исходных координат x_i . Формируя вектор $\mathbf{y} = (y_1, \dots, y_D)^T$, получаем

$$\mathbf{y} = \mathbf{U}(\mathbf{x} - \boldsymbol{\mu}), \quad (3.35)$$

где \mathbf{U} – это матрица, строки которой заданы \mathbf{u}_i^T . Из (3.29) следует, что \mathbf{U} – это ортогональная матрица (см. приложение А), т. е. она удовлетворяет условию $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$, где \mathbf{I} – это единичная матрица.

Квадратичная форма и, следовательно, гауссова плотность распределения вероятности постоянна на поверхностях, для которых справедливо (3.33). Если все собственные значения λ_i положительны, то эти поверхности представляют собой эллипсоиды с центрами в точках $\boldsymbol{\mu}$ и осями, ориентированными вдоль \mathbf{u}_i , с коэффициентами масштабирования по направлениям осей, равными $\lambda_i^{1/2}$, как показано на рис. 3.3. Оси эллипса определяются собственными векторами \mathbf{u}_i ковариационной матрицы с соответствующими собственными значениями λ_i .

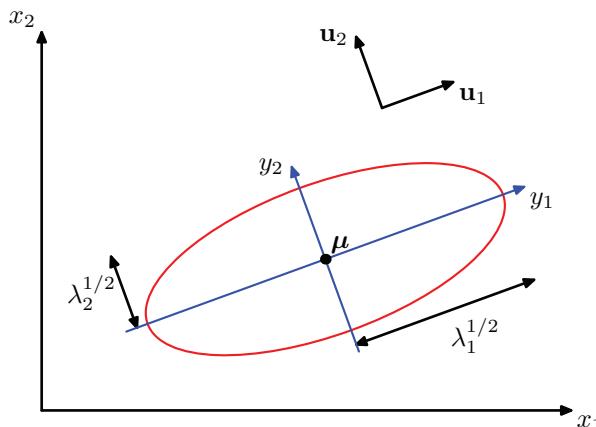


РИС. 3.3 Эллиптическая поверхность постоянной плотности вероятности (красная кривая) для гауссова распределения в двумерном пространстве $\mathbf{x} = (x_1, x_2)$, на которой плотность равна $\exp(-1/2)$ от ее значения при $\mathbf{x} = \boldsymbol{\mu}$

Для точного определения гауссова распределения необходимо, чтобы все собственные значения λ_i ковариационной матрицы были сугубо положительными, иначе распределение не может быть правильно нормализовано. Матрица, собственные значения которой сугубо положительны, называется положительно определенной (positive definite, см. главу 16). При изучении моделей скрытых переменных нам будут встречаться гауссова распределения, для которых одно или несколько собственных значений равны нулю, и в этом случае распределение является сингулярным и ограничено подпространством меньшей размерности. Если все собственные значения неотрицательны, то ковариационная матрица считается положительно полуопределенной (positive semidefinite).

Теперь рассмотрим форму гауссова распределения в новой системе координат, определяемой y_i . При переходе от системы координат x к системе координат y получается матрица Якоби J с элементами, заданными как

$$J_{ij} = \frac{\partial x_i}{\partial y_j} = U_{ji}, \quad (3.36)$$

где U_{ji} – элементы матрицы U^T . Используя свойство ортонормированности матрицы U , получаем, что квадрат определителя матрицы Якоби равен

$$|J|^2 = |U^T|^2 = |U^T||U| = |U^T U| = |\mathbf{I}| = 1 \quad (3.37)$$

и, следовательно, $|J| = 1$. Кроме того, определитель $|\Sigma|$ ковариационной матрицы может быть записан как произведение ее собственных значений, следовательно,

$$|\Sigma|^{1/2} = \prod_{j=1}^D \lambda_j^{1/2}. \quad (3.38)$$

Таким образом, в системе координат y_i гауссово распределение имеет вид

$$p(\mathbf{y}) = p(\mathbf{x})|J| = \prod_{j=1}^D \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\left\{-\frac{y_j^2}{2\lambda_j}\right\}, \quad (3.39)$$

которое является производным D независимых одномерных гауссовых распределений. Таким образом, собственные векторы определяют новый набор сдвинутых и повернутых координат, относительно которых совместное распределение вероятностей факторизуется в произведение независимых распределений. Тогда интеграл распределения в системе координат y имеет вид

$$\int p(\mathbf{y}) d\mathbf{y} = \prod_{j=1}^D \int_{-\infty}^{\infty} \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\left\{-\frac{y_j^2}{2\lambda_j}\right\} dy_j = 1, \quad (3.40)$$

где результат (2.51) использован для нормализации одномерного гауссова распределения. Таким образом, подтверждается, что многомерное гауссово распределение (3.26) действительно нормализовано.

3.2.2. Моменты

Теперь перейдем к рассмотрению моментов гауссова распределения и, соответственно, к интерпретации параметров μ и Σ . Математическое ожидание x для гауссова распределения выражается как

$$\begin{aligned}\mathbb{E}[x] &= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \int \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\} \mathbf{x} d\mathbf{x} \\ &= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \int \exp\left\{-\frac{1}{2}\mathbf{z}^T \boldsymbol{\Sigma}^{-1} \mathbf{z}\right\} (\mathbf{z} + \boldsymbol{\mu}) d\mathbf{z},\end{aligned}\quad (3.41)$$

где переменные заменены на $\mathbf{z} = \mathbf{x} - \boldsymbol{\mu}$. Обратите внимание, что экспонента является четной функцией компонентов \mathbf{z} , и, поскольку интегралы по ним берутся в диапазоне $(-\infty, \infty)$, член \mathbf{z} в множителе $(\mathbf{z} + \boldsymbol{\mu})$ будет исчезать вследствие симметрии.

Таким образом,

$$\mathbb{E}[x] = \boldsymbol{\mu}, \quad (3.42)$$

и поэтому $\boldsymbol{\mu}$ называют средним значением гауссова распределения.

Теперь рассмотрим моменты второго порядка гауссова распределения. В одномерном случае рассматривался момент второго порядка, определяемый $\mathbb{E}[x^2]$. Для многомерного гауссова распределения существуют D^2 моментов второго порядка $\mathbb{E}[x_i x_j]$, которые могут быть объединены в матрицу $\mathbb{E}[\mathbf{x} \mathbf{x}^T]$. Эта матрица может быть записана в виде:

$$\begin{aligned}\mathbb{E}[\mathbf{x} \mathbf{x}^T] &= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \int \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\} \mathbf{x} \mathbf{x}^T d\mathbf{x} \\ &= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \int \exp\left\{-\frac{1}{2}\mathbf{z}^T \boldsymbol{\Sigma}^{-1} \mathbf{z}\right\} (\mathbf{z} + \boldsymbol{\mu})(\mathbf{z} + \boldsymbol{\mu})^T d\mathbf{z}.\end{aligned}\quad (3.43)$$

Здесь вновь произведена замена переменных на $\mathbf{z} = \mathbf{x} - \boldsymbol{\mu}$. Обратите внимание, что скрещивающиеся члены $\boldsymbol{\mu} \mathbf{z}^T$ и $\boldsymbol{\mu}^T \mathbf{z}$ опять исчезают ввиду симметрии. Член $\boldsymbol{\mu} \boldsymbol{\mu}^T$ постоянен и может быть вынесен из интеграла, который сам по себе един, поскольку гауссово распределение является нормализованным. Рассмотрим член с участием $\mathbf{z} \mathbf{z}^T$. И вновь можно воспользоваться собственным векторным разложением ковариационной матрицы, заданной в (3.28), а также полнотой набора собственных векторов, чтобы записать:

$$\mathbf{z} = \sum_{j=1}^D y_j \mathbf{u}_j, \quad (3.44)$$

где $y_j = \mathbf{u}_j^T \mathbf{z}$, что приводит к

$$\begin{aligned} & \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \int \exp \left\{ -\frac{1}{2} \mathbf{z}^T \Sigma^{-1} \mathbf{z} \right\} \mathbf{z} \mathbf{z}^T d\mathbf{z} \\ &= \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \sum_{i=1}^D \sum_{j=1}^D \mathbf{u}_i \mathbf{u}_j^T \int \exp \left\{ -\sum_{k=1}^D \frac{y_k^2}{2\lambda_k} \right\} y_i y_j d\mathbf{y}, \\ &= \sum_{i=1}^D \mathbf{u}_i \mathbf{u}_i^T \lambda_i, \end{aligned} \quad (3.45)$$

где используется уравнение для собственных векторов (3.28), а также то обстоятельство, что интеграл в средней строке исчезает в силу симметрии при $i = j$. В последней строке используются результаты (2.53) и (3.38), а также (3.31). Таким образом, получается:

$$\mathbb{E}[\mathbf{x}\mathbf{x}^T] = \boldsymbol{\mu}\boldsymbol{\mu}^T + \Sigma. \quad (3.46)$$

При определении дисперсии для отдельной случайной величины до получения второго момента из нее вычитается среднее значение. Аналогично в многомерном случае также удобно вычесть среднее значение, в результате чего *ковариация* случайного вектора \mathbf{x} определяется как

$$\text{cov}[\mathbf{x}] = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^T]. \quad (3.47)$$

Для частного случая гауссова распределения можно использовать $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$, а также результат (3.46), чтобы получить

$$\text{cov}[\mathbf{x}] = \Sigma. \quad (3.48)$$

Поскольку матрица параметров Σ определяет ковариацию \mathbf{x} при гауссовом распределении, она называется ковариационной матрицей (covariance matrix).

3.2.3. Ограничения

Хотя гауссово распределение (3.26) зачастую рассматривают в качестве простой модели распределения плотности, оно имеет ряд существенных ограничений. Рассмотрим количество свободных параметров в распределении. Общая симметричная ковариационная матрица Σ будет иметь $D(D + 1)/2$ независимых параметров (см. упражнение 3.15), и есть еще D независимых параметров в $\boldsymbol{\mu}$, что в сумме дает $D(D + 3)/2$ параметров. Таким образом, для больших значений D общее число параметров растет квадратично по мере увеличения D , и вычислительная задача по обработке и инвертированию больших матриц может стать непомерно сложной. Одним из способов решения этой проблемы является использование ограниченных форм ковариационной матрицы. Если рассматривать ковариационные матрицы, которые являются диагональными, так что $\Sigma = \text{diag}(\sigma_i^2)$, то в модели распределения

плотности будет в общей сложности $2D$ независимых параметров. Соответствующие контуры постоянной плотности задаются ориентированными по осям эллипсоидами. В дальнейшем можно еще ограничить ковариационную матрицу, чтобы она была пропорциональна единичной матрице, $\Sigma = \sigma^2 \mathbf{I}$, известной как изотропная (isotropic) ковариация, что приводит к $D + 1$ независимым параметрам в модели вместе со сферическими поверхностями постоянной плотности. Три варианта общей, диагональной и изотропной ковариационных матриц показаны на рис. 3.4. К сожалению, несмотря на ограничение числа степеней свободы в распределении и ускорение инверсии ковариационной матрицы, такие подходы сильно ограничивают формат распределения плотности вероятности и ограничивают ее возможности по выявлению возможных корреляций в данных.

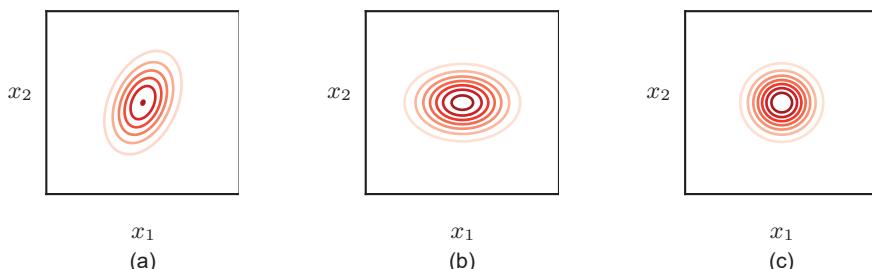


РИС. 3.4 Контуры постоянной плотности вероятности для гауссова распределения в двух измерениях, где ковариационная матрица (a) имеет общий вид, (b) является диагональной (эллиптические контуры совпадают с координатными осями) и (c) пропорциональна единичной матрице (контуры в виде концентрических окружностей)

Другим ограничением гауссова распределения является то, что оно по своей сути унимодально (т. е. содержит один максимум) и поэтому не может обеспечить хорошую аппроксимацию мультимодальных распределений. Получается, гауссово распределение может быть слишком гибким с точки зрения наличия большого количества параметров и слишком ограниченным в диапазоне распределений, которые оно может соответствующим образом представлять. Позже будет показано, что введение латентных (*latent*) переменных, также называемых скрытыми (*hidden*) и/или ненаблюдаемыми (*unobserved*) переменными, позволяет решить обе эти проблемы. В частности, обширное семейство мультимодальных распределений получается при введении дискретных скрытых переменных, приводящих к образованию смешанных гауссовых распределений (см. раздел 3.2.9). Подобным образом введение непрерывных латентных переменных приводит к моделям, в которых число свободных параметров может контролироваться независимо от размерности D пространства данных и при этом позволяет модели отражать доминирующие корреляции в наборе данных (см. главу 16).

3.2.4. Условное распределение

Важным свойством многомерного гауссова распределения можно считать то, что при совместном гауссовом распределении двух наборов переменных условное распределение одного из них, обусловленное другим, также является гауссовым. Аналогично маргинальное распределение любого из наборов также является гауссовым.

Сначала рассмотрим случай условных распределений. Предположим, что \mathbf{x} – это D -мерный вектор с гауссовым распределением $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$, которое разбивается на два несовпадающих подмножества \mathbf{x}_a и \mathbf{x}_b . Без потери общности можно предположить, что \mathbf{x}_a образует первые M компонентов \mathbf{x} , а \mathbf{x}_b – оставшиеся $-M$ компонентов, так что

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}. \quad (3.49)$$

Также определим соответствующие разбиения для среднего вектора $\boldsymbol{\mu}$ в виде

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix} \quad (3.50)$$

и ковариационной матрицы $\boldsymbol{\Sigma}$, заданной выражением

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}. \quad (3.51)$$

Отметим, что из симметрии $\boldsymbol{\Sigma}^T = \boldsymbol{\Sigma}$ ковариационной матрицы следует, что $\boldsymbol{\Sigma}_{aa}$ и $\boldsymbol{\Sigma}_{bb}$ симметричны и что $\boldsymbol{\Sigma}_{ba} = \boldsymbol{\Sigma}_{ab}^T$.

Во многих ситуациях будет удобно работать с обратной ковариационной матрицей:

$$\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}, \quad (3.52)$$

которая известна как *точная матрица* (*precision matrix*). На самом деле в дальнейшем будет показано, что некоторые свойства гауссовых распределений наиболее естественно выражаются в контексте ковариаций, в то время как другие принимают более простую форму в контексте точности. Поэтому также введем разделенную форму матрицы точности:

$$\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}. \quad (3.53)$$

Поскольку обратная симметричная матрица также симметрична (см. упражнение 3.16), получается, что $\boldsymbol{\Lambda}_{aa}$ и $\boldsymbol{\Lambda}_{bb}$ симметричны и что $\boldsymbol{\Lambda}_{ba} = \boldsymbol{\Lambda}_{ab}^T$. Следует подчеркнуть, что, например, $\boldsymbol{\Lambda}_{aa}$ не просто является обратной величиной

от Σ_{aa} . На самом деле в ближайшее время будет рассмотрена связь между инверсией матрицы разбиения и инверсиями ее разбиений.

Начнем с определения выражения для условного распределения $p(\mathbf{x}_a | \mathbf{x}_b)$. Из правила произведения вероятностей видно, что это условное распределение можно вычислить из совместного распределения $p(\mathbf{x}) = p(\mathbf{x}_a, \mathbf{x}_b)$, просто привязав \mathbf{x}_b к наблюдаемому значению и выполнив нормализацию полученного выражения для получения действительного распределения вероятностей по \mathbf{x}_a . Вместо проведения этой нормализации в явном виде можно найти более эффективное решение, рассмотрев квадратичную форму в экспоненте гауссова распределения, заданного в (3.27), а затем восстановив коэффициент нормализации при завершении вычислений. Если воспользоваться разбиениями (3.49), (3.50) и (3.53), можно получить

$$\begin{aligned} -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = \\ -\frac{1}{2}(\mathbf{x}_a - \boldsymbol{\mu}_a)^T \boldsymbol{\Lambda}_{aa} (\mathbf{x}_a - \boldsymbol{\mu}_a) - \frac{1}{2}(\mathbf{x}_a - \boldsymbol{\mu}_a)^T \boldsymbol{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b) \\ -\frac{1}{2}(\mathbf{x}_b - \boldsymbol{\mu}_b)^T \boldsymbol{\Lambda}_{ba} (\mathbf{x}_a - \boldsymbol{\mu}_a) - \frac{1}{2}(\mathbf{x}_b - \boldsymbol{\mu}_b)^T \boldsymbol{\Lambda}_{bb} (\mathbf{x}_b - \boldsymbol{\mu}_b). \end{aligned} \quad (3.54)$$

Здесь видно, для функции \mathbf{x}_a это снова квадратичная форма, и, следовательно, соответствующее условное распределение $p(\mathbf{x}_a | \mathbf{x}_b)$ будет гауссовым. Поскольку это распределение полностью характеризуется своим средним значением и ковариацией, задача состоит в определении выражений для среднего значения и ковариации $p(\mathbf{x}_a | \mathbf{x}_b)$ путем проверки (3.54).

Это пример довольно распространенной операции, связанной с гауссовыми распределениями, иногда называемой «дополнением до полного квадрата», когда дана квадратичная форма, определяющая члены экспоненты в гауссовом распределении, и нужно найти соответствующие среднее значение и ковариацию. Такие задачи можно решить простым способом, если учесть, что экспонента в общем гауссовом распределении $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$ может быть записана в виде

$$-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = -\frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x} + \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \text{const}, \quad (3.55)$$

где const обозначает члены, не зависящие от \mathbf{x} . Кроме того, в данном случае была использована симметрия $\boldsymbol{\Sigma}$. Таким образом, если взять общую квадратичную форму и выразить ее в виде, заданном правой частью (3.55), то матрицу коэффициентов, входящих в член второго порядка в \mathbf{x} , можно сразу приравнять к обратной ковариационной матрице $\boldsymbol{\Sigma}^{-1}$, а коэффициент линейного члена в \mathbf{x} приравнять к $\boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$, из чего можно вывести $\boldsymbol{\mu}$.

Теперь применим эту процедуру к условному гауссовому распределению $p(\mathbf{x}_a | \mathbf{x}_b)$, для которого квадратичная форма в экспоненте задается выражением (3.54). Обозначим среднее значение и ковариацию этого распределения через $\boldsymbol{\mu}_{a|b}$ и $\boldsymbol{\Sigma}_{a|b}$ соответственно. Рассмотрим функциональную зависимость

(3.54) от \mathbf{x}_a , в которой \mathbf{x}_b рассматривается как константа. Если в \mathbf{x}_a отобрать все члены второго порядка, то получится

$$-\frac{1}{2} \mathbf{x}_a^T \boldsymbol{\Lambda}_{aa} \mathbf{x}_a, \quad (3.56)$$

из чего можно сразу же сделать вывод, что ковариация (обратная точность) $p(\mathbf{x}_a | \mathbf{x}_b)$ задается как

$$\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Lambda}_{aa}^{-1}. \quad (3.57)$$

Теперь рассмотрим все члены в (3.54), которые линейны в \mathbf{x}_a :

$$\mathbf{x}_a^T \{\boldsymbol{\Lambda}_{aa} \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b)\}, \quad (3.58)$$

где используется $\boldsymbol{\Lambda}_{ba}^T = \boldsymbol{\Lambda}_{ab}$. Из нашего анализа общей формы (3.55) следует, что коэффициент \mathbf{x}_a в этом выражении должен быть равен $\boldsymbol{\Sigma}_{a|b}^{-1} \boldsymbol{\mu}_{a|b}$ и, следовательно,

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\Sigma}_{a|b} \{\boldsymbol{\Lambda}_{aa} \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b)\} = \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1} \boldsymbol{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b), \quad (3.59)$$

где используется (3.57).

Результаты (3.57) и (3.59) выражаются в виде разделенной матрицы точности исходного совместного распределения $p(\mathbf{x}_a, \mathbf{x}_b)$. Эти результаты также можно выразить в виде соответствующей разделенной ковариационной матрицы. Для этого воспользуемся следующим тождеством для обратной части разделенной матрицы (см. упражнение 3.18):

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{M} & -\mathbf{MBD}^{-1} \\ -\mathbf{D}^{-1}\mathbf{CM} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{CMBD}^{-1} \end{pmatrix}, \quad (3.60)$$

где задано

$$\mathbf{M} = (\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1}. \quad (3.61)$$

Величина \mathbf{M}^{-1} известна как *дополнение Шура* (*Schur complement*) для матрицы в левой части (3.60) относительно подматрицы \mathbf{D} . Используя определение

$$\begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix} \quad (3.62)$$

и используя (3.60), получаем:

$$\boldsymbol{\Lambda}_{aa} = (\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1} \boldsymbol{\Sigma}_{ba})^{-1}, \quad (3.63)$$

$$\boldsymbol{\Lambda}_{ab} = -(\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1} \boldsymbol{\Sigma}_{ba})^{-1} \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1}. \quad (3.64)$$

Отсюда получаем следующие выражения для среднего значения и ковариации условного распределения $p(\mathbf{x}_a | \mathbf{x}_b)$:

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1} (\mathbf{x}_b - \boldsymbol{\mu}_b), \quad (3.65)$$

$$\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab} \boldsymbol{\Sigma}_{bb}^{-1} \boldsymbol{\Sigma}_{ba}. \quad (3.66)$$

Сравнивая (3.57) и (3.66), можно видеть, что условное распределение $p(\mathbf{x}_a | \mathbf{x}_b)$ имеет более простую форму, когда оно выражается в виде разделенной матрицы точности, нежели когда оно выражается в виде разделенной ковариационной матрицы. Обратите внимание, что среднее значение условного распределения $p(\mathbf{x}_a | \mathbf{x}_b)$, заданное в (3.65), является линейной функцией \mathbf{x}_b , а ковариация, заданная в (3.66), не зависит от \mathbf{x}_b . Она представляет собой пример линейной гауссовой (*linear-Gaussian*) модели (см. раздел 11.1.4).

3.2.5. Маргинальное распределение

Ранее было показано, что в случае, когда совместное распределение $p(\mathbf{x}_a, \mathbf{x}_b)$ является гауссовым, условное распределение $p(\mathbf{x}_a | \mathbf{x}_b)$ также будет гауссовым. Теперь рассмотрим маргинальное распределение, выраженное как

$$p(\mathbf{x}_a) = \int p(\mathbf{x}_a, \mathbf{x}_b) d\mathbf{x}_b, \quad (3.67)$$

которое, как будет показано далее, также является гауссовым. И вновь стратегией вычисления этого распределения будет обращение к квадратичной форме в экспоненте совместного распределения, чтобы затем определить среднее значение и ковариацию маргинального распределения $p(\mathbf{x}_a)$.

Квадратичная форма для совместного распределения может быть выражена с использованием разделенной матрицы точности в виде (3.54). Целью является интегрирование по \mathbf{x}_b , что проще всего сделать, рассмотрев сначала члены, включающие \mathbf{x}_b , а затем возведя их в квадрат для упрощения интегрирования. Выбирая только те члены, которые связаны с \mathbf{x}_b , получим

$$-\frac{1}{2} \mathbf{x}_b^T \boldsymbol{\Lambda}_{bb} \mathbf{x}_b + \mathbf{x}_b^T \mathbf{m} = -\frac{1}{2} (\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m})^T \boldsymbol{\Lambda}_{bb} (\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m}) + \frac{1}{2} \mathbf{m}^T \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m}, \quad (3.68)$$

где определяется

$$\mathbf{m} = \boldsymbol{\Lambda}_{bb} \boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{ba} (\mathbf{x}_a - \boldsymbol{\mu}_a). \quad (3.69)$$

Как видно, зависимость от \mathbf{x}_b была приведена к стандартной квадратичной форме гауссова распределения, соответствующей первому члену в правой части (3.68), плюс член, не зависящий от \mathbf{x}_b (но зависящий от \mathbf{x}_a). Таким образом, взяв экспоненту этой квадратичной формы, можно убедиться, что интегрирование по \mathbf{x}_b , необходимое в (3.67), примет вид:

$$\int \exp \left\{ -\frac{1}{2} (\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m})^T \boldsymbol{\Lambda}_{bb} (\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m}) \right\} d\mathbf{x}_b. \quad (3.70)$$

Это интегрирование выполняется достаточно просто за счет того, что это интеграл по ненормализованному гауссовому распределению, и поэтому результат будет обратным коэффициенту нормализации. Из формы ненор-

мализованного гауссова распределения (3.26) понятно, что этот коэффициент не зависит от среднего значения и определяется только детерминантом ковариационной матрицы. Таким образом, возведя в квадрат \mathbf{x}_b , можно проинтегрировать \mathbf{x}_b так, что из всего содержимого левой части (3.68), зависящего от \mathbf{x}_a , останется только последний член в правой части (3.68), в которой \mathbf{m} задается условием (3.69). Комбинируя этот член с остальными членами из (3.54), зависящими от \mathbf{x}_a , получаем

$$\begin{aligned} & \frac{1}{2} [\Lambda_{bb}\boldsymbol{\mu}_b - \Lambda_{bb}(\mathbf{x}_a - \boldsymbol{\mu}_a)]^T \Lambda_{bb}^{-1} [\Lambda_{bb}\boldsymbol{\mu}_b - \Lambda_{ba}(\mathbf{x}_a - \boldsymbol{\mu}_a)] \\ & - \frac{1}{2} \mathbf{x}_a^T \Lambda_{aa} \mathbf{x}_a + \mathbf{x}_a^T (\Lambda_{aa}\boldsymbol{\mu}_a + \Lambda_{ab}\boldsymbol{\mu}_b) + \text{const} \\ & = -\frac{1}{2} \mathbf{x}_a^T (\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}) \mathbf{x}_a \\ & + \mathbf{x}_a^T (\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}) \boldsymbol{\mu}_a + \text{const}, \end{aligned} \quad (3.71)$$

где const обозначает величины, не зависящие от \mathbf{x}_a . Опять же, сравнивая с (3.55), можно увидеть, что ковариация маргинального распределения $p(\mathbf{x}_a)$ задается через

$$\boldsymbol{\Sigma}_a = (\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba})^{-1}. \quad (3.72)$$

Точно так же среднее значение определяется

$$\boldsymbol{\Sigma}_a (\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}) \boldsymbol{\mu}_a = \boldsymbol{\mu}_a, \quad (3.73)$$

где используется (3.72). Ковариация (3.72) выражается в виде разделенной матрицы точности, заданной в (3.53). Ее можно переписать в виде соответствующего разбиения ковариационной матрицы, заданной в (3.51), как это было сделано для условного распределения. Эти разделенные матрицы связаны между собой соотношением:

$$\begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}. \quad (3.74)$$

Используя (3.60), получаем:

$$(\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba})^{-1} = \boldsymbol{\Sigma}_{aa}. \quad (3.75)$$

Отсюда следует достаточно очевидный результат: маргинальное распределение $p(\mathbf{x}_a)$ имеет среднее значение и ковариацию, заданные как

$$\mathbb{E}[\mathbf{x}_a] = \boldsymbol{\mu}_a, \quad (3.76)$$

$$\text{cov}[\mathbf{x}_a] = \boldsymbol{\Sigma}_{aa}. \quad (3.77)$$

Как видно, для маргинального распределения среднее значение и ковариация выражаются наиболее просто в виде разделенной ковариационной

матрицы, в отличие от условного распределения, для которого разделенная матрица точности приводит к более простым формулам.

Наши результаты для маргинального и условного распределений разделенного гауссова распределения можно обобщить следующим образом. Дано совместное гауссово распределение $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$ с $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$ и следующие его разделения:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}, \quad (3.78)$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}. \quad (3.79)$$

Тогда условное распределение будет иметь вид:

$$p(\mathbf{x}_a | \mathbf{x}_b) = \mathcal{N}(\mathbf{x}_a | \boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}), \quad (3.80)$$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1} \boldsymbol{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b), \quad (3.81)$$

а маргинальное распределение задается как

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a | \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}). \quad (3.82)$$

На рис. 3.5 представлена концепция условных и маргинальных распределений для многомерного гауссова распределения на примере двух переменных.

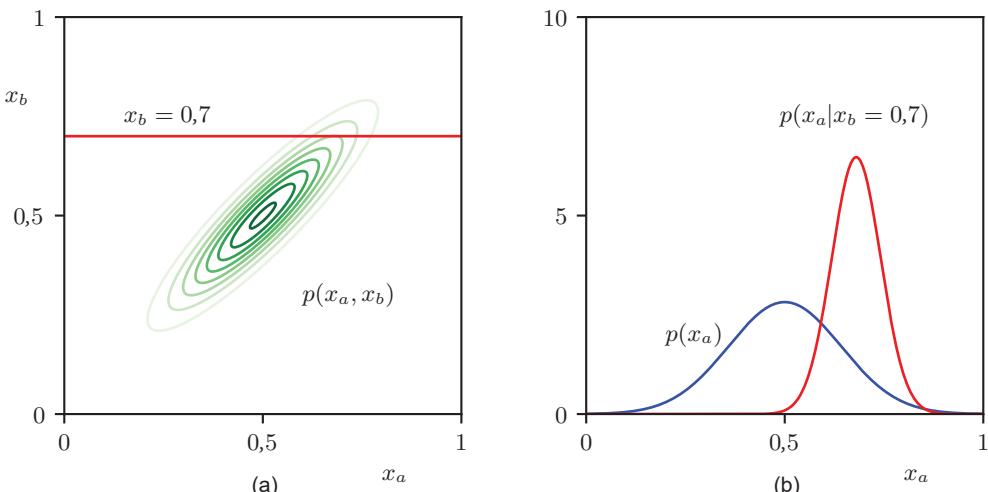


РИС. 3.5 (a) Контуры гауссова распределения $p(\mathbf{x}_a, \mathbf{x}_b)$ по двум переменным. (б) Маргинальное распределение $p(\mathbf{x}_a)$ (синяя кривая) и условное распределение $p(\mathbf{x}_a | \mathbf{x}_b)$ для $\mathbf{x}_b = 0,7$ (красная кривая)

3.2.6. Теорема Байеса

В разделах 3.2.4 и 3.2.5 было рассмотрено гауссово распределение $p(\mathbf{x})$, в котором вектор \mathbf{x} был разбит на два подвектора $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, а затем найдены выражения для условного распределения $p(\mathbf{x}_a | \mathbf{x}_b)$ и маргинального распределения $p(\mathbf{x}_a)$. Как было отмечено, среднее значение условного распределения $p(\mathbf{x}_a | \mathbf{x}_b)$ является линейной функцией \mathbf{x}_b . Здесь мы предположим, что нам дано гауссово маргинальное распределение $p(\mathbf{x})$ и гауссово условное распределение $p(\mathbf{y} | \mathbf{x})$, в котором $p(\mathbf{y} | \mathbf{x})$ имеет среднее значение в виде линейной функции по \mathbf{x} и ковариацию, которая не зависит от \mathbf{x} . Это пример линейной гауссовой модели (Roweis and Ghahramani, 1999) (см. раздел 11.1.4). Необходимо найти маргинальное распределение $p(\mathbf{y})$ и условное распределение $p(\mathbf{x} | \mathbf{y})$. Это структура, которая встречается в некоторых типах генеративных моделей (см. главу 16), и поэтому в данном случае будет удобнее вывести обобщенные результаты.

Будем считать, что маргинальное и условное распределения имеют вид:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad (3.83)$$

$$p(\mathbf{y} | \mathbf{x}) = \mathcal{N}(\mathbf{y} | \mathbf{Ax} + \mathbf{b}, \mathbf{L}^{-1}), \quad (3.84)$$

где $\boldsymbol{\mu}$, \mathbf{A} и \mathbf{b} – это параметры, определяющие значения, а $\boldsymbol{\Lambda}$ и \mathbf{L} – матрицы точности. Если \mathbf{x} имеет размерность M , а \mathbf{y} – размерность D , то матрица \mathbf{A} имеет размер $D \times M$.

Сначала найдем выражение для совместного распределения по \mathbf{x} и \mathbf{y} . Для этого определим

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}, \quad (3.85)$$

а затем рассчитаем логарифм совместного распределения:

$$\begin{aligned} \ln p(\mathbf{z}) &= \ln p(\mathbf{x}) + \ln p(\mathbf{y} | \mathbf{x}) \\ &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (\mathbf{x} - \boldsymbol{\mu}) \\ &\quad -\frac{1}{2}(\mathbf{y} - \mathbf{Ax} - \mathbf{b})^T \mathbf{L} (\mathbf{y} - \mathbf{Ax} - \mathbf{b}) + \text{const}, \end{aligned} \quad (3.86)$$

где const обозначает члены, не зависящие от \mathbf{x} и \mathbf{y} . Как и раньше, здесь наблюдается квадратичная функция по компонентам \mathbf{z} , а это значит, что $p(\mathbf{z})$ есть гауссово распределение. Чтобы определить точность этого гауссова распределения, рассмотрим члены второго порядка в (3.86), которые можно записать в виде

$$\begin{aligned} &-\frac{1}{2} \mathbf{x}^T (\boldsymbol{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{A}) \mathbf{x} - \frac{1}{2} \mathbf{y}^T \mathbf{L} \mathbf{y} + \frac{1}{2} \mathbf{y}^T \mathbf{L} \mathbf{A} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{A}^T \mathbf{L} \mathbf{y} \\ &= -\frac{1}{2} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{A} & -\mathbf{A}^T \mathbf{L} \\ -\mathbf{L} \mathbf{A} & \mathbf{L} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = -\frac{1}{2} \mathbf{z}^T \mathbf{R} \mathbf{z}, \end{aligned} \quad (3.87)$$

и поэтому гауссово распределение по \mathbf{z} имеет точную (обратную ковариационную) матрицу, заданную как

$$\mathbf{R} = \begin{pmatrix} \Lambda + \mathbf{A}^T \mathbf{L} \Lambda & -\mathbf{A}^T \mathbf{L} \\ -\mathbf{L} \Lambda & \mathbf{L} \end{pmatrix}. \quad (3.88)$$

Ковариационная матрица определяется путем взятия обратной величины точности, что можно сделать с помощью формулы инверсии матрицы (3.60) (см. упражнение 3.23), и это дает

$$\text{cov}[\mathbf{z}] = \mathbf{R}^{-1} = \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1} \mathbf{A}^T \\ \mathbf{A} \Lambda^{-1} & \mathbf{L}^{-1} + \mathbf{A} \Lambda^{-1} \mathbf{A}^T \end{pmatrix}. \quad (3.89)$$

Точно так же можно найти среднее значение гауссова распределения по \mathbf{z} , идентифицируя линейные члены в (3.86) по формуле:

$$\mathbf{x}^T \Lambda \boldsymbol{\mu} - \mathbf{x}^T \mathbf{A}^T \mathbf{L} \mathbf{b} + \mathbf{y}^T \mathbf{L} \mathbf{b} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \begin{pmatrix} \Lambda \boldsymbol{\mu} - \mathbf{A}^T \mathbf{L} \mathbf{b} \\ \mathbf{L} \mathbf{b} \end{pmatrix}. \quad (3.90)$$

Используя предыдущий результат (3.55), полученный путем возведения в квадрат квадратичной формы многомерного гауссова распределения, можно определить, что среднее значение \mathbf{z} задается как

$$\mathbb{E}[\mathbf{z}] = \mathbf{R}^{-1} \begin{pmatrix} \Lambda \boldsymbol{\mu} - \mathbf{A}^T \mathbf{L} \mathbf{b} \\ \mathbf{L} \mathbf{b} \end{pmatrix}. \quad (3.91)$$

Используя (3.89) (см. упражнение 3.24), получаем:

$$\mathbb{E}[\mathbf{z}] = \begin{pmatrix} \boldsymbol{\mu} \\ \mathbf{A} \boldsymbol{\mu} + \mathbf{b} \end{pmatrix}. \quad (3.92)$$

Далее найдем выражение для маргинального распределения $p(\mathbf{y})$, в котором маргинализация произведена по \mathbf{x} . Напомним, что маргинальное распределение по подмножеству компонент гауссова случайного вектора имеет особенно простую форму, если выразить его в виде разделенной ковариационной матрицы (см. раздел 3.2). В частности, ее среднее значение и ковариация задаются в (3.76) и (3.77) соответственно. Используя (3.89) и (3.92), можно определить, что среднее значение и ковариация маргинального распределения $p(\mathbf{y})$ задаются как

$$\mathbb{E}[\mathbf{y}] = \mathbf{A} \boldsymbol{\mu} + \mathbf{b}, \quad (3.93)$$

$$\text{cov}[\mathbf{y}] = \mathbf{L}^{-1} + \mathbf{A} \Lambda^{-1} \mathbf{A}^T. \quad (3.94)$$

Частным случаем этого результата является условие $\mathbf{A} = \mathbf{I}$, и в этом случае маргинальное распределение сводится к сворачиванию двух гауссовых рас-

пределений, для которых среднее значение сворачивания равно сумме средних значений двух гауссовых распределений, а ковариация сворачивания равна сумме их ковариаций.

Наконец, найдем формулу для условного распределения $p(\mathbf{x}|\mathbf{y})$. Напомним, что результаты для условного распределения проще всего выразить в виде разделенной матрицы точности (см. раздел 3.2) с использованием (3.57) и (3.59). Применяя эти результаты к (3.89) и (3.92), можно увидеть, что условное распределение $p(\mathbf{x}|\mathbf{y})$ имеет среднее значение и ковариацию, заданные как

$$\mathbb{E}[\mathbf{x}|\mathbf{y}] = (\Lambda + \mathbf{A}^T \mathbf{L} \Lambda)^{-1} \{ \mathbf{A}^T \mathbf{L}(\mathbf{y} - \mathbf{b}) + \Lambda \boldsymbol{\mu} \}, \quad (3.95)$$

$$\text{cov}[\mathbf{x}|\mathbf{y}] = (\Lambda + \mathbf{A}^T \mathbf{L} \Lambda)^{-1}. \quad (3.96)$$

Оценку этого условного распределения можно рассматривать как пример теоремы Байеса, в которой $p(\mathbf{x})$ интерпретируется как априорное распределение по \mathbf{x} . Если переменная \mathbf{y} наблюдаема, то условное распределение $p(\mathbf{x}|\mathbf{y})$ представляет собой соответствующее апостериорное распределение по \mathbf{x} . Найдя маргинальное и условное распределения, можно эффективно выразить совместное распределение $p(\mathbf{z}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$ в виде $p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$.

Эти результаты можно обобщить следующим образом. При маргинальном гауссовом распределении для \mathbf{x} и условном гауссовом распределении для \mathbf{y} по \mathbf{x} получается

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Lambda^{-1}), \quad (3.97)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{Ax} + \mathbf{b}, \mathbf{L}^{-1}), \quad (3.98)$$

и тогда маргинальное распределение \mathbf{y} и условное распределение \mathbf{x} по \mathbf{y} задаются как

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\Lambda^{-1}\mathbf{A}^T), \quad (3.99)$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma} \{ \mathbf{A}^T \mathbf{L}(\mathbf{y} - \mathbf{b}) + \Lambda \boldsymbol{\mu} \}, \boldsymbol{\Sigma}), \quad (3.100)$$

где

$$\boldsymbol{\Sigma} = (\Lambda + \mathbf{A}^T \mathbf{L} \Lambda)^{-1}. \quad (3.101)$$

3.2.7. Максимальное правдоподобие

Для набора данных $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$, где наблюдения $\{\mathbf{x}_n\}$ считаются взятыми независимо друг от друга из многомерного гауссова распределения, параметры распределения можно оценить методом максимального правдоподобия. Функция логарифмического правдоподобия имеет вид:

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{ND}{2} \ln(2\pi) - \frac{N}{2} \ln |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}). \quad (3.102)$$

Путем несложной перестановки можно убедиться, что функция правдоподобия зависит от набора данных только по двум параметрам:

$$\sum_{n=1}^N \mathbf{x}_n, \quad \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T. \quad (3.103)$$

Эти параметры известны как *достаточные статистики* (*sufficient statistics*) для гауссова распределения. Используя (A.19) (см. приложение A), можно получить производную логарифмического правдоподобия по μ в виде

$$\frac{\partial}{\partial \mu} \ln p(\mathbf{X} | \mu, \Sigma) = \sum_{n=1}^N \Sigma^{-1} (\mathbf{x}_n - \mu), \quad (3.104)$$

и далее, обращая эту производную в ноль, можно получить решение для оценки среднего значения с максимальным правдоподобием:

$$\boldsymbol{\mu}_{ML} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n, \quad (3.105)$$

которое является средним значением наблюдаемого набора точек данных. Максимизация (3.102) по Σ является более сложной задачей. Самый простой подход состоит в игнорировании ограничения симметрии (см. упражнение 3.28) и доказательстве того, что полученное решение является симметричным, как и должно быть.

Альтернативные способы получения этого результата, в которых ограничения на симметрию и положительную определенность накладываются в явном виде, можно найти в работе (Magnus and Neudecker, 1999). Результат, как и следовало ожидать, имеет вид

$$\boldsymbol{\Sigma}_{ML} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu}_{ML})(\mathbf{x}_n - \boldsymbol{\mu}_{ML})^T, \quad (3.106)$$

в котором фигурирует $\boldsymbol{\mu}_{ML}$, поскольку это следствие совместной максимизации по μ и Σ . Обратите внимание, что решение (3.105) для $\boldsymbol{\mu}_{ML}$ не зависит от $\boldsymbol{\Sigma}_{ML}$, и поэтому сначала можно определить $\boldsymbol{\mu}_{ML}$, а затем использовать эту оценку для определения $\boldsymbol{\Sigma}_{ML}$.

Если мы оценим ожидания решений максимального правдоподобия при истинном распределении (см. упражнение 3.29), то в результате получится следующее:

$$\mathbb{E}[\boldsymbol{\mu}_{ML}] = \boldsymbol{\mu}, \quad (3.107)$$

$$\mathbb{E}[\boldsymbol{\Sigma}_{ML}] = \frac{N-1}{N} \boldsymbol{\Sigma}. \quad (3.108)$$

Получается, ожидание оценки максимального правдоподобия для среднего значения равно истинному среднему значению. Однако оценка мак-

симального правдоподобия для ковариации характеризуется ожиданием, которое меньше истинного значения, и, следовательно, она является смещенной. Это смещение можно устранить, определив другую оценку $\tilde{\Sigma}$, которая выглядит как

$$\tilde{\Sigma} = \frac{1}{N-1} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu}_{\text{ML}})(\mathbf{x}_n - \boldsymbol{\mu}_{\text{ML}})^T. \quad (3.109)$$

Из (3.106) и (3.108) следует, что ожидание $\tilde{\Sigma}$ равно Σ .

3.2.8. Последовательная оценка

Рассмотренный выше метод максимального правдоподобия представляет собой *метод групповой (пакетной) обработки (batch method)*, при котором рассматривается сразу весь набор обучающих данных. Альтернативой являются *последовательные методы (sequential methods)*, которые позволяют обрабатывать точки данных по одной и затем удалять их. Это важно для онлайн-приложений и для больших данных, когда групповая обработка всех точек данных за один раз нецелесообразна.

Рассмотрим результат (3.105) для оценки максимального правдоподобия среднего значения $\boldsymbol{\mu}_{\text{ML}}$, которую обозначим как $\boldsymbol{\mu}_{\text{ML}}^{(N)}$, исходя из N наблюдений. Если вычленить вклад от конечной точки данных \mathbf{x}_N , то получим:

$$\begin{aligned} \boldsymbol{\mu}_{\text{ML}}^{(N)} &= \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \\ &= \frac{1}{N} \mathbf{x}_N + \frac{1}{N} \sum_{n=1}^{N-1} \mathbf{x}_n \\ &= \frac{1}{N} \mathbf{x}_N + \frac{N-1}{N} \boldsymbol{\mu}_{\text{ML}}^{(N-1)} \\ &= \boldsymbol{\mu}_{\text{ML}}^{(N-1)} + \frac{1}{N} (\mathbf{x}_N - \boldsymbol{\mu}_{\text{ML}}^{(N-1)}). \end{aligned} \quad (3.110)$$

Этот результат имеет следующую интересную интерпретацию. После наблюдения $N-1$ точек данных оценим $\boldsymbol{\mu}$ по $\boldsymbol{\mu}_{\text{ML}}^{(N-1)}$. Теперь наблюдается точка данных \mathbf{x}_N , и получается пересмотренная оценка $\boldsymbol{\mu}_{\text{ML}}^{(N)}$ путем перемещения старой оценки на небольшую величину, пропорциональную $1/N$, в направлении «сигнала ошибки» $(\mathbf{x}_N - \boldsymbol{\mu}_{\text{ML}}^{(N-1)})$. Обратите внимание, что с увеличением N вклад последовательных точек данных становится меньше.

3.2.9. Гауссовые смеси

Хотя гауссово распределение обладает рядом важных аналитических свойств, его использование для моделирования реальных наборов данных имеет существенные ограничения. Рассмотрим пример на рис. 3.6а. Этот набор данных известен как Олд-Фейтфул (Old Faithful). Он состоит из 272 измерений извержения гейзера Old Faithful в Йеллоустонском национальном парке

(США). Каждое измерение показывает продолжительность извержения в минутах (горизонтальная ось) и время в минутах до следующего извержения (вертикальная ось). Как видно, набор данных образует две доминирующие совокупности, и простое гауссово распределение не в состоянии отразить эту структуру. На рисунке красные кривые соответствуют контурам постоянной плотности вероятности. (a) Одиночное гауссово распределение подогнано к данным с помощью метода максимального правдоподобия. Обратите внимание, что это распределение не отражает двух групп в данных, и большая часть его вероятностной массы находится в центральной области между группами, где данные относительно скучные. (b) Распределение, полученное линейной комбинацией двух гауссовых распределений, также подогнанное по методу максимального правдоподобия, которое дает лучшее представление о данных.

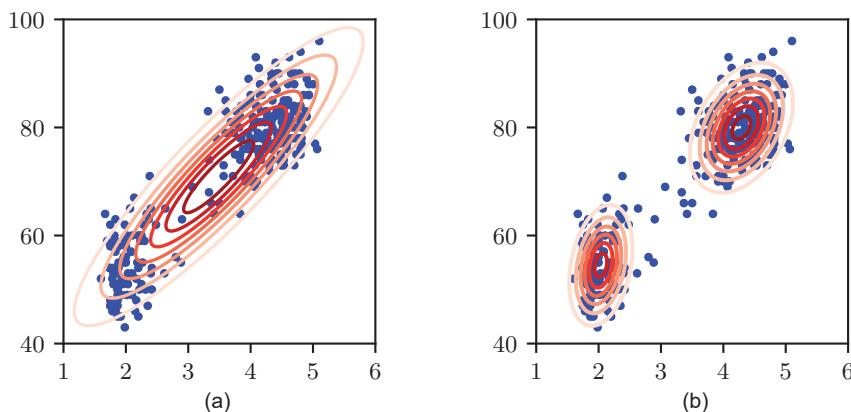
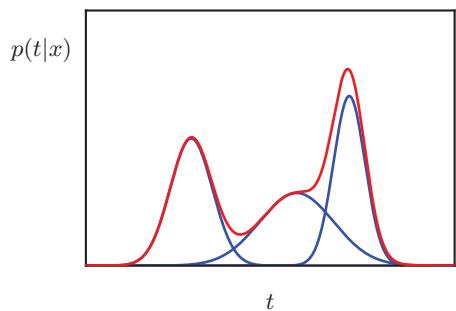


РИС. 3.6 Графики данных *Old Faithful*

Можно было бы ожидать, что суперпозиция двух гауссовых распределений сможет гораздо лучше отразить структуру этого набора данных, и это действительно так, что видно из рис. 3.6б. Такие суперпозиции, образованные линейными комбинациями более простых распределений, таких как гауссовые, могут быть выражены в виде вероятностных моделей, известных как *смешанные распределения* (*mixture distributions*) (см. главу 15). В этом разделе для иллюстрации структуры смешанных моделей будут рассматриваться гауссовые распределения. В более общем случае смешанные модели могут представлять собой линейные комбинации других распределений, например смеси распределений Бернулли для бинарных переменных. На рис. 3.7 видно, что линейная комбинация гауссовых распределений может давать очень сложную структуру распределения плотностей.

РИС. 3.7 Пример распределения гауссовой смеси в одном измерении, где три гауссовых распределения (каждое из которых масштабируется коэффициентом) показаны синим цветом, а их сумма – красным



Используя достаточное количество гауссовых распределений и регулируя их средние значения и ковариации, а также коэффициенты в линейной комбинации, можно аппроксимировать практически любое непрерывное распределение с произвольной точностью.

Соответственно, рассмотрим суперпозицию K гауссовых плотностей вида

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad (3.111)$$

которая называется *гауссовой смесью* (*mixture of Gaussians*). Каждое гауссово распределение плотности $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ называется *компонентой* этой смеси и имеет свое среднее значение $\boldsymbol{\mu}_k$ и ковариацию $\boldsymbol{\Sigma}_k$.

Контурные и поверхностные графики для двумерной гауссовой смеси с тремя компонентами показаны на рис. 3.8. Здесь: (а) контуры постоянной плотности для каждого из компонентов смеси, где три компонента обозначены красным, синим и зеленым, а значения коэффициентов смешивания показаны под каждым компонентом; (б) контуры предельной плотности вероятности $p(\mathbf{x})$ распределения смеси; (с) поверхностный график распределения $p(\mathbf{x})$.

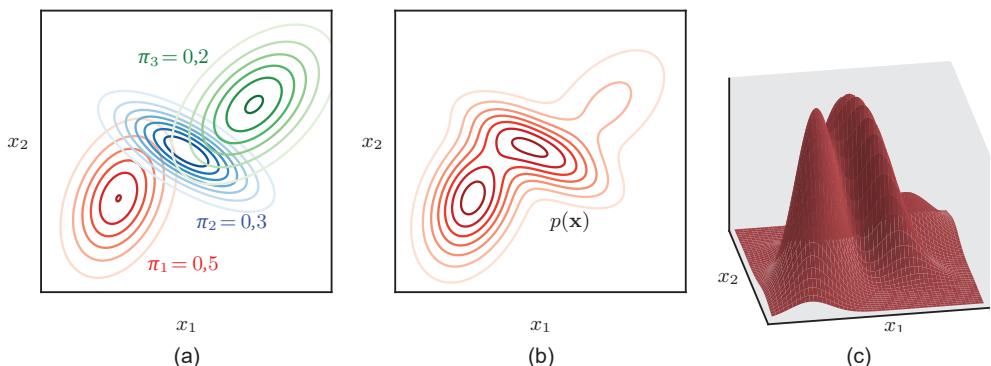


РИС. 3.8 Иллюстрация смеси трех гауссовых распределений в двумерном пространстве

Параметры π_k в (3.111) называются коэффициентами смешивания (mixing coefficients). Если проинтегрировать обе стороны (3.111) по \mathbf{x} и учесть, что и $p(\mathbf{x})$ и отдельные гауссовые компоненты нормализованы, то получится:

$$\sum_{k=1}^K \pi_k = 1. \quad (3.112)$$

Кроме того, если $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) > 0$, то достаточным условием для требования $p(\mathbf{x}) > 0$ является то, что $\pi_k > 0$ для всех k . Комбинируя это с условием (3.112), получаем:

$$0 \leq \pi_k \leq 1. \quad (3.113)$$

Таким образом, можно видеть, что коэффициенты смешивания удовлетворяют требованиям быть вероятностями, и в дальнейшем будет показано, что эта вероятностная интерпретация распределений смесей является очень эффективной (см. главу 15).

Исходя из правил суммы и произведения вероятностей, маргинальная плотность может быть выражена в виде

$$p(\mathbf{x}) = \sum_{k=1}^K p(k)p(\mathbf{x}|k), \quad (3.114)$$

что эквивалентно (3.111), где $\pi_k = p(k)$ можно рассматривать как априорную вероятность выбора k -го компонента, а плотность $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = p(\mathbf{x} | k)$ – как вероятность \mathbf{x} в зависимости от k . Как будет показано в последующих главах, важную роль играют соответствующие апостериорные вероятности $p(k | \mathbf{x})$, которые также называют *обязанностями* (*responsibilities*). По теореме Байеса они определяются как

$$\begin{aligned} r_k(\mathbf{x}) &\equiv p(k | \mathbf{x}) \\ &= \frac{p(k)p(\mathbf{x}|k)}{\sum_l p(l)p(\mathbf{x}|l)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_l \pi_l \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \end{aligned} \quad (3.115)$$

Форма распределения гауссовой смеси определяется параметрами $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ и $\boldsymbol{\Sigma}$, где используются обозначения $\boldsymbol{\pi} \equiv \{\pi_1, \dots, \pi_K\}$, $\boldsymbol{\mu} \equiv \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$ и $\boldsymbol{\Sigma} \equiv \{\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K\}$. Одним из способов задания значений этих параметров является использование метода максимального правдоподобия. Из (3.111) следует, что логарифм функции правдоподобия имеет вид:

$$\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}, \quad (3.116)$$

где $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. Теперь видно, что ситуация намного сложнее, чем с одним гауссом, из-за суммирования по k внутри логарифма. В результате решение максимального правдоподобия для параметров больше не имеет аналитического решения в замкнутой форме. Один из подходов к максимизации функции правдоподобия заключается в использовании итерационных методов численной оптимизации. В качестве альтернативы можно использовать эффективный механизм *максимизации ожидания* (*expectation maximization*) (см. главу 15), который широко применяется при работе с различными глубокими генеративными моделями.

3.3. Периодические переменные

Несмотря на большую практическую значимость гауссовых распределений, как самих по себе, так и в качестве составных частей более сложных вероятностных моделей, существуют ситуации, когда они оказываются непригодными в качестве моделей распределения плотности для непрерывных переменных. Одним из таких возможных случаев в практическом применении являются периодические переменные.

Примером периодической переменной является направление ветра в определенной географической точке. Например, можно измерить направление ветра в нескольких местах и обобщить эти данные с помощью параметрического распределения. Другой пример – календарное время, когда необходимо смоделировать количественные показатели, считающиеся периодическими в течение 24 часов или годового цикла. Такие величины удобно представлять с помощью угловых (полярных) координат $0 \leq \theta < 2\pi$.

Может возникнуть желание рассматривать периодические переменные при помощи выбора некоторого направления в качестве начала координат и последующего применения обычного распределения, например гауссова. Однако такой подход приведет к результатам, которые будут сильно зависеть от произвольного выбора начала координат. Предположим, например, что имеются два наблюдения в точках $\theta_1 = 1^\circ$ и $\theta_2 = 359^\circ$, и для их моделирования используется стандартное одномерное гауссово распределение. Если поместить начало координат в 0° , то выборочное среднее значение этого набора данных будет равно 180° со стандартным отклонением 179° ; если же поместить начало координат в 180° , то среднее значение будет равно 0° , а стандартное отклонение составит 1° . Очевидно, что для периодических переменных требуется разработать специальный метод.

3.3.1. Распределение фон Мизеса

Рассмотрим задачу оценки среднего значения набора наблюдений $\mathcal{D} = \{\theta_1, \dots, \theta_N\}$ периодической переменной θ , где θ измеряется в радианах. Ранее было отмечено, что простое среднее значение $(\theta_1 + \dots + \theta_N)/N$ будет сильно зависеть

от координат. Чтобы найти инвариантную меру среднего значения, нужно обратить внимание на то, что наблюдения можно рассматривать как точки на единичном круге и, следовательно, описывать двумерными единичными векторами $\mathbf{x}_1, \dots, \mathbf{x}_N$, где $\|\mathbf{x}_n\| = 1$ для $n = 1, \dots, N$, как показано на рис. 3.9. Вместо этого можно усреднить векторы $\{\mathbf{x}_n\}$, чтобы получить

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n, \quad (3.117)$$

и затем найти соответствующий угол $\bar{\theta}$ этого среднего значения. Очевидно, что такое определение гарантирует, что местоположение среднего значения не зависит от начала угловой координаты. Обратите внимание, что $\bar{\mathbf{x}}$ обычно лежит внутри единичного круга. Декартовы координаты наблюдений задаются в виде $\mathbf{x}_n = (\cos \theta_n, \sin \theta_n)$, и декартовы координаты выборочного среднего значения можно записать в виде $\bar{\mathbf{x}} = (\bar{r} \cos \bar{\theta}, \bar{r} \sin \bar{\theta})$. Подстановка в (3.117) и уравнивание компонентов \mathbf{x}_1 и \mathbf{x}_2 дает:

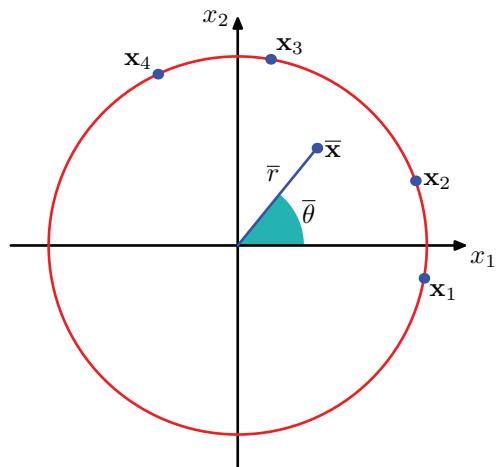
$$\bar{x}_1 = \bar{r} \cos \bar{\theta} = \frac{1}{N} \sum_{n=1}^N \cos \theta_n, \quad \bar{x}_2 = \bar{r} \sin \bar{\theta} = \frac{1}{N} \sum_{n=1}^N \sin \theta_n. \quad (3.118)$$

Получив соотношение и используя тождество $\tan \theta = \sin \theta / \cos \theta$, можно определить $\bar{\theta}$, чтобы получить

$$\bar{\theta} = \tan^{-1} \left(\frac{\sum_n \sin \theta_n}{\sum_n \cos \theta_n} \right). \quad (3.119)$$

Далее будет показано, как этот результат возникает естественным образом в виде оценки максимального правдоподобия.

РИС. 3.9 Иллюстрация представления значений θ_n периодической переменной в виде двумерных векторов \mathbf{x}_n на единичном круге. Также показано среднее значение $\bar{\mathbf{x}}$ этих векторов



Для начала необходимо определить периодическое обобщение гауссова распределения, которое называется распределением фон Мизеса (von Mises distribution). В данном случае рассмотрим только одномерные распределения, хотя аналогичные периодические распределения встречаются и на гиперсферах произвольной размерности (Mardia and Jupp, 2000).

Согласно условию рассмотрим распределения $p(\theta)$ с периодом 2π . Любая плотность распределения вероятности $p(\theta)$, определенная по θ , должна быть не только неотрицательной и интегрируемой по единице, но и периодической. Таким образом, значение $p(\theta)$ должно удовлетворять трем условиям:

$$p(\theta) > 0, \quad (3.120)$$

$$\int_0^{2\pi} p(\theta) d\theta = 1, \quad (3.121)$$

$$p(\theta + 2\pi) = p(\theta). \quad (3.122)$$

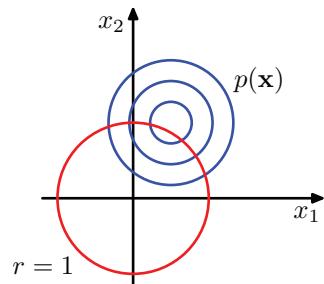
Из (3.122) следует, что $p(\theta + M2\pi) = p(\theta)$ для любого целого числа M .

Гауссово распределение, удовлетворяющее этим трем свойствам, можно с легкостью получить следующим образом. Рассмотрим гауссово распределение по двум переменным $\mathbf{x} = (x_1, x_2)$ со средним значением $\boldsymbol{\mu} = (\mu_1, \mu_2)$ и ковариационной матрицей $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, где \mathbf{I} – матрица тождества 2×2 , чтобы

$$p(x_1, x_2) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{(x_1 - \mu_1)^2 + (x_2 - \mu_2)^2}{2\sigma^2}\right\}. \quad (3.123)$$

Контуры постоянной $p(\mathbf{x})$ представляют собой окружности, как показано на рис. 3.10.

РИС. 3.10 Распределение фон Мизеса выводится из двумерного гауссова распределения (3.123), контуры плотности которого показаны синим цветом, а условия на единичной окружности – красным



Теперь представим, что значение этого распределения лежит вдоль окружности фиксированного радиуса. Тогда по определению это распределение будет периодическим, хотя и не будет нормализованным. Определить форму этого распределения можно путем преобразования декартовых координат (x_1, x_2) в полярные (r, θ) так, чтобы

$$x_1 = r \cos \theta, \quad x_2 = r \sin \theta. \quad (3.124)$$

Кроме того, переведем среднее значение μ в полярные координаты следующим образом:

$$\mu_1 = r_0 \cos \theta_0, \quad \mu_2 = r_0 \sin \theta_0. \quad (3.125)$$

Далее подставим эти преобразования в двумерное гауссово распределение (3.123), а затем поставим условие на единичной окружности $r = 1$, с учетом того что нас интересует только зависимость по θ . Ориентируясь на экспоненту в гауссовом распределении, имеем

$$\begin{aligned} & -\frac{1}{2\sigma^2} \left\{ (r \cos \theta - r_0 \cos \theta_0)^2 + (r \sin \theta - r_0 \sin \theta_0)^2 \right\} \\ &= -\frac{1}{2\sigma^2} \{1 + r_0^2 - 2r_0 \cos \theta \cos \theta_0 - 2r_0 \sin \theta \sin \theta_0\} \\ &= \frac{r_0}{\sigma^2} \cos(\theta - \theta_0) = \text{const}, \end{aligned} \quad (3.126)$$

где const обозначает члены, не зависящие от θ . Здесь используются следующие тригонометрические тождества:

$$\cos^2 A + \sin^2 A = 1, \quad (3.127)$$

$$\cos A \cos B + \sin A \sin B = \cos(A - B). \quad (3.128)$$

Если теперь определить $m = r_0/\sigma_2$, то получится окончательное выражение для распределения $p(\theta)$ вдоль единичной окружности $r = 1$ в виде

$$p(\theta|\theta_0, m) = \frac{1}{2\pi I_0(m)} \exp\{m \cos(\theta - \theta_0)\}, \quad (3.129)$$

которое называется *распределением фон Мизеса* (*von Mises distribution*), или *круговым нормальным распределением* (*circular normal*). Здесь параметр θ_0 соответствует среднему значению распределения, а m , известный как *параметр концентрации* (*concentration parameter*), аналогичен обратной дисперсии (т. е. точности) для гауссова распределения. Коэффициент нормализации в (3.129) выражается в виде $I_0(m)$ и представляет собой модифицированную функцию Бесселя первого рода нулевого порядка (Abramowitz and Stegun, 1965), которая определяется как

$$I_0(m) = \frac{1}{2\pi} \int_0^{2\pi} \exp\{m \cos \theta\} d\theta. \quad (3.130)$$

При больших значениях m (см. упражнение 3.31) характер распределения становится близким к гауссовому. Распределение фон Мизеса показано на рис. 3.11, а функция $I_0(m)$ – на рис. 3.12.

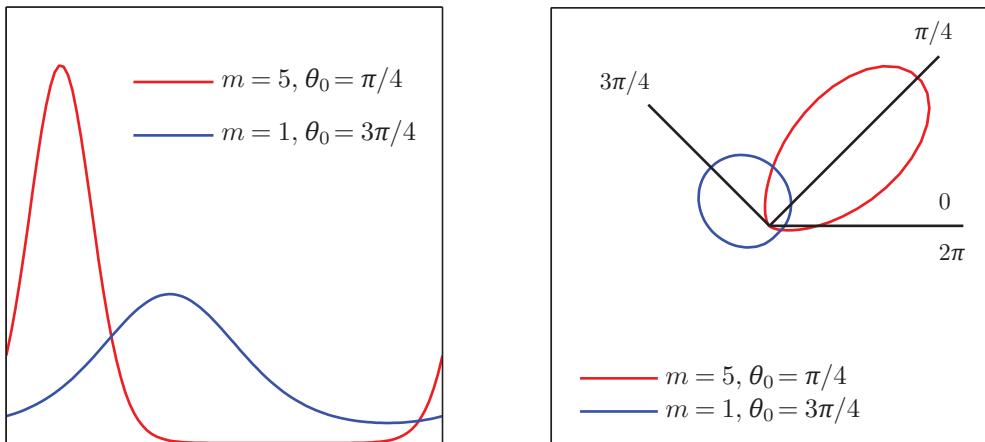


РИС. 3.11 Распределение фон Мизеса для двух различных значений параметров в виде декартовой диаграммы слева и соответствующей полярной диаграммы справа

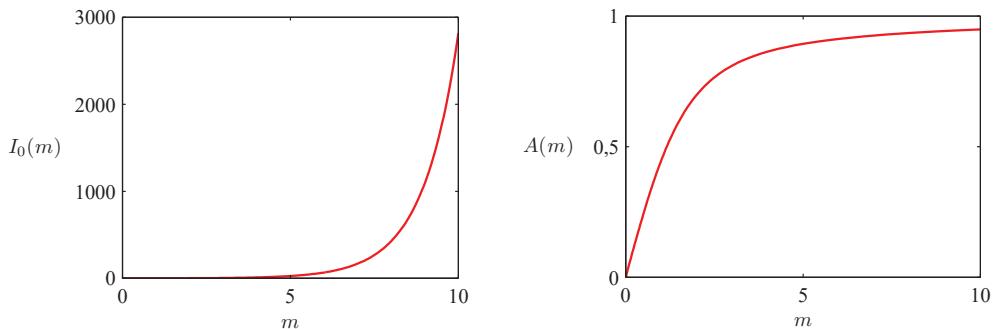


РИС. 3.12 График функции Бесселя $I_0(m)$, определенной в (3.130), вместе с функцией $A(m)$, определенной в (3.136)

Теперь рассмотрим оценки максимального правдоподобия параметров θ_0 и m для распределения фон Мизеса. Функция логарифмического правдоподобия имеет вид:

$$\ln p(\mathcal{D}|\theta_0, m) = -N \ln(2\pi) - N \ln I_0(m) + m \sum_{n=1}^N \cos(\theta_n - \theta_0). \quad (3.131)$$

Если производную по θ_0 приравнять к нулю, то получится

$$\sum_{n=1}^N \sin(\theta_n - \theta_0) = 0. \quad (3.132)$$

Для определения θ_0 воспользуемся тригонометрическим тождеством

$$\sin(A - B) = \cos B \sin A - \cos A \sin B, \quad (3.133)$$

из которого (см. упражнение 3.32) получается

$$\theta_0^{\text{ML}} = \tan^{-1} \left\{ \frac{\sum_n \sin \theta_n}{\sum_n \cos \theta_n} \right\}, \quad (3.134)$$

что соответствует результату (3.119), полученному ранее для среднего значения наблюдений, рассматриваемых в двумерном декартовом пространстве.

Точно так же, максимизируя (3.131) относительно m и используя $I'_0(m) = I_1(m)$ (Abramowitz and Stegun, 1965), получаем:

$$A(m_{\text{ML}}) = \frac{1}{N} \sum_{n=1}^N \cos(\theta_n - \theta_0^{\text{ML}}), \quad (3.135)$$

где на место максимального правдоподобия подставлено решение θ_0^{ML} (напомним, что речь идет о совместной оптимизации по θ и m), и в этом случае определяем:

$$A(m) = \frac{I_1(m)}{I_0(m)}. \quad (3.136)$$

Функция $A(m)$ изображена на рис. 3.12. Используя тригонометрическое тождество (3.128), можно записать (3.135) в виде

$$A(m_{\text{ML}}) = \left(\frac{1}{N} \sum_{n=1}^N \cos \theta_n \right) \cos \theta_0^{\text{ML}} + \left(\frac{1}{N} \sum_{n=1}^N \sin \theta_n \right) \sin \theta_0^{\text{ML}}. \quad (3.137)$$

Правая часть (3.137) вычисляется достаточно легко, а функция $A(m)$ может быть преобразована количественно. Одним из ограничений распределения фон Мизеса является его унимодальность. Формируя *смешанные* распределения фон Мизеса, можно получить гибкую структуру для моделирования периодических переменных, которая может работать с мультимодальностью.

Для полноты картины рассмотрим некоторые альтернативные методы построения периодических распределений. Самый простой подход заключается в использовании гистограммы наблюдений, когда угловая координата делится на фиксированные ячейки. Этот подход обладает достоинством простоты и гибкости, но в то же время испытывает существенные ограничения, как будет показано при более подробном обсуждении гистограммных методов (см. раздел 3.5). Другой подход, подобно распределению фон Мизеса, начинается с гауссова распределения в евклидовом пространстве, но теперь маргинализация производится не по условию, а по единичному кругу (Mardia and Jupp, 2000). Однако это приводит к более сложным формам распределения и далее не рассматривается. Наконец, любое действительное распределение на вещественной оси (например, гауссово) можно превратить в периодическое распределение, отобразив последовательные интервалы шириной 2π на периодическую переменную $(0, 2\pi)$, что соответствует

«оборачиванию» вещественной оси вокруг единичной окружности. И вновь полученное распределение оказывается более сложным для обработки, чем распределение фон Мизеса.

3.4. Семейство экспоненциальных распределений

Распределения вероятностей, рассмотренные до сих пор в этой главе (за исключением смешанных моделей), являются конкретными примерами обширного класса распределений, называемого *экспоненциальным семейством* (*exponential family*) (Duda and Hart, 1973; Bernardo and Smith, 1994). Семейство экспоненциальных распределений обладает многими общими важными свойствами, и будет полезно обсудить эти свойства в общих чертах.

Экспоненциальное семейство распределений по x , заданных параметрами η , определяется как множество распределений вида

$$p(x|\eta) = h(x)g(\eta)\exp\{\eta^T u(x)\}, \quad (3.138)$$

где x может быть скаляром или вектором, а также может быть дискретным или непрерывным. Здесь η – это собственные параметры распределения, а $u(x)$ – это некоторая функция от x . Функцию $g(\eta)$ можно интерпретировать как коэффициент, обеспечивающий нормализацию распределения, и, следовательно, она удовлетворяет следующему условию:

$$g(\eta) \int h(x)\exp\{\eta^T u(x)\}dx = 1, \quad (3.139)$$

где интегрирование заменяется суммированием в том случае, если x является дискретной переменной.

Для начала рассмотрим несколько примеров распределений, введенных ранее в этой главе, и покажем, что они действительно являются членами экспоненциального семейства.

Сначала рассмотрим распределение Бернулли:

$$p(x|\mu) = \text{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x}. \quad (3.140)$$

Выражая правую часть как экспоненту логарифма, получаем:

$$\begin{aligned} p(x|\mu) &= \exp\{x \ln \mu + (1-x) \ln(1-\mu)\} \\ &= (1-\mu) \exp\left\{\ln\left(\frac{\mu}{1-\mu}\right)x\right\}. \end{aligned} \quad (3.141)$$

Сравнение с (3.138) позволяет определить

$$\eta = \ln\left(\frac{\mu}{1-\mu}\right), \quad (3.142)$$

что позволяет определить μ и получить $\mu = \sigma(\eta)$, где

$$\sigma(\eta) = \frac{1}{1 + \exp(-\eta)} \quad (3.143)$$

называется логистической сигмоидной функцией (*logistic sigmoid function*). Таким образом, распределение Бернулли можно записать с помощью стандартного представления (3.138) в виде

$$p(x|\eta) = \sigma(-\eta)\exp(\eta x), \quad (3.144)$$

где используется $1 - \sigma(\eta) = \sigma(-\eta)$, и это легко доказать из (3.143). Сравнение с (3.138) показывает, что

$$u(x) = x, \quad (3.145)$$

$$h(x) = 1, \quad (3.146)$$

$$g(\eta) = \sigma(-\eta). \quad (3.147)$$

Далее рассмотрим мультиномиальное распределение, для которого в случае отдельного наблюдения \mathbf{x} оно имеет вид:

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^M \mu_k^{x_k} = \exp \left\{ \sum_{k=1}^M x_k \ln \mu_k \right\}, \quad (3.148)$$

где $\mathbf{x} = (x_1, \dots, x_M)^T$. И вновь это можно записать в стандартном представлении (3.138), так что

$$p(\mathbf{x}|\boldsymbol{\eta}) = \exp(\boldsymbol{\eta}^T \mathbf{x}), \quad (3.149)$$

где $\eta_k = \ln \mu_k$, и определяется $\boldsymbol{\eta} = (\eta_1, \dots, \eta_M)^T$. Сравнивая с (3.138), вновь получаем:

$$\mathbf{u}(\mathbf{x}) = \mathbf{x}, \quad (3.150)$$

$$h(\mathbf{x}) = 1, \quad (3.151)$$

$$g(\boldsymbol{\eta}) = 1. \quad (3.152)$$

Обратите внимание, что параметры η_k не являются независимыми, поскольку на параметры μ_k накладывается ограничение

$$\sum_{k=1}^M x_k = 1, \quad (3.153)$$

так что при любом $M - 1$ из параметров μ_k значение оставшегося параметра является фиксированным. При определенных обстоятельствах будет удобнее снять это ограничение, выразив распределение в виде только $M - 1$ параметров. Этого можно достичь с помощью соотношения (3.153), исключив μ_M

и выражив его в виде оставшихся $\{\mu_k\}$, где $k = 1, \dots, M - 1$, и, таким образом, остается $M - 1$ параметров. Следует отметить, что эти оставшиеся параметры по-прежнему подчиняются ограничениям

$$0 \leq \mu_k \leq 1, \quad \sum_{k=1}^{M-1} \mu_k \leq 1. \quad (3.154)$$

Используя ограничение (3.153), мультиномиальное распределение в этом представлении приобретает вид:

$$\begin{aligned} & \exp \left\{ \sum_{k=1}^M x_k \ln \mu_k \right\} \\ &= \exp \left\{ \sum_{k=1}^{M-1} x_k \ln \mu_k + \left(1 - \sum_{k=1}^{M-1} x_k \right) \ln \left(1 - \sum_{k=1}^{M-1} \mu_k \right) \right\} \\ &= \exp \left\{ \sum_{k=1}^{M-1} x_k \ln \left(\frac{\mu_k}{1 - \sum_{j=1}^{M-1} \mu_j} \right) + \ln \left(1 - \sum_{k=1}^{M-1} \mu_k \right) \right\}. \end{aligned} \quad (3.155)$$

Теперь определим

$$\ln \left(\frac{\mu_k}{1 - \sum_j \mu_j} \right) = \eta_k, \quad (3.156)$$

что можно делать для μ_k , сначала просуммировав обе стороны по k , а затем перегруппировав и сделав обратную подстановку, что дает

$$\mu_k = \frac{\exp(\eta_k)}{1 + \sum_j \exp(\eta_j)}. \quad (3.157)$$

Такое отображение называется функцией *softmax* (Или функцией «мягкого максимума». – Прим. перев.), а также *нормализованной экспоненциальной* (*normalized exponential*) функцией. Таким образом, в этом случае мультиномиальное распределение принимает вид:

$$p(\mathbf{x}|\boldsymbol{\eta}) = \left(1 + \sum_{k=1}^{M-1} \exp(\eta_k) \right)^{-1} \exp(\boldsymbol{\eta}^T \mathbf{x}). \quad (3.158)$$

Это стандартная форма экспоненциального семейства с вектором параметров $\boldsymbol{\eta} = (\eta_1, \dots, \eta_{M-1})^T$, где

$$\mathbf{u}(\mathbf{x}) = \mathbf{x}, \quad (3.159)$$

$$h(\mathbf{x}) = 1, \quad (3.160)$$

$$g(\boldsymbol{\eta}) = \left(1 + \sum_{k=1}^{M-1} \exp(\eta_k) \right)^{-1}. \quad (3.161)$$

Наконец, рассмотрим гауссово распределение. Для одномерного гауссова распределения это

$$p(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\} \quad (3.162)$$

$$= \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2} + \frac{\mu}{\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2\right\}, \quad (3.163)$$

что после несложных преобразований можно привести к виду стандартного экспоненциального семейства (см упражнение 3.35) в форме (3.138) при

$$\boldsymbol{\eta} = \begin{pmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{pmatrix}, \quad (3.164)$$

$$\mathbf{u}(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix}, \quad (3.165)$$

$$h(\mathbf{x}) = (2\pi)^{-1/2}, \quad (3.166)$$

$$g(\boldsymbol{\eta}) = (-2\eta_2)^{1/2} \exp\left(\frac{\eta_1^2}{4\eta_2}\right). \quad (3.167)$$

Наконец, иногда будет использоваться ограниченная форма (3.138), в которой выбирается $\mathbf{u}(\mathbf{x}) = \mathbf{x}$. Однако ее можно несколько обобщить за счет того, что если $f(\mathbf{x})$ – это нормализованная плотность, тогда

$$\frac{1}{s} f\left(\frac{1}{s}\mathbf{x}\right) \quad (3.168)$$

также является нормализованной плотностью, где $s > 0$ является параметром масштабирования. В результате их комбинирования получается ограниченный набор условных плотностей класса экспоненциального семейства вида

$$p(\mathbf{x}|\boldsymbol{\lambda}_k, s) = \frac{1}{s} h\left(\frac{1}{s}\mathbf{x}\right) g(\boldsymbol{\lambda}_k) \exp\left\{\frac{1}{s}\boldsymbol{\lambda}_k^\top \mathbf{x}\right\}. \quad (3.169)$$

Обратите внимание, что для каждого класса допускается наличие собственного вектора параметров $\boldsymbol{\lambda}_k$, но предполагается, что классы имеют один и тот же масштабный параметр s .

3.4.1. Достаточная статистика

Теперь рассмотрим задачу оценки вектора параметров $\boldsymbol{\eta}$ общего распределения экспоненциального семейства (3.138) с помощью метода максимального правдоподобия. Взяв градиент обеих сторон (3.139) относительно $\boldsymbol{\eta}$, получим:

$$\nabla g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp\{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})\} d\mathbf{x} + g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp\{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})\} \mathbf{u}(\mathbf{x}) d\mathbf{x} = 0. \quad (3.170)$$

Перегруппировка и повторное использование (3.139) дают

$$-\frac{1}{g(\boldsymbol{\eta})} \nabla g(\boldsymbol{\eta}) = g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp\{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})\} \mathbf{u}(\mathbf{x}) d\mathbf{x} = \mathbb{E}[\mathbf{u}(\mathbf{x})]. \quad (3.171)$$

Таким образом, получается

$$-\nabla \ln g(\boldsymbol{\eta}) = \mathbb{E}[\mathbf{u}(\mathbf{x})]. \quad (3.172)$$

Обратите внимание, что ковариация $\mathbf{u}(\mathbf{x})$ может быть выражена через вторые производные $g(\boldsymbol{\eta})$ (см. упражнение 3.36), и точно так же для моментов более высокого порядка. Таким образом, при условии, что распределение из семейства экспоненциальных удастся нормализовать, всегда можно определить его моменты простым дифференцированием.

Теперь рассмотрим набор независимых идентично распределенных данных, обозначенных $\mathbf{X} = \{x_1, \dots, x_n\}$, для которого функция правдоподобия имеет вид:

$$p(\mathbf{X} | \boldsymbol{\eta}) = \left(\prod_{n=1}^N h(\mathbf{x}_n) \right) g(\boldsymbol{\eta})^N \exp \left\{ \boldsymbol{\eta}^T \sum_{n=1}^N \mathbf{u}(\mathbf{x}_n) \right\}. \quad (3.173)$$

Установливая градиент $\ln p(\mathbf{X} | \boldsymbol{\eta})$ относительно $\boldsymbol{\eta}$ равным нулю, получаем следующее условие, которому должна удовлетворять оценка максимального правдоподобия $\boldsymbol{\eta}_{ML}$:

$$-\nabla \ln g(\boldsymbol{\eta}_{ML}) = \frac{1}{N} \sum_{n=1}^N \mathbf{u}(\mathbf{x}_n), \quad (3.174)$$

которая вполне может быть решена для получения $\boldsymbol{\eta}_{ML}$. Как видно, решение для оценки максимального правдоподобия зависит от данных только за счет параметра $\sum_n \mathbf{u}(\mathbf{x}_n)$, который, соответственно, называется *достаточной статистикой* (*sufficient statistic*) распределения (3.138). В этом случае нет необходимости хранить весь набор данных, достаточно знать лишь значение достаточной статистики. Например, для распределения Бернулли функция $\mathbf{u}(x)$ зависит только от x , и потому необходимо сохранить только сумму точек данных $\{x_n\}$, тогда как для гауссова распределения $\mathbf{u}(x) = (x, x^2)^T$, и потому необходимо сохранить и сумму $\{x_n\}$, и сумму $\{x_n^2\}$.

Если рассматривать предел $N \rightarrow \infty$, то правая часть (3.174) примет вид $\mathbb{E}[\mathbf{u}(\mathbf{x})]$, и, сравнивая с (3.172), можно увидеть, что в этом пределе $\boldsymbol{\eta}_{ML}$ будет равно истинному значению $\boldsymbol{\eta}$.

3.5. Непараметрические методы

В этой главе основное внимание уделяется использованию распределений вероятностей с определенными функциональными формами и небольшим

числом параметров, значения которых необходимо определить по набору данных. Это называется параметрическим (parametric) подходом к моделированию распределения плотности. Важным ограничением этого подхода является тот факт, что выбранная плотность может быть неудачной моделью распределения, генерирующего данные, в результате чего эффективность прогнозирования может оказаться невысокой. Например, если процесс, генерирующий данные, является мультимодальным, то этот аспект распределения невозможно отразить с помощью гауссова распределения, которое в любом случае является унимодальным. В этом заключительном разделе будут рассмотрены некоторые непараметрические (nonparametric) подходы к оценке распределения плотности, которые не содержат особых предположений о форме распределения.

3.5.1. Гистограммы

Начнем с обсуждения гистограммных методов оценки плотности, которые уже рассматривались в контексте маргинальных и условных распределений с примером на рис. 2.5, а также в контексте центральной предельной теоремы с примером на рис. 3.2. Здесь же будут более подробно рассмотрены свойства гистограммных моделей распределения плотности с акцентом на ситуации с одной непрерывной переменной x . Для стандартных гистограмм достаточно разбить x на отдельные ячейки шириной Δ_i и подсчитать число n_i наблюдений x , попавших в ячейку i . Чтобы превратить это число в нормализованную плотность распределения вероятности, достаточно разделить на общее число N наблюдений и ширину ячеек Δ_i , чтобы получить значения вероятности для каждой ячейки:

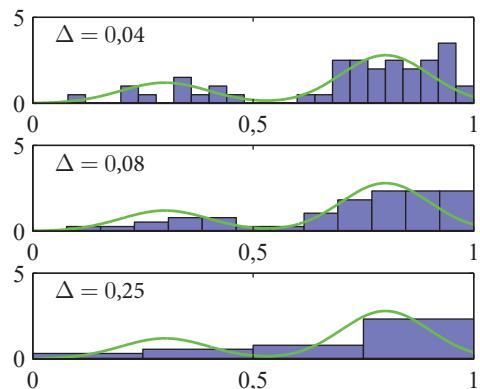
$$p_i = \frac{n_i}{N\Delta_i}, \quad (3.175)$$

для которых, как легко убедиться, $\int p(x)dx = 1$. Таким образом, получается модель для плотности $p(x)$, которая постоянна по ширине каждой ячейки. Часто ячейки выбираются одинаковой ширины $\Delta_i = \Delta$.

На рис. 3.13 показан пример оценки плотности распределения гистограммы. Здесь данные взяты из распределения, соответствующего зеленой кривой, которое образовано из смеси двух гауссовых распределений. Как видно, приведены три примера оценки плотности гистограммы, соответствующие трем различным вариантам ширины ячейки Δ . Оценки плотности распределения гистограммы, основанные на (3.175) с общей шириной ячейки Δ , показаны для различных значений Δ . Как видно, при очень малом значении Δ (верхний рисунок) результирующая модель распределения плотности получается весьма скачкообразной, с большим количеством структур, отсутствующих в базовом распределении, на основе которого был получен набор данных. И наоборот, если значение Δ слишком велико (нижний рисунок), то модель получается слишком гладкой и, следовательно, не отражает

бимодальное свойство зеленой кривой. Наилучшие результаты получаются при некотором промежуточном значении Δ (средний рисунок). В принципе, модель плотности гистограммы также зависит от выбора расположения краев ячеек, хотя, как правило, это гораздо менее важно, чем ширина ячеек Δ .

РИС. 3.13 Иллюстрация гистограммного подхода к оценке плотности, где набор данных из 50 точек генерируется из распределения, показанного зеленой кривой. Оценки плотности по гистограмме, основанные на (3.175) с общей шириной ячейки Δ , показаны для различных значений Δ



Обратите внимание, что метод гистограмм обладает таким свойством (в отличие от методов, которые будут рассмотрены ниже), что после вычисления гистограммы сам набор данных можно отбросить, что может быть полезно, если набор данных достаточно велик. Кроме того, гистограммный подход удобно использовать при последовательном поступлении точек данных.

На практике метод гистограмм может быть полезен для быстрой визуализации данных в одном или двух измерениях, но не подходит для большинства приложений оценки распределения плотности. Одна из очевидных проблем заключается в том, что оцениваемая плотность будет иметь разрывы, обусловленные краями ячеек, а не каким-либо свойством основного распределения, на основе которого были получены данные. Основным ограничением гистограммного подхода является его масштабирование с размерностью. Если разделить каждую переменную в D -мерном пространстве на M ячеек, то общее количество этих ячеек будет равно M^D . Это экспоненциальное масштабирование с увеличением D является примером так называемого *проклятия размерности* (*curse of dimensionality*) (см. раздел 6.1.1). В пространстве с высокой размерностью количество данных, необходимых для получения значимых данных для оценки локальной плотности вероятности, будет непомерно большим.

Тем не менее гистограммный подход к оценке распределения плотности дает возможность извлечь два важных урока. Во-первых, для оценки плотности вероятности в определенной точке необходимо рассматривать точки данных в пределах какой-либо локальной близости от этой точки. Обратите внимание, что концепция локальности предполагает наличие некоторой меры измерения расстояния, и в данном случае используется евклидово

расстояние. Для гистограмм это свойство локальности определяется ячейками, при этом существует соответствующий параметр «сглаживания», описывающий пространственную протяженность локальной области, а в данном случае – это ширина ячейки. Во-вторых, для получения хороших результатов значение параметра сглаживания не должно быть ни слишком большим, ни слишком маленьким. Это напоминает выбор сложности модели в полиномиальной регрессии, где степень M полинома, или, иначе, значение λ параметра регуляризации, было оптимальным для некоторого промежуточного значения – ни слишком большого, ни слишком маленького. Вооружившись этими знаниями, перейдем к рассмотрению двух широко используемых непараметрических методов оценки плотности, а именно метода оценки ядра и метода ближайших соседей, которые отличаются лучшим качеством масштабирования размерности по сравнению с простой гистограммной моделью.

3.5.2. Ядерная оценка плотности

Предположим, что измерения ведутся по неизвестной плотности распределения вероятностей $p(\mathbf{x})$ в некотором D -мерном пространстве, которое условно будем считать евклидовым, и требуется оценить значение $p(\mathbf{x})$. Опираясь на ранее представленное определение локальности, рассмотрим некоторую небольшую область \mathcal{R} , содержащую \mathbf{x} . Вероятностная масса, связанная с этой областью, задается как

$$P = \int_{\mathcal{R}} p(\mathbf{x}) d\mathbf{x}. \quad (3.176)$$

Теперь предположим, что имеется набор данных, состоящий из N наблюдений, взятых из $p(\mathbf{x})$. Поскольку каждая точка данных с вероятностью P попадает в область \mathcal{R} , общее число K точек, лежащих внутри \mathcal{R} , будет распределено в соответствии с биномиальным распределением (см. раздел 3.1.2):

$$\text{Bin}(K|N, P) = \frac{N!}{K!(N-K)!} P^K (1-P)^{N-K}. \quad (3.177)$$

Используя (3.11), можно убедиться, что среднее значение части точек, находящихся внутри этой области, равно $\mathbb{E}[K/N] = P$, и точно так же, используя (3.12), можно убедиться, что дисперсия этого среднего значения равна $\text{var}[K/N] = P(1 - P)/N$. При больших значениях N это распределение будет иметь резкий пик в окрестностях среднего значения, и поэтому

$$K \simeq NP. \quad (3.178)$$

Тем не менее если также предположить, что область \mathcal{R} достаточно мала, так что плотность вероятности $p(\mathbf{x})$ примерно постоянна в этой области, тогда получится

$$P \simeq p(\mathbf{x})V, \quad (3.179)$$

где V обозначает размер \mathcal{R} . Комбинируя (3.178) и (3.179), получаем оценку плотности в виде

$$p(\mathbf{x}) = \frac{K}{NV}. \quad (3.180)$$

Обратите внимание, что корректность (3.180) зависит от двух противоречивых предположений, а именно: область \mathcal{R} достаточно мала и плотность в ней приблизительно постоянна, но при этом достаточно велика (по отношению к ее величине), чтобы число K точек, попадающих внутрь области, было достаточным для появления резкого пика биномиального распределения.

Результат (3.180) можно использовать в двух вариантах. Либо можно зафиксировать K и определить значение V по данным, и в этом случае используется *методика K-ближайших соседей* (*K-nearest-neighbour technique*), о которой будет сказано ниже, либо зафиксировать V и определить K по данным, и тогда – речь об использовании *ядерной методики* (*kernel approach*). В общем случае можно показать, что и K -оценка плотности ближайшего соседа, и ядерная оценка плотности сходятся к истинной плотности распределения вероятности в пределе $N \rightarrow \infty$ при условии, что V уменьшается с N , а K растет с N с соответствующей интенсивностью (Duda and Hart, 1973).

Начнем с подробного обсуждения ядерной методики. Для начала рассмотрим область \mathcal{R} в качестве небольшого гиперкуба с центром в точке \mathbf{x} , для которой требуется определить плотность распределения вероятности. Для подсчета числа K точек, попадающих в эту область, удобно задать функцию

$$k(\mathbf{u}) = \begin{cases} 1, & |\mathbf{u}_i| \leq 1/2, \\ 0 & \text{в ином случае,} \end{cases} \quad i = 1, \dots, D, \quad (3.181)$$

которая обозначает элементарный куб с центром в начале координат. Функция $k(\mathbf{u})$ является примером *ядерной функции* (*kernel function*), и в данном контексте ее также называют *окном Парзена* (*Parzen window*). Из (3.181) следует, что значение $k((\mathbf{x} - \mathbf{x}_n)/h)$ будет равно 1, если точка данных \mathbf{x}_n лежит внутри куба со стороной h и центром в \mathbf{x} , и нулю в противном случае. Таким образом, общее количество точек данных, лежащих внутри этого куба, будет равно:

$$K = \sum_{n=1}^N k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right). \quad (3.182)$$

Подставив это выражение в (3.180), получим результат для расчетной плотности в точке \mathbf{x} :

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right), \quad (3.183)$$

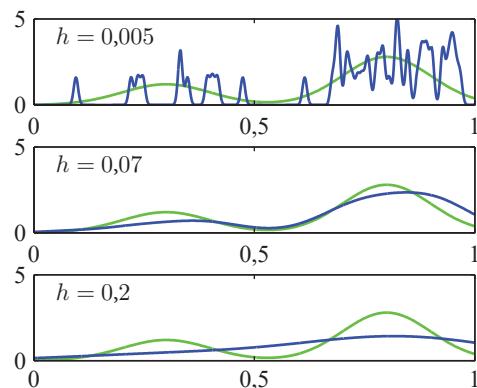
где $V = h^D$ – это объем гиперкуба со стороной h в D измерениях. Используя симметрию функции $k(\mathbf{u})$, теперь можно интерпретировать это уравнение не только для единичного куба с центром \mathbf{x} , а как сумму по N кубам с центрами в N точках данных \mathbf{x}_n .

В таком виде ядерная оценка плотности (3.183) будет подвержена одной из тех же проблем, что и гистограммный метод, а именно присутствию искусственных разрывов, в данном случае на границах кубов. Более плавную модель распределения плотности можно получить при выборе более плавной ядерной функции, и распространенным выбором является использование гауссова распределения, что приводит к следующей ядерной модели распределения плотности:

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{(2\pi h^2)^{D/2}} \exp \left\{ -\frac{\|\mathbf{x} - \mathbf{x}_n\|^2}{2h^2} \right\}, \quad (3.184)$$

где h соответствует стандартному отклонению компонентов гауссова распределения. Таким образом, наша модель плотности получается путем размещения гауссианы по каждой точке данных, суммирования их составляющих по всему набору данных, а затем деления на N для корректной нормализации плотности. На рис. 3.14 модель (3.184) применена к набору данных, использованному ранее для демонстрации методики гистограмм. Как и предполагалось, параметр h играет роль сглаживающего параметра, при этом наблюдается определенный баланс между чувствительностью к шуму при малых h и чрезмерным сглаживанием при больших h . Если установить слишком маленький параметр h (верхняя панель), то получится очень шумная модель плотности, а если установить слишком большой (нижняя панель), то бимодальный характер распределения, на основе которого генерируются данные (показан зеленой кривой), стирается. Наилучшая модель плотности получается при некотором промежуточном значении h (средняя панель). И вновь оптимизация h представляет собой проблему сложности модели, аналогичную выбору ширины ячейки при оценке плотности гистограммы или степени полинома, используемого при подгонке кривой.

РИС. 3.14 Иллюстрация модели плотности ядра (3.184), примененной ранее к набору данных для демонстрации методики гистограмм на рис. 3.13



В качестве ядра можно выбрать любую другую функцию $k(\mathbf{u})$ из (3.183) при соблюдении условий

$$k(\mathbf{u}) > 0, \quad (3.185)$$

$$\int k(\mathbf{u}) d\mathbf{u} = 1, \quad (3.186)$$

которые гарантируют, что результирующее распределение вероятностей везде неотрицательно и интегрируется в единицу. Класс моделей распределения плотности, заданный в (3.183), называют *ядерной оценкой плотности (kernel density estimator)* или *оценщиком Парзена (Parzen estimator)*. Ее достоинством является отсутствие вычислений на этапе «обучения», поскольку в этом случае требуется лишь сохранить обучающий набор. Однако это же является и недостатком, поскольку вычислительные затраты на оценку плотности растут линейно с размером набора данных.

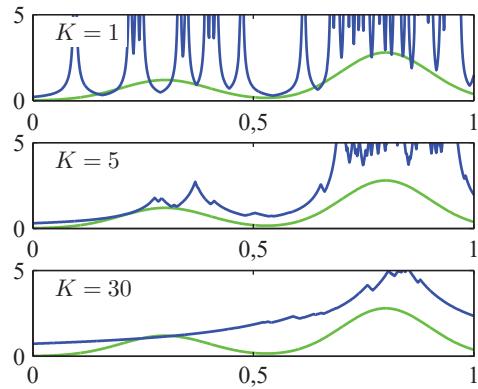
3.5.3. Методика ближайших соседей

Одна из трудностей при использовании ядерной методики для оценки распределения плотности состоит в том, что параметр h , определяющий ширину ядра, является постоянным для всех ядер. В областях с высокой плотностью данных большое значение h может привести к чрезмерному сглаживанию и вымыванию информации о структуре, которую в противном случае можно было бы извлечь из данных. Однако уменьшение h может привести к зашумлению оценок в других областях пространства данных, где плотность информации невелика. Таким образом, оптимальный выбор h может зависеть от местоположения в пространстве данных. Эта проблема решается с помощью методики ближайших соседей для оценки плотности распределения.

Поэтому вернемся к нашему основному результату (3.180) для локальной оценки распределения плотности, и вместо фиксации V и определения значения K по данным обратимся к фиксированному значению K и будем использовать данные для поиска подходящего значения V . Для этого рассмотрим небольшую сферу с центром в точке \mathbf{x} , где требуется оценить плотность $p(\mathbf{x})$, и позволим радиусу сферы расти до тех пор, пока она не будет содержать ровно K точек данных. Тогда оценка плотности $p(\mathbf{x})$ определяется по (3.180), а V обозначает объем полученной сферы. Такая методика известна как метод *K -ближайших соседей (K nearest neighbours)*. Она проиллюстрирована на рис. 3.15 для различных вариантов параметра K с использованием того же набора данных, что и на рис. 3.13 и 3.14. Параметр K регулирует степень сглаживания, так что малое значение K приводит к очень шумной модели плотности (верхняя часть), в то время как большое значение (нижняя часть) сглаживает бимодальный характер истинного распределения (показано зеленой кривой), из которого был сгенерирован набор данных. Как видно, и вновь для K найдется оптимальный вариант, который не будет ни слишком большим, ни слишком маленьким. Обратите внимание, что модель, полученная с помощью методики K -ближайших соседей, не является истинной

моделью распределения плотности, поскольку интеграл по всему пространству расходится (см. упражнение 3.38).

РИС. 3.15 Иллюстрация оценки плотности с помощью K -ближайших соседей на том же наборе данных, что и на рис. 3.14 и 3.13



В конце этой главы рассмотрим возможность распространения принципа оценки плотности с помощью методики K -ближайших соседей на задачи классификации. Для этого необходимо применить методику оценки плотности K -ближайших соседей к каждому классу отдельно, а затем прибегнуть к теореме Байеса. Пусть имеется набор данных из N_k точек категории \mathcal{C}_k общим числом N точек, так что $\sum_k N_k = N$. Если требуется классифицировать новую точку \mathbf{x} , нужно нарисовать сферу с центром в \mathbf{x} , содержащую ровно K точек независимо от их класса. Предположим, что эта сфера имеет объем V и содержит K_k точек класса \mathcal{C}_k . Тогда из (3.180) можно получить оценку плотности, связанной с каждым классом:

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V}. \quad (3.187)$$

Точно так же плотность безусловного распределения определяется как

$$p(\mathbf{x}) = \frac{K}{NV}, \quad (3.188)$$

а априорные вероятности классов задаются как

$$p(\mathcal{C}_k) = \frac{N_k}{N}. \quad (3.189)$$

Теперь можно объединить (3.187), (3.188) и (3.189) с использованием теоремы Байеса и получить апостериорную вероятность принадлежности к определенному классу:

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}. \quad (3.190)$$

Вероятность неправильной классификации можно минимизировать с помощью отнесения тестовой точки \mathbf{x} к классу с наибольшей апостериорной вероятностью, соответствующей наибольшему значению K_k/K . Таким образом, чтобы классифицировать новую точку, необходимо определить K ближайших точек из обучающего набора данных, а затем отнести новую точку к классу с наибольшим числом его представителей среди этого набора. Связи могут быть разорваны случайным образом. Частный случай $K = 1$ называется *правилом ближайших соседей (nearest-neighbour rule)*, поскольку тестовая точка просто приписывается к тому же классу, что и ближайшая точка из обучающего набора. Эти принципы проиллюстрированы на рис. 3.16. Как показано на рисунке, (а) в классификаторе K -ближайших соседей новая точка (черный ромб) классифицируется в соответствии с принадлежностью к большинству классов K ближайших точек обучающих данных, в данном случае $K = 3$; (б) при подходе классификации по методике ближайших соседей ($K = 1$) результирующая граница принятия решения состоит из гиперплоскостей, образующих перпендикулярные биссектрисы пар точек из разных классов.

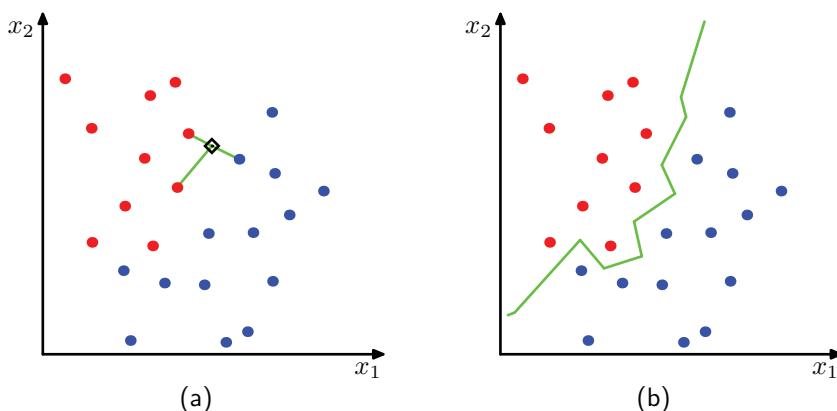


РИС. 3.16 Классификация новой точки

Интересным свойством методики классификации по ближайшим соседям ($K = 1$) является то, что в пределе $N \rightarrow \infty$ коэффициент ошибок никогда не превышает более чем в два раза минимально достижимый коэффициент ошибок при использовании методики оптимальной классификации, т. е. классификации, использующей истинные распределения классов (Cover and Hart, 1967).

Как уже говорилось ранее, как методика K -ближайших соседей, так и ядерная оценка плотности требуют хранения всего обучающего набора данных, что увеличивает стоимость вычислений в случае большого набора данных. Этот эффект может быть компенсирован за счет дополнительных единовременных вычислений путем построения древовидных поисковых структур, позволяющих эффективно находить (приблизительных) ближайших соседей без необходимости проведения исчерпывающего поиска по всему набору данных. Тем не менее эти непараметрические методы все равно существен-

но ограничены. С другой стороны, как было показано ранее, простые параметрические модели весьма ограничены в возможностях представления форм распределения. Поэтому возникает необходимость найти такие гибкие модели распределения плотности, у которых сложность модели можно контролировать вне зависимости от размера обучающего набора. Все это можно осуществить с помощью глубоких нейронных сетей.

Упражнения

- 3.1** (\star) Докажите, что распределение Бернулли (3.2) удовлетворяет следующим свойствам:

$$\sum_{x=0}^1 p(x|\mu) = 1, \quad (3.191)$$

$$\mathbb{E}[x] = \mu, \quad (3.192)$$

$$\text{var}[x] = \mu(1 - \mu). \quad (3.193)$$

Докажите, что энтропия $H[x]$ случайной бинарной переменной x с распределением Бернулли задается как

$$H[x] = -\mu \ln \mu - (1 - \mu) \ln(1 - \mu). \quad (3.194)$$

- 3.2** ($\star\star$) Форма распределения Бернулли, заданная (3.2), не является симметричной относительно двух значений x . В некоторых ситуациях будет удобнее использовать эквивалентную формулировку, для которой $x \in \{-1, 1\}$, и тогда распределение может быть записано в виде

$$p(x|\mu) = \left(\frac{1-\mu}{2}\right)^{(1-x)/2} \left(\frac{1+\mu}{2}\right)^{(1+x)/2}, \quad (3.195)$$

где $\mu \in [-1, 1]$. Докажите, что распределение (3.195) нормализовано, и оцените его среднее значение, дисперсию и энтропию.

- 3.3** ($\star\star$) В этом упражнении необходимо доказать, что биномиальное распределение (3.9) является нормализованным. Сначала воспользуемся определением (3.10) числа комбинаций из m одинаковых объектов, выбранных из общего числа N , чтобы доказать, что

$$\binom{N}{m} + \binom{N}{m-1} = \binom{N+1}{m}. \quad (3.196)$$

Используйте этот факт для доказательства по индукции следующего результата:

$$(1+x)^N = \sum_{m=0}^N \binom{N}{m} x^m, \quad (3.197)$$

который известен как *биномиальная теорема* (*binomial theorem*) и который справедлив для всех действительных значений x . Наконец, докажите, что биномиальное распределение нормализовано, поэтому

$$\sum_{m=0}^N \binom{N}{m} \mu^m (1-\mu)^{N-m} = 1, \quad (3.198)$$

что можно сделать, сначала вычтя из результата суммирования коэффициент $(1-\mu)^N$, а затем воспользовавшись биномиальной теоремой.

- 3.4** (**) Докажите, что среднее значение биномиального распределения определяется по (3.11). Для этого продифференцируйте обе стороны условия нормализации (3.198) по μ , а затем перегруппируйте, чтобы получить выражение для среднего значения n . Точно так же, дважды про-дифференцировав (3.198) по μ и воспользовавшись результатом (3.11) для среднего биномиального распределения, докажите результат (3.12) для дисперсии биномиального распределения.
- 3.5** (*) Докажите, что мода многомерного гауссова распределения (3.26) задается μ .
- 3.6** (**) Предположим, что \mathbf{x} характеризуется гауссовым распределением со средним значением μ и ковариацией Σ . Докажите, что линейно пре-образованная переменная $\mathbf{Ax} + \mathbf{b}$ также имеет гауссово распределение, и найдите ее среднее значение и ковариацию.
- 3.7** (***) Докажите, что расхождение Кульбака–Лейблера между двумя га-уссовыми распределениями $q(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ и $p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ имеет вид:

$$\begin{aligned} & \text{KL}(q(\mathbf{x}) \| p(\mathbf{x})) \\ &= \frac{1}{2} \left\{ \ln \frac{|\boldsymbol{\Sigma}_p|}{|\boldsymbol{\Sigma}_q|} - D + \text{Tr}(\boldsymbol{\Sigma}_p^{-1} \boldsymbol{\Sigma}_q) + (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T \boldsymbol{\Sigma}_p^{-1} (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q) \right\}, \end{aligned} \quad (3.199)$$

где $\text{Tr}(\cdot)$ обозначает след матрицы, а D означает размерность \mathbf{x} .

- 3.8** (**) Это упражнение позволяет доказать, что многомерное распределение с максимальной энтропией при заданной ковариации является гауссовым. Энтропия распределения $p(\mathbf{x})$ определяется как

$$H[\mathbf{x}] = - \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x}. \quad (3.200)$$

Требуется максимизировать $H[\mathbf{x}]$ по всем распределениям $p(\mathbf{x})$ при условии, что $p(\mathbf{x})$ нормализовано и имеет определенные с среднее значение и ковариацию, так что

$$\int p(\mathbf{x}) d\mathbf{x} = 1, \quad (3.201)$$

$$\int p(\mathbf{x}) \mathbf{x} d\mathbf{x} = \boldsymbol{\mu}, \quad (3.202)$$

$$\int p(\mathbf{x}) (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T d\mathbf{x} = \boldsymbol{\Sigma}. \quad (3.203)$$

Выполнив вариационную максимизацию (3.200) и используя множители Лагранжа для обеспечения ограничений (3.201), (3.202) и (3.203), докажите, что распределение максимального правдоподобия задается гауссовым распределением (3.26).

- 3.9** (****) Докажите, что энтропия многомерного гауссова распределения $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$ задается как

$$H[\mathbf{x}] = \frac{1}{2} \ln |\boldsymbol{\Sigma}| + \frac{D}{2} (1 + \ln(2\pi)), \quad (3.204)$$

где D является размерностью \mathbf{x} .

- 3.10** (*** Рассмотрим две случайные переменные x_1 и x_2 , имеющие гауссовые распределения со средними значениями μ_1 и μ_2 и точностями τ_1 и τ_2 соответственно. Выведите выражение для дифференциальной энтропии переменной $x = x_1 + x_2$. Для этого сначала найдите распределение x , используя соотношение

$$p(x) = \int_{-\infty}^{\infty} p(x|x_2)p(x_2)dx_2, \quad (3.205)$$

и затем возведите в квадрат экспоненту. Далее обратите внимание, что это представляет собой свертку двух гауссовых распределений, которые сами будут гауссовыми, и, наконец, воспользуйтесь результатом (2.99) для энтропии одномерного гауссова распределения.

- 3.11** (*) Рассмотрим многомерное гауссово распределение, заданное в (3.26). Записав матрицу точности (обратную ковариационную матрицу) как сумму симметричной и антисимметричной матриц, докажите, что антисимметричный член не появляется в экспоненте гауссова распределения, и, следовательно, матрицу точности можно считать симметричной без потери общности. Поскольку инверсия симметричной матрицы также симметрична (см. упражнение 3.16), из этого следует, что ковариационная матрица также может быть выбрана симметричной без потери общности.

- 3.12** (*** Рассмотрим вещественную симметричную матрицу $\boldsymbol{\Sigma}$, уравнение с собственными значениями которой приведено в (3.28). Взяв комплексное сопряженное этого уравнения, вычтя из него исходное уравнение и образовав внутреннее произведение с собственным вектором \mathbf{u}_i , докажите, что собственные значения λ_i вещественны. Точно так же, используя свойство симметрии $\boldsymbol{\Sigma}$, докажите, что два собственных вектора \mathbf{u}_i и \mathbf{u}_j будут ортогональны при условии, что $\lambda_i \neq \lambda_j$. Наконец, до-

кажите, что набор собственных векторов без потери общности можно выбрать ортонормированным так, чтобы удовлетворить (3.29), даже если некоторые собственные значения равны нулю.

- 3.13** (**) Докажите, что вещественная симметричная матрица Σ , удовлетвряющая собственному уравнению (3.28), может быть выражена в виде разложения по собственным векторам с коэффициентами, заданными собственными значениями, в форме (3.31). Точно так же докажите, что обратная матрица Σ^{-1} имеет представление в виде (3.32).
- 3.14** (**) Для положительно определенной матрицы Σ можно найти такую матрицу, для которой квадратичная форма

$$\mathbf{a}^T \Sigma \mathbf{a} \quad (3.206)$$

положительна при любом действительном значении вектора \mathbf{a} . Докажите, что необходимым и достаточным условием положительной определенности Σ является то, что все собственные значения λ_i матрицы Σ , определяемые по (3.28), положительны.

- 3.15** (*) Докажите, что вещественная симметричная матрица размера $D \times D$ имеет $D(D + 1)/2$ независимых параметров.
- 3.16** (*) Докажите, что инверсия симметричной матрицы сама является симметричной.
- 3.17** (**) Диагонализируя систему координат с помощью разложения по собственным векторам (3.31), докажите, что объем, содержащийся в гиперэллипсоиде, соответствующем постоянному расстоянию Махаланобиса Δ , определяется как

$$V_D |\Sigma|^{1/2} \Delta^D, \quad (3.207)$$

где V_D – это объем единичной сферы в D измерениях, а расстояние Махаланобиса определяется из (3.27).

- 3.18** (**) Докажите тождество (3.60), умножив обе стороны на матрицу

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \quad (3.208)$$

и используя определение (3.61).

- 3.19** (***) В разделах 3.2.4 и 3.2.5 были рассмотрены условные и маргинальные распределения для многомерного гауссова распределения. В более общем случае можно рассмотреть разбиение компонент \mathbf{x} на три группы \mathbf{x}_a , \mathbf{x}_b и \mathbf{x}_c с соответствующим разбиением среднего вектора μ и ковариационной матрицы Σ в виде

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \\ \boldsymbol{\mu}_c \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} & \boldsymbol{\Sigma}_{ac} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} & \boldsymbol{\Sigma}_{bc} \\ \boldsymbol{\Sigma}_{ca} & \boldsymbol{\Sigma}_{cb} & \boldsymbol{\Sigma}_{cc} \end{pmatrix}. \quad (3.209)$$

Используя результаты раздела 3.2, найдите выражение для условного распределения $p(\mathbf{x}_a | \mathbf{x}_b)$, в котором \mathbf{x}_c отброшено.

- 3.20** (**) Весьма полезный инструмент линейной алгебры – формула инверсии матрицы Вудбери (*Woodbury matrix*), которая выглядит как

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}. \quad (3.210)$$

Умножив обе стороны на $(\mathbf{A} + \mathbf{BCD})$, докажите правильность этого результата.

- 3.21** (*) Пусть \mathbf{x} и \mathbf{z} – два независимых произвольных вектора, так что $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{z})$. Докажите, что среднее значение их суммы $\mathbf{y} = \mathbf{x} + \mathbf{z}$ определяется суммой средних значений каждой из переменных в отдельности. Точно так же докажите, что ковариационная матрица \mathbf{y} определяется суммой ковариационных матриц \mathbf{x} и \mathbf{z} .

- 3.22** (****) Определите совместное распределение по переменной

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}, \quad (3.211)$$

среднее значение и ковариация которого заданы в (3.92) и (3.89) соответственно. Используя результаты (3.76) и (3.77), докажите, что маргинальное распределение $p(\mathbf{x})$ имеет вид (3.83). Таким же образом, используя результаты (3.65) и (3.66), докажите, что условное распределение $p(\mathbf{y} | \mathbf{x})$ имеет вид (3.84).

- 3.23** (**) Используя формулу инверсии раздделенной матрицы (3.60), докажите, что инверсия матрицы точности (3.88) задается ковариационной матрицей (3.89).

- 3.24** (**) Исходя из (3.91) и используя результат (3.89), докажите правильность (3.92).

- 3.25** (**) Рассмотрим два многомерных произвольных вектора \mathbf{x} и \mathbf{z} с гауссовыми распределениями $p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_x, \Sigma_x)$ и $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}_z, \Sigma_z)$ соответственно вместе с их суммой $\mathbf{y} = \mathbf{x} + \mathbf{z}$. Считая линейную гауссову модель произведением маргинального распределения $p(\mathbf{x})$ и условного распределения $p(\mathbf{y} | \mathbf{x})$, а также используя результаты (3.93) и (3.94), докажите, что маргинальное распределение $p(\mathbf{y})$ определяется как

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \boldsymbol{\mu}_x + \boldsymbol{\mu}_z, \Sigma_x + \Sigma_z). \quad (3.212)$$

- 3.26** (**) Это и следующее упражнения дают возможность попрактиковаться в работе с квадратичными формами, которые появляются в линейных гауссовых моделях, а также помогают осуществить независимую проверку результатов, полученных в основном тексте. Рассмотрим совместное распределение $p(\mathbf{x}, \mathbf{y})$, определяемое маргинальным и условным распределениями, заданными в (3.83) и (3.84). Исследуя квадра-

тичную форму в экспоненте совместного распределения и используя технику «дополнения до полного квадрата», рассмотренную в разделе 3.2, найдите выражения для среднего значения и ковариации маргинального распределения $p(y)$, в котором переменная x была проинтегрирована. Для этого воспользуйтесь формулой инверсии матрицы Вудбери (3.210). Убедитесь, что эти результаты согласуются с (3.93) и (3.94).

- 3.27** (★★) Рассмотрим то же совместное распределение, что и в упражнении 3.26, но теперь воспользуемся техникой дополнения до полного квадрата для нахождения выражения для среднего значения и ковариации условного распределения $p(x|y)$. И снова убедитесь, что они совпадают с соответствующими выражениями (3.95) и (3.96).
- 3.28** (★★) Чтобы найти решение максимального правдоподобия для ковариационной матрицы многомерного гауссова распределения, необходимо максимизировать функцию логарифмического правдоподобия (3.102) относительно Σ , при этом следует учитывать, что ковариационная матрица должна быть симметричной и положительно определенной. Здесь можно пренебречь этими ограничениями и выполнить прямую максимизацию. Используя результаты (A.21), (A.26) и (A.28) из приложения А, докажите, что ковариационная матрица Σ , максимизирующая функцию логарифмического правдоподобия (3.102), задается ковариацией выборки (3.106). Обратите внимание, что конечный результат обязательно симметричен и положительно определен (при условии, что выборочная ковариация несингулярна).
- 3.29** (★★) Используйте результат (3.42) для доказательства (3.46). Далее, используя результаты (3.42) и (3.46), докажите, что

$$\mathbb{E}[x_n x_m^T] = \mu \mu^T + I_{nm} \Sigma, \quad (3.213)$$

где x_n – это точка данных, отобранная из гауссова распределения со средним значением μ и ковариацией Σ , а I_{nm} – это элемент (n, m) матрицы тождеств. Таким образом, можно доказать результат (3.108).

- 3.30** (*) Различные тригонометрические тождества, использованные при обсуждении периодических переменных в этой главе, удобно доказывать из соотношения

$$\exp(iA) = \cos A + i \sin A, \quad (3.214)$$

где i – это квадратный корень из минус единицы. С учетом тождества

$$\exp(iA)\exp(-iA) = 1 \quad (3.215)$$

докажите результат (3.127). Аналогично, используя тождество

$$\cos(A - B) = \Re \exp\{i(A - B)\}, \quad (3.216)$$

где \Re обозначает вещественную часть, докажите (3.128). Наконец, используя $\sin(A - B) = \Im \exp\{i(A - B)\}$, где \Im обозначает мнимую часть, докажите результат (3.133).

- 3.31** (**) При больших значениях m распределение фон Мизеса (3.129) имеет резкий пик вблизи моды θ_0 . Определив $\xi = m^{1/2}(\theta - \theta_0)$ и взяв разложение Тейлора для функции косинуса, заданное как

$$\cos \alpha = 1 - \frac{\alpha^2}{2} + O(\alpha^4), \quad (3.217)$$

докажите, что при $m \rightarrow \infty$ распределение фон Мизеса стремится к гауссову распределению.

- 3.32** (*) Используя тригонометрическое тождество (3.133), докажите, что решение (3.132) для θ_0 имеет вид (3.134).
- 3.33** (*) Вычислив первую и вторую производные распределения фон Мизеса (3.129) и используя условие $I_0(m) > 0$ для $m > 0$, докажите, что максимум распределения наступает при $\theta = \theta_0$, а минимум – при $\theta = \theta_0 + \pi \pmod{2\pi}$.
- 3.34** (*) Используя результат (3.118) вместе с (3.134) и тригонометрическим тождеством (3.128), докажите, что решение максимального правдоподобия m_{ML} для концентрации распределения фон Мизеса удовлетворяет $A(m_{ML}) = \bar{r}$, где \bar{r} – это радиус среднего значения из наблюдений, рассматриваемых как единичные векторы в двумерной евклидовой плоскости, как показано на рис. 3.9.
- 3.35** (*) Подтвердите, что многомерное гауссово распределение может быть приведено к экспоненциальному семейству (3.138), и выведите выражения для η , $\mathbf{u}(\mathbf{x})$, $h(\mathbf{x})$ и $g(\eta)$, аналогичные (3.164)–(3.167).
- 3.36** (*) Результат (3.172) свидетельствует о том, что отрицательный градиент $\nabla \ln g(\eta)$ для экспоненциального семейства определяется ожиданием $\mathbf{u}(\mathbf{x})$. Взяв вторые производные от (3.139), докажите, что
- $$-\nabla \nabla \ln g(\eta) = \mathbb{E}[\mathbf{u}(\mathbf{x})\mathbf{u}(\mathbf{x})^T] - \mathbb{E}[\mathbf{u}(\mathbf{x})]\mathbb{E}[\mathbf{u}(\mathbf{x})^T] = \text{cov}[\mathbf{u}(\mathbf{x})]. \quad (3.218)$$
- 3.37** (**) Рассмотрим гистограммоподобную модель распределения плотности, в которой пространство \mathbf{x} разделено на фиксированные области, для которых плотность $p(\mathbf{x})$ принимает постоянное значение h_i в i -й области. Объем области i обозначается Δ_i . Предположим, что имеется набор из N наблюдений \mathbf{x} , из которых n_i попадают в область i . Используя множитель Лагранжа для обеспечения ограничения на нормализацию плотности, выведите выражение для оценки максимального правдоподобия для $\{h_i\}$.
- 3.38** (*) Докажите, что модель плотности K -ближайших соседей определяет неправильное распределение, интеграл которого по всему пространству является расходящимся.

Глава 4

Однослойные сети: регрессия

В этой главе рассматриваются базовые концепции нейронных сетей на примере линейной регрессии (см. раздел 1.2), которая уже упоминалась ранее в контексте подгонки полиномиальных кривых. В дальнейшем будут приведены доказательства того, что модель линейной регрессии соответствует простой форме нейронной сети с одним слоем обучаемых параметров. Хотя однослойные сети имеют весьма ограниченное практическое применение, они обладают простыми аналитическими свойствами и служат удобным инструментом для введения многих основных принципов, закладывающих основу для обсуждения глубоких нейронных сетей в последующих главах.

4.1. Линейная регрессия

Задачей регрессии является прогнозирование значения одной или нескольких непрерывных целевых (*target*) переменных t по значению D -мерного вектора \mathbf{x} входных (*input*) переменных. Как правило, задается обучающий набор данных из N наблюдений $\{\mathbf{x}_n\}$, где $n = 1, \dots, N$, вместе с соответствующими целевыми значениями $\{t_n\}$, и далее ставится задача предсказать значение t для нового значения x . Для этого необходимо сформулировать функцию $y(\mathbf{x}, \mathbf{w})$, значения которой для новых входных данных \mathbf{x} являются прогнозами для соответствующих значений t и где \mathbf{w} представляет собой вектор параметров, которые могут быть получены из обучающих данных.

Наиболее простой формой регрессии является модель, включающая линейную комбинацию входных переменных:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_D x_D, \quad (4.1)$$

где $\mathbf{x} = (x_1, \dots, x_D)^T$. Термин «линейная регрессия» (*linear regression*) обычно относят именно к этой форме модели. Ключевым ее свойством является линейная функция параметров w_0, \dots, w_D . Однако она также является линейной функцией входных переменных x_i , и это накладывает на такую модель существенные ограничения.

4.1.1. Базисные функции

Класс моделей, определяемых в (4.1), можно расширить линейными комбинациями фиксированных нелинейных функций входных переменных в виде

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \varphi_j(\mathbf{x}), \quad (4.2)$$

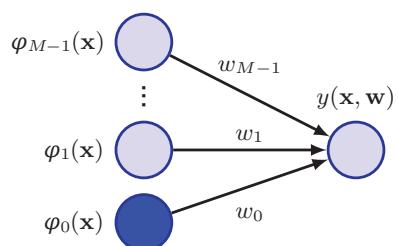
где $\varphi_j(\mathbf{x})$ называют *базисными функциями* (*basis functions*). Если обозначить максимальное значение индекса j через $M - 1$, то общее число параметров в этой модели будет равно M .

Параметр w_0 дает возможность учесть любое фиксированное смещение в данных и иногда называется *параметром смещения* (*bias parameter*, не путать со смещением в терминологии статистического анализа) (см. раздел 4.3). Зачастую оказывается удобным определить дополнительную *фиктивную* (*dummy*) базисную функцию $\varphi_0(\mathbf{x})$, значение которой фиксировано и равно $\varphi_0(\mathbf{x}) = 1$, так что (4.2) приобретает вид

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{M-1} w_j \varphi_j(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}), \quad (4.3)$$

где $\mathbf{w} = (w_0, \dots, w_{M-1})^T$ и $\varphi = (\varphi_0, \dots, \varphi_{M-1})^T$. Модель (4.3) можно представить с помощью диаграммы нейронной сети, как показано на рис. 4.1. Здесь каждая базисная функция $\varphi_j(\mathbf{x})$ представлена входным узлом, сплошной узел представляет «смещенную» базисную функцию φ_0 , а функция $y(\mathbf{x}, \mathbf{w})$ представлена выходным узлом. Каждый из параметров w_j показан линией, соединяющей соответствующую базисную функцию с выходом.

РИС. 4.1 Модель линейной регрессии (4.3) в виде простой диаграммы нейронной сети с одним слоем параметров



Благодаря использованию нелинейных базисных функций обеспечивается возможность получения нелинейной функции $y(\mathbf{x}, \mathbf{w})$ относительно входного вектора \mathbf{x} . Функции вида (4.2) называются линейными моделями (*linear models*) в силу того, что они линейны относительно \mathbf{w} . Именно эта линейность параметров значительно упрощает анализ моделей этого класса. Однако она же является причиной ряда существенных ограничений (см. раздел 6.1).

До появления глубокого обучения обычной практикой в машинном обучении было использование какой-либо фиксированной предварительной обработки входных переменных \mathbf{x} , называемой также *извлечением признаков*

(*feature extraction*), которая выражалась в виде набора базисных функций $\{\varphi_j(x)\}$. Основной задачей был выбор достаточно мощного набора базисных функций, чтобы полученная задача обучения могла быть решена с помощью простой сетевой модели. К сожалению, подобрать подходящие базисные функции вручную крайне сложно для любых задач, кроме самых простых. Глубокое обучение позволяет избежать этой проблемы, поскольку необходимые нелинейные преобразования данных можно получить из самого набора данных.

Пример проблемы регрессии был уже рассмотрен при обсуждении подгонки кривых с помощью полиномов (см. главу 1). Полиномиальная функция (1.1) может быть выражена в виде (4.3), если рассматривается одна входная переменная x и если выбраны базисные функции, определяемые $\varphi_j(x) = x^j$. Существует множество других возможных вариантов базисных функций, например таких как

$$\varphi_j(x) = \exp\left\{-\frac{(x - \mu_j)^2}{2s^2}\right\}, \quad (4.4)$$

где μ_j определяет расположение базисных функций в пространстве входных данных, а параметр s определяет их пространственный масштаб. Обычно их называют *гауссовыми* базисными функциями, хотя следует отметить, что они не обязательно должны иметь вероятностную интерпретацию. В частности, коэффициент нормализации не играет роли, поскольку эти базисные функции будут умножаться на обучаемые параметры w_j .

Еще одним вариантом является сигмоидная базисная функция вида

$$\varphi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right), \quad (4.5)$$

где $\sigma(a)$ – это логистическая сигмоидная функция, определяемая как

$$\sigma(a) = \frac{1}{1 + \exp(-a)}. \quad (4.6)$$

Точно так же можно использовать функцию \tanh , поскольку она связана с логистической сигмоидной функцией $\tanh(a) = 2\sigma(2a) - 1$, и потому общая линейная комбинация логистических сигмоидных функций (см. упражнение 4.3) эквивалентна общей линейной комбинации функций \tanh в том смысле, что они могут представлять один и тот же класс функций ввода-вывода. Эти различные варианты базисных функций представлены на рис. 4.2.

Еще один возможный выбор базисной функции – это базис Фурье, который приводит к разложению по синусоидальным функциям. Каждая базисная функция представляет определенную частоту и имеет бесконечную пространственную протяженность. В отличие от этого базисные функции, локализованные в конечных областях входного пространства, в обязательном порядке представляют собой спектр различных пространственных частот.

В приложениях по обработке сигналов зачастую интересно рассматривать базисные функции, локализованные как в пространстве, так и по частоте, что приводит к классу функций, известных как вейвлеты (wavelets) (Ogden, 1997; Mallat, 1999; Vidakovic, 1999). Для упрощения их применения они также определяются как взаимно ортогональные. Наиболее эффективно вейвлеты применяются в тех случаях, когда входные значения располагаются на регулярной решетке, например последовательные временные точки во временной последовательности или пиксели на изображении.

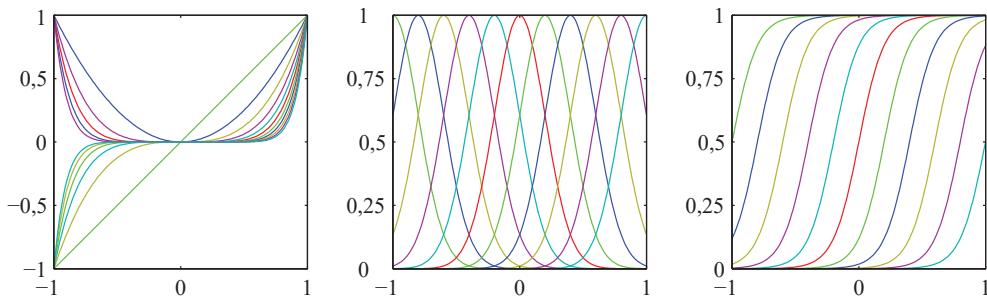


РИС. 4.2 Примеры базисных функций: слева – полиномы, в центре – гауссова распределение в форме (4.4), справа – сигмоидные базисные функции в форме (4.5)

Однако большинство обсуждаемых в этой главе вопросов не зависят от выбора набора базисных функций, поэтому здесь можно не указывать конкретную форму базисных функций, за исключением числовых иллюстраций. Кроме того, чтобы сохранить простоту обозначений, рассмотрим случай одной целевой переменной t , но при этом кратко опишем модификации (см. раздел 4.1.7), необходимые для работы с несколькими целевыми переменными.

4.1.2. Функция правдоподобия

Задача подгонки полиномиальной функции к данным решается путем минимизации функции ошибки суммы квадратов (см. раздел 1.2), и также нами было показано, что эта функция ошибки может быть сформулирована в виде максимума правдоподобия для предполагаемой модели гауссова шума. Теперь вернемся к этому вопросу и более подробно рассмотрим методику наименьших квадратов, а также ее связь с максимальным правдоподобием.

Как и прежде, предполагается, что целевая переменная t задается детерминированной функцией $y(\mathbf{x}, \mathbf{w})$ с аддитивным гауссовым шумом, так что

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon, \quad (4.7)$$

где ϵ – это гауссова случайная величина с нулевым средним значением и дисперсией σ^2 . Таким образом, можно записать:

$$p(t | \mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(t | y(\mathbf{x}, \mathbf{w}), \sigma^2). \quad (4.8)$$

Теперь рассмотрим набор входных данных $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ с соответствующими целевыми значениями t_1, \dots, t_N . Сгруппируем целевые переменные $\{t_n\}$ в вектор-столбец, который обозначим \mathbf{t} , где выбран другой шрифт с целью отличить его от единичного наблюдения многомерной цели, которое обозначим \mathbf{t} . Предполагая, что эти точки данных взяты независимо из распределения (4.8), получим выражение для функции правдоподобия, которая является функцией настраиваемых параметров \mathbf{w} и σ^2 :

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(t_n | \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}_n), \sigma^2), \quad (4.9)$$

где применено (4.3). Взяв логарифм функции правдоподобия и воспользовавшись стандартной формой (2.49) для одномерного гауссова распределения, получим

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) &= \sum_{n=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}_n), \sigma^2) \\ &= -\frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi) - \frac{1}{\sigma^2} E_D(\mathbf{w}), \end{aligned} \quad (4.10)$$

где функция ошибки суммы квадратов определяется как

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}_n)\}^2. \quad (4.11)$$

Первые два члена в (4.10) при определении \mathbf{w} можно считать константами, поскольку они не зависят от \mathbf{w} (см. раздел 2.3.4). Следовательно, как уже говорилось ранее, максимизация функции правдоподобия при гауссовом распределении шума эквивалентна минимизации функции ошибки по сумме квадратов (4.11).

4.1.3. Максимальное правдоподобие

После записи функции правдоподобия можно воспользоваться методом максимального правдоподобия для определения \mathbf{w} и σ^2 . Сначала рассмотрим максимизацию относительно \mathbf{w} . Градиент функции логарифмического правдоподобия (4.10) относительно \mathbf{w} имеет вид:

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \frac{1}{\sigma^2} \sum_{n=1}^N \{t_n - \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}_n)\} \boldsymbol{\varphi}(\mathbf{x}_n)^\top. \quad (4.12)$$

Обращение этого градиента в ноль дает

$$0 = \sum_{n=1}^N t_n \boldsymbol{\varphi}(\mathbf{x}_n)^\top - \mathbf{w}^\top \left(\sum_{n=1}^N \boldsymbol{\varphi}(\mathbf{x}_n) \boldsymbol{\varphi}(\mathbf{x}_n)^\top \right). \quad (4.13)$$

Решение для \mathbf{w} дает

$$\mathbf{w}_{\text{ML}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}, \quad (4.14)$$

которые известны как *нормальные уравнения* (*normal equations*) для задачи наименьших квадратов. Здесь Φ – это матрица $N \times M$, называемая *матрицей плана* (*design matrix*), и ее элементы задаются через $\Phi_{nj} = \varphi_j(\mathbf{x}_n)$, так что

$$\Phi = \begin{pmatrix} \varphi_0(\mathbf{x}_1) & \varphi_1(\mathbf{x}_1) & \cdots & \varphi_{M-1}(\mathbf{x}_1) \\ \varphi_0(\mathbf{x}_2) & \varphi_1(\mathbf{x}_2) & \cdots & \varphi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_0(\mathbf{x}_N) & \varphi_1(\mathbf{x}_N) & \cdots & \varphi_{M-1}(\mathbf{x}_N) \end{pmatrix}. \quad (4.15)$$

Величина

$$\Phi^\dagger \equiv (\Phi^T \Phi)^{-1} \Phi^T \quad (4.16)$$

известна как *псевдоинверсия* (*псевдообратная матрица*) *Мура–Пенроуза* (*Moore–Penrose pseudo-invers*) для матрицы Φ (Rao and Mitra, 1971; Golub and Van Loan, 1996). Ее можно рассматривать как обобщение концепции обратной матрицы для неквадратных матриц. И действительно, если Φ – это квадратная и инвертируемая матрица, то, используя свойство $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$, получаем $\Phi^\dagger \equiv \Phi^{-1}$.

На этом этапе становится понятна роль параметра смещения w_0 . Если сделать параметр смещения явным, то функция ошибки (4.11) приобретает вид:

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \left\{ t_n - w_0 - \sum_{j=1}^{M-1} w_j \varphi_j(\mathbf{x}_n) \right\}^2. \quad (4.17)$$

Установливая производную по w_0 равной нулю и находя решение для w_0 , получаем

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \bar{\varphi}_j, \quad (4.18)$$

где определяется

$$\bar{t} = \frac{1}{N} \sum_{n=1}^N t_n, \quad \bar{\varphi}_j = \frac{1}{N} \sum_{n=1}^N \varphi_j(\mathbf{x}_n). \quad (4.19)$$

Таким образом, смещение w_0 компенсирует разницу между средними значениями (по обучающему множеству), значениями цели и взвешенной суммой средних значений базисных функций.

Также можно максимизировать функцию логарифмического правдоподобия (4.10) с учетом дисперсии σ^2 , что дает

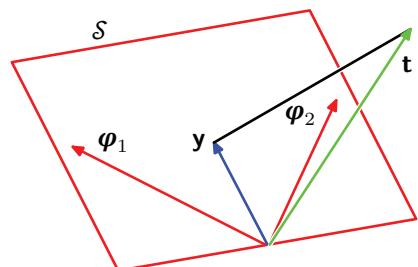
$$\sigma_{\text{ML}}^2 = \frac{1}{N} \sum_{n=1}^N \{t_n - \mathbf{w}_{\text{ML}}^T \varphi(\mathbf{x}_n)\}^2, \quad (4.20)$$

и здесь можно видеть, что максимальное значение правдоподобия параметра дисперсии задается остаточной дисперсией целевых значений вблизи функции регрессии.

4.1.4. Геометрия наименьших квадратов

На этом этапе целесообразно рассмотреть геометрическую интерпретацию решения по методу наименьших квадратов. Для этого рассмотрим N -мерное пространство, оси которого заданы через t_n , так что $\mathbf{t} = (t_1, \dots, t_N)^T$ – это вектор в этом пространстве. Каждая базисная функция $\varphi_j(\mathbf{x}_n)$, оцененная в N точках данных, также может быть представлена как вектор в том же пространстве, который обозначается $\boldsymbol{\varphi}_j$, как показано на рис. 4.3, что $\boldsymbol{\varphi}_j$ соответствует j -му столбцу Φ , а $\varphi_j(\mathbf{x}_n)$ соответствует транспонированию n -й строки Φ . Функция регрессии наименьших квадратов получается путем нахождения ортогональной проекции вектора данных \mathbf{t} на подпространство, охватываемое базисными функциями $\varphi_j(\mathbf{x})$, в котором каждая базисная функция рассматривается как вектор $\boldsymbol{\varphi}_j$ длины N с элементами $\varphi_j(\mathbf{x}_n)$. Если число M базисных функций меньше числа N точек данных, то M векторов $\varphi_j(\mathbf{x}_n)$ будут охватывать линейное подпространство \mathcal{S} с размерностью M . Определим \mathbf{y} как N -мерный вектор, n -й элемент которого задается $y(\mathbf{x}_n, \mathbf{w})$, где $n = 1, \dots, N$. Поскольку \mathbf{y} является произвольной линейной комбинацией векторов $\boldsymbol{\varphi}_j$, он может находиться в любой точке M -мерного подпространства. В этом случае ошибка суммы квадратов (4.11) равна (с точностью до $1/2$) квадрату евклидова расстояния между \mathbf{y} и \mathbf{t} . Таким образом, решение с наименьшими квадратами для \mathbf{w} соответствует такому выбору \mathbf{y} , который лежит в подпространстве \mathcal{S} и наиболее близок к \mathbf{t} . Исходя из рис. 4.3, можно предположить, что это решение соответствует ортогональной проекции \mathbf{t} на подпространство \mathcal{S} . Это действительно так, в чем можно легко убедиться на примере решения для \mathbf{y} , заданного $\Phi \mathbf{w}_{ML}$, а затем удостовериться, что оно имеет форму ортогональной проекции (см. упражнение 4.4).

РИС. 4.3 Геометрическая интерпретация решения по методу наименьших квадратов в N -мерном пространстве, осями которого являются значения t_1, \dots, t_N



На практике прямое решение нормальных уравнений может привести к затруднениям численного характера, когда $\Phi^T \Phi$ близка к сингулярной. В частности, когда два или более базисных вектора $\boldsymbol{\varphi}_j$ оказываются совместно-линейными или почти таковыми, результирующие значения параметров

могут иметь большие величины. Такие близкие вырождения нередки при работе с реальными наборами данных. Возникающие при этом числовые трудности можно преодолеть с помощью *техники разложения по сингулярным значениям (singular value decomposition, или SVD)* (Deisenroth, Faisal and Ong, 2020). Обратите внимание, что добавление члена регуляризации приводит к тому, что матрица остается несингулярной даже при наличии вырождений.

4.1.5. Последовательное обучение

Решение на основе максимального правдоподобия (4.14) предполагает обработку всего обучающего набора за один раз, что называется *пакетным методом (batch method)*. Для больших наборов данных такой метод может оказаться слишком затратным в плане вычислений. Если набор данных достаточно велик, возможно, имеет смысл использовать *последовательные алгоритмы (sequential algorithms)*, также известные как *онлайн-алгоритмы (online algorithms)*, когда точки данных рассматриваются по одной, а параметры модели обновляются после каждого такого обращения. Последовательное обучение также подходит для приложений реального времени, когда данные наблюдений поступают непрерывным потоком, а прогнозы должны быть сделаны до поступления всех точек данных.

Алгоритм последовательного обучения можно создать с использованием техники *стохастического градиентного спуска (stochastic gradient descent)* (см. главу 7), также называемого техникой *последовательного градиентного спуска (sequential gradient descent)*. Если функция ошибки представляет собой сумму по точкам данных $E = \sum_n E_n$, то после предъявления точки данных n алгоритм стохастического градиентного спуска обновляет вектор параметров \mathbf{w} с использованием

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n, \quad (4.21)$$

где τ означает номер итерации, а η – это подходящий параметр скорости обучения. Значение \mathbf{w} инициализируется некоторым начальным вектором $\mathbf{w}^{(0)}$. Для функции ошибки в виде суммы квадратов (4.11) это дает

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)\top} \varphi_n) \varphi_n, \quad (4.22)$$

где $\varphi_n = \varphi(\mathbf{x}_n)$. Этот алгоритм известен как алгоритм наименьших среднеквадратичных значений (least-mean-squares), или алгоритм LMS.

4.1.6. Регуляризованный метод наименьших квадратов

Ранее (см. раздел 1.2) уже была представлена идея добавления регуляризирующего члена к функции ошибки для предотвращения чрезмерной подгонки, так что общая функция ошибки, которую необходимо минимизировать, принимает вид:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w}), \quad (4.23)$$

где λ – это коэффициент регуляризации, который управляет относительным значением зависимой от данных ошибки $E_D(\mathbf{w})$ и регуляризационного члена $E_W(\mathbf{w})$. Одна из простейших форм регуляризатора задается суммой квадратов элементов весового вектора:

$$E_W(\mathbf{w}) = \frac{1}{2} \sum_j w_j^2 = \frac{1}{2} \mathbf{w}^T \mathbf{w}. \quad (4.24)$$

Если также рассмотреть функцию ошибки в виде суммы квадратов, то она будет иметь вид:

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_n))^2, \quad (4.25)$$

и тогда функция суммарной ошибки приобретает вид:

$$\frac{1}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_n))^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}. \quad (4.26)$$

В статистике этот регуляризатор является примером метода уменьшения параметров (*parameter shrinkage*), поскольку он сокращает значения параметров до нуля. Его преимущество в том, что функция ошибки остается квадратичной функцией от \mathbf{w} , и поэтому его точный минимизатор может быть найден в замкнутой форме. В частности, задав градиент (4.26) относительно \mathbf{w} равным нулю (см. упражнение 4.6) и найдя \mathbf{w} , как и раньше, получаем:

$$\mathbf{w} = (\lambda \mathbf{I} + \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{t}. \quad (4.27)$$

Это и есть простое расширение решения по методу наименьших квадратов (4.14).

4.1.7. Множественные выходы

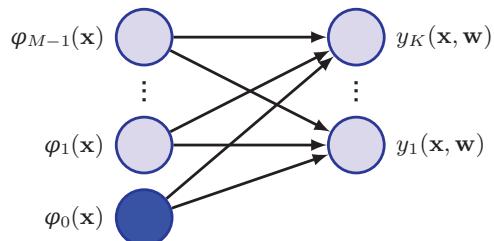
До сих пор речь шла о ситуациях с одной целевой переменной t . В некоторых приложениях может потребоваться прогнозирование $K > 1$ целевых переменных, которые будем обозначать совокупно целевым вектором $\mathbf{t} = (t_1, \dots, t_K)^T$. Для этого можно ввести отдельный набор базисных функций для каждой компоненты \mathbf{t} , что позволит решать задачи множественной, независимой регрессии. Однако более распространенным подходом является использование одного и того же набора базисных функций для моделирования всех компонент целевого вектора в виде

$$\mathbf{y}(\mathbf{x}, \mathbf{w}) = \mathbf{W}^T \boldsymbol{\varphi}(\mathbf{x}), \quad (4.28)$$

где \mathbf{y} – это K -мерный вектор-столбец, \mathbf{W} – это матрица параметров $M \times K$, а $\boldsymbol{\varphi}(\mathbf{x})$ – это M -мерный вектор-столбец с элементами $\varphi_i(\mathbf{x})$, при этом, как и прежде, $\varphi_0(\mathbf{x}) = 1$. И вновь это можно представить в виде нейронной сети

с одним слоем параметров, как показано на рис. 4.4. Каждая базисная функция представлена узлом, причем сплошной узел представляет «смещенную» базисную функцию φ_0 . Аналогично каждый выход y_1, \dots, y_K представлен узлом. Связи между узлами представляют собой соответствующие параметры веса и смещения.

РИС. 4.4 Представление модели линейной регрессии в виде нейронной сети с одним слоем связей



Предположим, что условное распределение целевого вектора является изотропным гауссовым в форме

$$p(\mathbf{t} | \mathbf{x}, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{t} | \mathbf{W}^T \boldsymbol{\varphi}(\mathbf{x}), \sigma^2 \mathbf{I}). \quad (4.29)$$

При наличии набора наблюдений $\mathbf{t}_1, \dots, \mathbf{t}_N$ можно объединить их в матрицу \mathbf{T} размера $N \times K$ так, чтобы n -я строка была представлена \mathbf{t}_n^T . Точно так же можно объединить входные векторы $\mathbf{x}_1, \dots, \mathbf{x}_N$ в матрицу \mathbf{X} . Тогда функция логарифмического правдоподобия имеет вид:

$$\begin{aligned} \ln p(\mathbf{T} | \mathbf{X}, \mathbf{W}, \sigma^2) &= \sum_{n=1}^N \ln \mathcal{N}(\mathbf{t}_n | \mathbf{W}^T \boldsymbol{\varphi}(\mathbf{x}_n), \sigma^2 \mathbf{I}) \\ &= -\frac{NK}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=1}^N \|\mathbf{t}_n - \mathbf{W}^T \boldsymbol{\varphi}(\mathbf{x}_n)\|^2. \end{aligned} \quad (4.30)$$

Как и ранее, эту функцию можно максимизировать относительно \mathbf{W} , что дает

$$\mathbf{W}_{\text{ML}} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{T}, \quad (4.31)$$

где исходные векторы признаков $\boldsymbol{\varphi}(\mathbf{x}_1), \dots, \boldsymbol{\varphi}(\mathbf{x}_N)$ объединены в матрицу $\boldsymbol{\Phi}$. Если рассматривать этот результат для каждой целевой переменной t_k , то получится

$$\mathbf{w}_k = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{t}_k = \boldsymbol{\Phi}^\dagger \mathbf{t}_k, \quad (4.32)$$

где \mathbf{t}_k – это N -мерный вектор-столбец с компонентами t_{nk} для $n = 1, \dots, N$. Таким образом, решение задачи регрессии разделяет различные целевые переменные, и теперь необходимо вычислить лишь одну псевдообратную матрицу $\boldsymbol{\Phi}^\dagger$, которая является общей для всех векторов \mathbf{w}_k .

Расширение на общие распределения гауссова шума с произвольными ковариационными матрицами не представляет трудностей (см. упражне-

ние 4.7). Это снова приводит к разложению на K независимых задач регрессии. Этот результат не вызывает удивления, поскольку параметры \mathbf{W} определяют только среднее значение распределения гауссова шума, и хорошо известно, что решение максимального правдоподобия для среднего значения многомерного гауссова распределения не зависит от ковариации (см. раздел 3.2.7). Поэтому в дальнейшем для простоты будет рассматриваться одна целевая переменная t .

4.2. Теория принятия решений

Сформулировав задачу регрессии как задачу моделирования распределения условной вероятности $p(t|\mathbf{x})$, можно выбрать конкретную форму условной вероятности, а именно гауссову (4.8) с зависящим от \mathbf{x} средним $y(\mathbf{x}, \mathbf{w})$, управляемым параметрами \mathbf{w} и дисперсией, задаваемой параметром σ^2 . Параметры \mathbf{w} и σ^2 могут быть получены из данных с помощью метода максимального правдоподобия. В результате получается прогнозируемое распределение (predictive distribution) вида

$$p(t|\mathbf{x}, \mathbf{w}_{\text{ML}}, \sigma_{\text{ML}}^2) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}_{\text{ML}}), \sigma_{\text{ML}}^2). \quad (4.33)$$

Прогнозируемое распределение представляет собой наше предположение о неопределенности значения t для некоторого нового входного сигнала \mathbf{x} . Однако во многих практических приложениях необходимо предсказать конкретное значение t , а не возвращать все распределение, особенно в тех случаях, когда требуется совершить какие-либо конкретные действия. Например, если целью является определение оптимального уровня облучения для лечения опухоли, а наша модель предсказывает распределение вероятностей для дозы облучения, то нужно использовать это распределение для принятия решения по конкретной дозе облучения. Таким образом, задача разбивается на два этапа. На первом этапе, называемом *этапом вывода (inference stage)*, прогнозируемое распределение $p(t|\mathbf{x})$ определяется с помощью обучающих данных. На втором этапе, называемом *этапом принятия решения (decision stage)*, при помощи этого прогнозируемого распределения определяются конкретные значения $f(\mathbf{x})$, зависящие от входного вектора \mathbf{x} , которые являются оптимальными в соответствии с какими-либо критериями. Для этого нужно минимизировать функцию потерь, которая зависит как от прогнозируемого распределения $p(t|\mathbf{x})$, так и от f .

Вполне естественно выбрать среднее значение условного распределения, так что в этом случае можно использовать $f(\mathbf{x}) = y(\mathbf{x}, \mathbf{w}_{\text{ML}})$. В некоторых случаях такой интуитивный подход будет правильным, но в других ситуациях он может дать очень плохие результаты. Поэтому полезно формализовать этот процесс, чтобы понять, когда и при каких условиях он действительно может применяться, а механизм, позволяющий это сделать, называется *теорией принятия решений (decision theory)*.

Предположим, что для прогноза выбрано значение $f(\mathbf{x})$, когда истинное значение равно t . При этом возникает некоторая форма штрафных санкций или издержек. Это определяется *потерями* (*loss*), которые обозначим как $L(t, f(\mathbf{x}))$. Разумеется, истинное значение t нам неизвестно, поэтому вместо минимизации самого L необходимо минимизировать средние значения, или ожидаемые потери, которые определяются как

$$\mathbb{E}[L] = \int \int L(t, f(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt, \quad (4.34)$$

где происходит усреднение по распределению входных и целевых переменных, взвешенному по их совместному распределению $p(\mathbf{x}, t)$. Обычно в задачах регрессии в качестве функции потерь выбирают квадратичную потерю, задаваемую как $L(t, f(\mathbf{x})) = \{f(\mathbf{x}) - t\}^2$. В этом случае ожидаемые потери можно записать как

$$\mathbb{E}[L] = \int \int \{f(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt. \quad (4.35)$$

Важно не путать квадратичную функцию потерь с представленной ранее функцией ошибки суммы квадратов. Функция ошибки используется для настройки параметров во время обучения, чтобы определить условное распределение вероятностей $p(t|\mathbf{x})$, в то время как функция потерь определяет, как используется условное распределение для получения предсказательной функции $f(\mathbf{x})$, которая определяет предсказание для каждого значения \mathbf{x} .

Нашей целью является выбор $f(\mathbf{x})$ таким образом, чтобы минимизировать $\mathbb{E}[L]$. Если предположить, что функция $f(\mathbf{x})$ является полностью гибкой (см. приложение B), можно сделать это формально с помощью вариационного исчисления и получить

$$\frac{\delta \mathbb{E}[L]}{\delta f(\mathbf{x})} = 2 \int \{f(\mathbf{x}) - t\} p(\mathbf{x}, t) dt = 0. \quad (4.36)$$

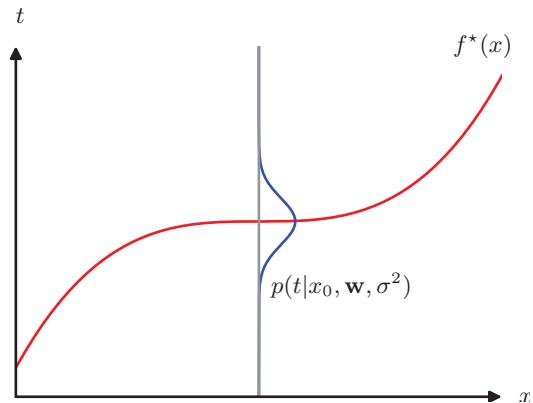
Вычисляя $f(\mathbf{x})$ и используя правила суммы и произведения вероятностей, получаем

$$f^*(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \int t p(\mathbf{x}, t) dt = \int t p(t|\mathbf{x}) dt = \mathbb{E}_t[t|\mathbf{x}], \quad (4.37)$$

что является условным средним значением t относительно \mathbf{x} и известно как функция *регрессии* (*regression function*). Этот результат проиллюстрирован на рис. 4.5. Его можно свободно распространить на несколько целевых переменных, представленных вектором \mathbf{t} , и в этом случае оптимальным решением будет условное среднее значение $f^*(\mathbf{x}) = \mathbb{E}_t[\mathbf{t}|\mathbf{x}]$ (см. упражнение 4.8). Для гауссова условного распределения вида (4.8) условное среднее значение будет выглядеть просто как

$$\mathbb{E}[t|\mathbf{x}] = \int t_p(t|\mathbf{x}) dt = y(\mathbf{x}, \mathbf{w}). \quad (4.38)$$

РИС. 4.5 Функция регрессии $f^*(x)$, минимизирующая ожидаемые квадратичные потери, задается средним значением условного распределения $p(t|x)$



Использование вариационного исчисления для получения (4.37) подразумевает необходимость проведения оптимизации по всем возможным функциям $f(\mathbf{x})$. Хотя любая параметрическая модель, которую можно реализовать на практике, ограничена в диапазоне функций, которые она может представлять, структура глубоких нейронных сетей, подробно рассмотренная в последующих главах, предоставляет чрезвычайно гибкий класс функций, которые для многих практических целей могут аппроксимировать любую желаемую функцию с высокой точностью.

Этот результат может быть получен несколько иным способом, который также пролетает свет на природу проблемы регрессии. Основываясь на информации о том, что оптимальным решением является условное ожидание, можно расширить квадратичный член до вида

$$\begin{aligned}\{f(\mathbf{x}) - t\}^2 &= \{f(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &= \{f(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{f(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} + \{\mathbb{E}[t|\mathbf{x}] - t\}^2,\end{aligned}$$

где для упрощения условных обозначений вместо $\mathbb{E}[t|\mathbf{x}]$ используется $\mathbb{E}_t[t|\mathbf{x}]$. Подставив в функцию потерь (4.35) и выполнив интегрирование по t , можно убедиться, что перекрестный член исчезает, и тогда получается выражение для функции потерь в виде

$$\mathbb{E}[L] = \int \{f(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x} + \int \text{var}[t|\mathbf{x}] p(\mathbf{x}) d\mathbf{x}. \quad (4.39)$$

Искомая функция $f(\mathbf{x})$ появляется только в первом члене, который будет минимизирован при условии, что $f(\mathbf{x})$ равна $\mathbb{E}[t|\mathbf{x}]$, и в этом случае этот член исчезает. Это всего лишь выведенный ранее результат, который показывает, что оптимальный прогноз по методу наименьших квадратов задается условным средним значением. Второй член – это дисперсия распределения t с усреднением по \mathbf{x} . Он представляет собой собственную вариативность целевых данных и может рассматриваться в качестве шума. Поскольку этот член не зависит от $f(\mathbf{x})$, то представляет собой неустранимое минимальное значение функции потерь.

Квадратичные потери не являются единственным возможным выбором функции потерь для регрессии. Далее кратко будет рассмотрено одно простое обобщение квадратичных потерь, называемое *потерями Минковского (Minkowski loss)*, где ожидание задается как

$$\mathbb{E}[L_q] = \iint |f(\mathbf{x}) - t|^q p(\mathbf{x}, t) d\mathbf{x} dt \quad (4.40)$$

и сводится к ожидаемым квадратичным потерям для $q = 2$. На рис. 4.6 показана функция $|f - t|^q$ в зависимости от $f - t$ для различных значений q . Минимум $\mathbb{E}[L_q]$ задается условным средним значением для $q = 2$, условной медианой для $q = 1$ и условной модой для $q \rightarrow 0$ (см. упражнение 4.12).

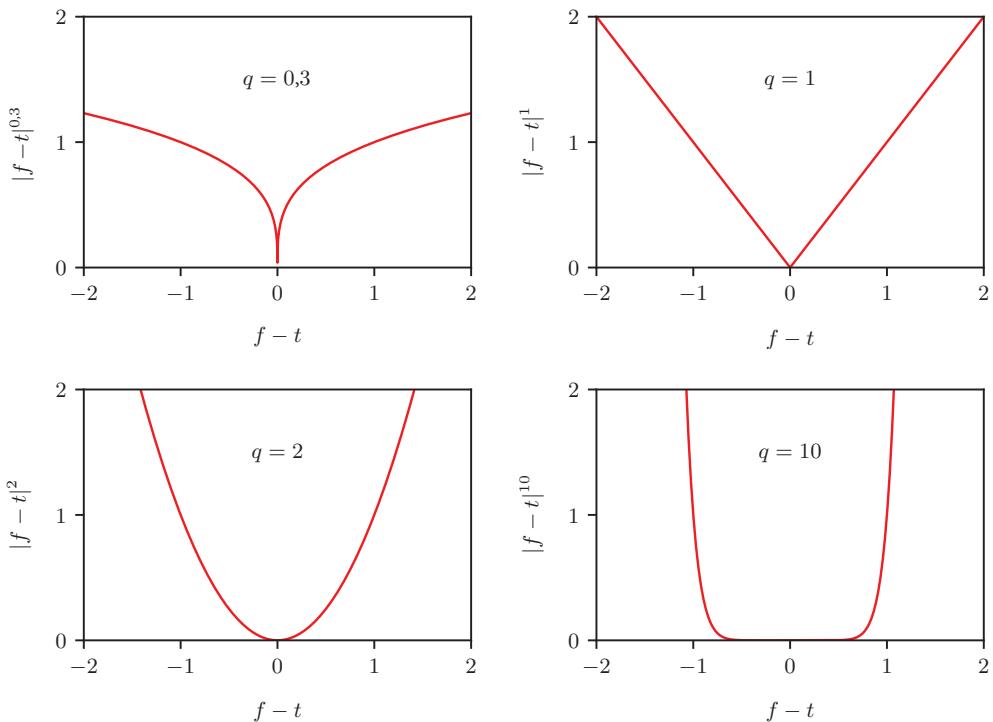


РИС. 4.6 Графики величины $L_q = |f - t|^q$ для различных значений q

Обратите внимание, что предположение о гауссовом шуме подразумевает, что условное распределение t по \mathbf{x} является унимодальным, что может оказаться неприемлемым для некоторых приложений. В этом случае метод квадратичных потерь может привести к очень плохим результатам, и тогда придется использовать более сложные варианты. Например, можно расширить эту модель с использованием смесей гауссовых распределений для получения мультимодальных условных распределений (см. раздел 6.5), которые зачастую возникают при решении обратных задач. В этом разделе основное внимание уделено теории принятия решений для задач регрессии,

а в следующей главе будут рассмотрены аналогичные концепции для задач классификации (см. раздел 5.2).

4.3. Обратное отношение между смещением и дисперсией

До сих пор в процессе обсуждения линейных моделей регрессии подразумевалось, что форма и количество базисных функций заданы изначально (см. раздел 1.2). Также было отмечено, что использование метода максимального правдоподобия может привести к чрезмерной подгонке при обучении сложных моделей на наборах данных ограниченного размера. Однако ограничение числа базисных функций во избежание чрезмерной подгонки имеет побочный эффект в виде ограничения гибкости модели для выявления интересующих и значимых тенденций в данных. Несмотря на то что член регуляризации может контролировать чрезмерную подгонку для моделей с большим количеством параметров, возникает вопрос определения подходящего значения коэффициента регуляризации λ . Поиск решения, которое минимизирует функцию ошибки регуляризации как относительно весового вектора w , так и относительно коэффициента регуляризации λ , определенно не является правильным подходом, поскольку это приводит к решению без регуляризации с $\lambda = 0$.

Здесь уместно рассмотреть проблему сложности модели с точки зрения частотного анализа, известную как *обратное отношение между смещением и дисперсией* (*bias-variance trade-off*). Несмотря на то что эта концепция представлена в контексте моделей линейных базисных функций, где ее проще всего проиллюстрировать на простых примерах, она вполне применима и в более широком контексте. Тем не менее следует отметить, что чрезмерная подгонка является досадным свойством максимального правдоподобия, и она не возникает при маргинализации параметров с использованием байесовского метода (Bishop, 2006).

При рассмотрении теории принятия решений для задач регрессии (см. раздел 4.2) были упомянуты различные функции потерь, каждая из которых приводит к соответствующему оптимальному прогнозному значению при заданном условном распределении $p(t|x)$. Наиболее популярным вариантом является квадратичная функция потерь, для которой оптимальный прогноз определяется условным ожиданием, обозначенным через $h(x)$ и задаваемым как

$$h(x) = \mathbb{E}[t|x] = \int tp(t|x)dt. \quad (4.41)$$

Выше также было отмечено, что ожидаемые квадратичные потери могут быть записаны в виде

$$\mathbb{E}[L] = \int \{f(x) - h(x)\}^2 p(x)dx + \iint \{h(x) - t\}^2 p(x, t)dx dt. \quad (4.42)$$

Напомним, что второй член, который не зависит от $f(\mathbf{x})$, возникает из-за собственного шума данных и представляет собой минимально достижимое значение ожидаемых потерь. Первый член зависит от нашего выбора функции $f(\mathbf{x})$, и поэтому необходимо найти такое решение для $f(\mathbf{x})$, при котором этот член будет минимальным. Поскольку он неотрицателен, наименьшее значение, на которое можно рассчитывать, равно нулю. При наличии неограниченного количества данных (и неограниченных вычислительных ресурсов) вполне можно было бы найти функцию регрессии $h(\mathbf{x})$ с любой желаемой степенью точности, и это было бы оптимальным выбором для $f(\mathbf{x})$. Однако на практике набор данных \mathcal{D} содержит лишь конечное число N точек данных, и, следовательно, невозможно в точности определить функцию регрессии $h(\mathbf{x})$.

Если бы для моделирования $h(\mathbf{x})$ использовалась функция, управляемая вектором параметров w , то с позиций байесовского подхода неопределенность нашей модели выражалась бы через апостериорное распределение по w . Однако с позиций частотного подхода необходимо произвести точечную оценку w на основе набора данных \mathcal{D} и затем попытаться интерпретировать неопределенность этой оценки с помощью следующего мыслительного эксперимента. Предположим, имеется большое количество наборов данных, каждый из которых имеет размер N и каждый берется независимо из распределения $p(t, \mathbf{x})$. Для любого заданного набора данных \mathcal{D} можно запустить алгоритм обучения и получить функцию предсказания $f(\mathbf{x}; \mathcal{D})$. Разные наборы данных из этой совокупности будут давать разные функции и, соответственно, разные значения квадратичных потерь. Для оценки эффективности конкретного алгоритма обучения берется среднее значение по всей совокупности наборов данных.

Рассмотрим интеграл первого члена в (4.42), который для конкретного набора данных \mathcal{D} имеет вид

$$\{f(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2. \quad (4.43)$$

Поскольку эта величина будет зависеть от конкретного набора данных \mathcal{D} , возьмем ее среднее значение по совокупности наборов данных. Если сложить и вычесть величину $\mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})]$ внутри скобок, а затем развернуть, то получится

$$\begin{aligned} & \{f(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] + \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &= \{f(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &+ 2\{f(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}. \end{aligned} \quad (4.44)$$

Возьмем ожидание этого выражения относительно \mathcal{D} и примем к сведению, что последний член исчезнет, что дает

$$\begin{aligned} & \mathbb{E}_{\mathcal{D}}\{f(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 \\ &= \underbrace{\{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{\text{(смещение)}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\{f(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})]\}^2}_{\text{дисперсия}}. \end{aligned} \quad (4.45)$$

Как видно, ожидаемая квадратичная разность между $f(\mathbf{x}; \mathcal{D})$ и функцией регрессии $h(\mathbf{x})$ может быть выражена в виде суммы двух членов. Первый член, называемый *квадратичным смещением* (*squared bias*), представляет собой степень, в которой среднее предсказание по всем наборам данных отличается от желаемой функции регрессии. Второй член, называемый *дисперсией* (*variance*), измеряет степень различия решений для отдельных наборов данных вблизи их среднего значения и, следовательно, определяет, в какой степени функция $f(\mathbf{x}; \mathcal{D})$ чувствительна к конкретному выбору набора данных. Вскоре при рассмотрении простого примера будут приведены некоторые интуитивные соображения в поддержку этих определений.

До сих пор рассматривалось единственное входное значение \mathbf{x} . Если представить это разложение обратно в (4.42), получится следующее разложение ожидаемых квадратичных потерь:

$$\text{ожидаемые потери} = (\text{смещение})^2 + \text{дисперсия} + \text{шум}, \quad (4.46)$$

где

$$(\text{смещение})^2 = \int \{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x}, \quad (4.47)$$

$$\text{дисперсия} = \int \mathbb{E}_{\mathcal{D}}\{f(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})]\}^2 p(\mathbf{x}) d\mathbf{x}, \quad (4.48)$$

$$\text{шум} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt, \quad (4.49)$$

при этом члены смещения и дисперсии теперь соответствуют интегральным величинам.

Наша цель – минимизировать ожидаемые потери, которые разложены на сумму (квадратичного) смещения, дисперсии и постоянного шума. Как будет показано далее, между смещением и дисперсией существует определенный компромисс: очень гибкие модели имеют низкое смещение и высокую дисперсию, а относительно жесткие модели – высокое смещение и низкую дисперсию. Модель с оптимальной возможностью прогнозирования является той, которая приводит к наилучшему балансу между смещением и дисперсией. Это можно проиллюстрировать на примере представленного ранее набора синусоидальных данных (см. раздел 1.2). Здесь независимо генерируется 100 наборов данных, каждый из которых содержит $N = 25$ точек данных, по синусоидальной кривой $h(\mathbf{x}) = \sin(2\pi\mathbf{x})$. Наборы данных индексируются $l = 1, \dots, L$, где $L = 100$. Для каждого набора данных $\mathcal{D}^{(l)}$ подбирается модель с $M = 24$ гауссовыми базисными функциями вместе с постоянной базисной функцией «смещения», что дает в общей сложности 25 параметров. Минимизируя регуляризованную функцию ошибки (4.26), получаем функцию предсказания $f^{(l)}(\mathbf{x})$, как показано на рис. 4.7. В левом столбце показан результат подгонки модели к наборам данных для различных значений $\ln \lambda$ (для наглядности показаны только 20 из 100 подгонок). В правой колонке показано соответствующее среднее значение из 100 подгонок (красный цвет), а также синусоидальная функция, на основе которой были получены наборы данных (зеленый цвет).

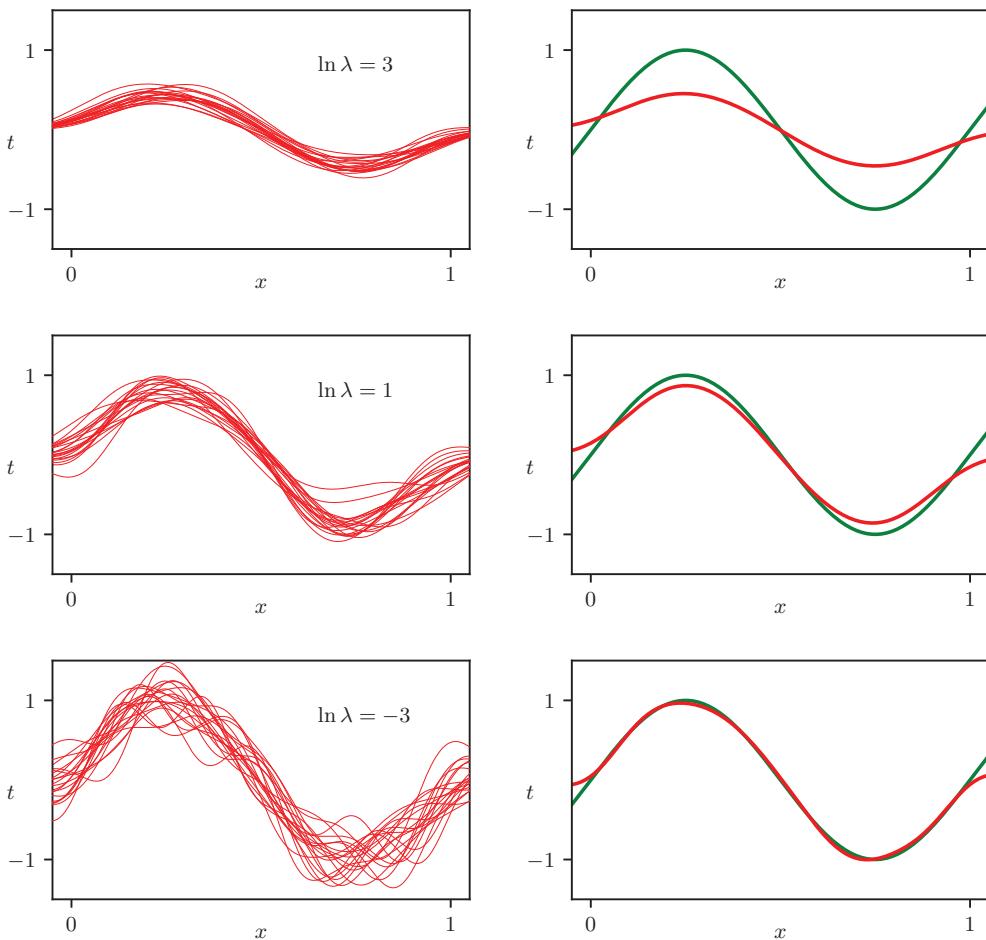


РИС. 4.7 Иллюстрация зависимости смещения и дисперсии от сложности модели, регулируемой параметром регуляризации λ , на примере синусоидальных данных из главы 1

Верхний ряд соответствует большому значению коэффициента регуляризации λ , который дает низкую дисперсию (поскольку красные кривые на левом графике похожи), но высокое смещение (поскольку две кривые на правом графике сильно отличаются). И наоборот, в нижнем ряду при малых значениях λ дисперсия велика (что видно по большой вариации между красными кривыми на левом графике), но смещение мало (что видно по хорошему соответствии между средним значением модели и исходной синусоидальной функцией). Обратите внимание, что результат получения средних значений многих решений для сложной модели с $M = 25$ очень хорошо соответствует функции регрессии, что свидетельствует о целесообразности усреднения. Действительно, взвешенное среднее значение множества решений лежит в основе байесовского подхода, хотя усреднение происходит по апостериорному распределению параметров, а не по множеству наборов данных.

На этом примере также можно количественно рассмотреть компромисс между смещением и дисперсией. Среднее предсказание оценивается по формуле:

$$\bar{f}(x) = \frac{1}{L} \sum_{l=1}^L f^{(l)}(x), \quad (4.50)$$

а интегрированное квадратичное смещение и интегрированная дисперсия определяются следующим образом:

$$(\text{смещение})^2 = \frac{1}{N} \sum_{n=1}^N \{\bar{f}(x_n) - h(x_n)\}^2, \quad (4.51)$$

$$\text{дисперсия} = \frac{1}{N} \sum_{n=1}^N \frac{1}{L} \sum_{l=1}^L \{f^{(l)}(x_n) - \bar{f}(x_n)\}^2, \quad (4.52)$$

где интеграл по x , взвешенный по распределению $p(x)$, аппроксимируется конечной суммой по точкам данных, взятых из этого распределения. Эти величины вместе с их суммой показаны на рис. 4.8 как функция $\ln \lambda$. Минимальное значение «(смещение)² + дисперсия» имеет место в районе $\ln \lambda = 0,43$, что близко к значению, которое дает минимальную ошибку на тестовых данных. Как видно, малые значения λ позволяют модели тонко подстраиваться под шум каждого отдельного набора данных, что приводит к большой дисперсии. И наоборот, большое значение λ тянет весовые параметры к нулю, что приводит к большой погрешности.

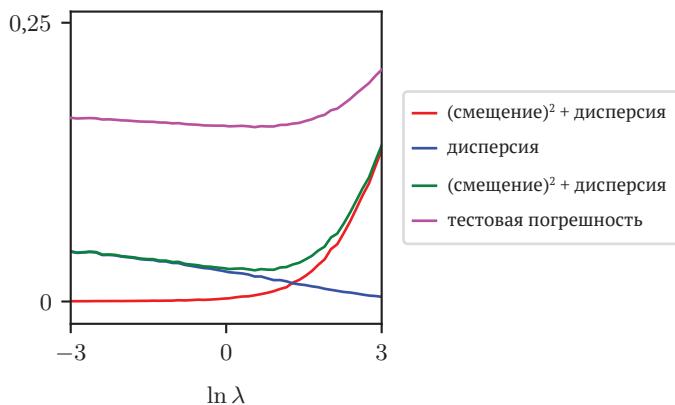


РИС. 4.8 График квадратичного смещения и дисперсии, а также их суммы, соответствующий результатам на рис. 4.7. Также показана средняя ошибка тестового набора для тестового набора данных размером 1000 точек

Обратите внимание, что разложение смещения-дисперсии имеет ограниченную практическую ценность, поскольку оно основано на средних значениях по совокупности наборов данных, в то время как на практике ис-

пользуется только один наблюдаемый набор данных. При наличии большого количества независимых обучающих наборов заданного размера было бы лучше объединить их в один более крупный обучающий набор, что, несомненно, уменьшило бы уровень избыточной подгонки при заданной сложности модели. Тем не менее разложение смещения-дисперсии часто дает полезное понимание проблемы сложности модели, и, хотя в этой главе разложение было представлено с точки зрения проблем регрессии, лежащая в его основе концепция имеет более широкое применение.

Упражнения

- 4.1** (*) Рассмотрим функцию ошибки суммы квадратов, заданную в (1.2), где функция $y(x, \mathbf{w})$ задана многочленом (1.1). Докажите, что коэффициенты $\mathbf{w} = \{w_i\}$, минимизирующие эту функцию ошибки, задаются решением следующего набора линейных уравнений:

$$\sum_{j=0}^M A_{ij} w_j = T_i, \quad (4.53)$$

где

$$A_{ij} = \sum_{n=0}^N (x_n)^{i+j}, \quad T_i = \sum_{n=1}^N (x_n)^i t_n. \quad (4.54)$$

Здесь индексы i или j обозначают индекс компонента, а $(x)^i$ обозначает x в степени i .

- 4.2** (*) Запишите набор связанных линейных уравнений по аналогии с (4.53), удовлетворяемых коэффициентами w_i , которые минимизируют регуляризованную функцию ошибки суммы квадратов, заданную в (1.4).
- 4.3** (*) Покажите, что функция \tanh , определяемая как

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}, \quad (4.55)$$

и логистическая сигмоидная функция, определяемая в (4.6), связаны между собой соотношением

$$\tanh(a) = 2\sigma(2a) - 1. \quad (4.56)$$

Отсюда следует, что общая линейная комбинация логистических сигмоидных функций вида

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^M w_j \sigma\left(\frac{x - \mu_j}{s}\right) \quad (4.57)$$

эквивалентна линейной комбинации тангенциальных функций вида

$$y(x, \mathbf{u}) = u_0 + \sum_{j=1}^M u_j \tanh\left(\frac{x - \mu_j}{2s}\right), \quad (4.58)$$

и теперь необходимо найти выражения, связывающие новые параметры $\{u_1, \dots, u_M\}$ с исходными параметрами $\{w_1, \dots, w_M\}$.

- 4.4** (★★★) Докажите, что матрица

$$\Phi(\Phi^T \Phi)^{-1} \Phi^T \quad (4.59)$$

позволяет взять любой вектор \mathbf{v} и спроектировать его на пространство, охватываемое столбцами Φ . Используйте этот результат, чтобы доказать, что решение по методу наименьших квадратов (4.14) соответствует ортогональной проекции вектора \mathbf{t} на многообразие \mathcal{S} , как показано на рис. 4.3.

- 4.5** (★) Рассмотрим набор данных, в котором каждая точка данных t_n связана с весовым коэффициентом $r_n > 0$, так что функция ошибки суммы квадратов имеет вид:

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N r_n \{t_n - \mathbf{w}^T \varphi(\mathbf{x}_n)\}^2. \quad (4.60)$$

Найдите выражение для решения \mathbf{w}^* , которое минимизирует эту функцию ошибки. Дайте две альтернативные интерпретации функции ошибки взвешенной суммы квадратов в выражениях (i) дисперсии шума, зависящей от данных, и (ii) реплицированных точек данных.

- 4.6** (★) Обратив градиент (4.26) относительно \mathbf{w} в ноль, докажите, что точный минимум регуляризованной функции ошибки суммы квадратов для линейной регрессии задается в (4.27).

- 4.7** (★★) Рассмотрим модель регрессии с линейной базисной функцией для многомерной целевой переменной \mathbf{t} , имеющей гауссово распределение вида

$$p(\mathbf{t} | \mathbf{W}, \Sigma) = \mathcal{N}(\mathbf{t} | \mathbf{y}(\mathbf{x}, \mathbf{W}), \Sigma), \quad (4.61)$$

где

$$\mathbf{y}(\mathbf{x}, \mathbf{W}) = \mathbf{W}^T \varphi(\mathbf{x}) \quad (4.62)$$

вместе с набором обучающих данных, состоящим из входных базисных векторов $\varphi(\mathbf{x}_n)$ и соответствующих целевых векторов \mathbf{t}_n , при $n = 1, \dots, N$. Докажите, что решение максимального правдоподобия \mathbf{W}_{ML} для матрицы параметров \mathbf{W} обладает тем свойством, что каждый столбец задается выражением вида (4.14), которое было решением для изотропного распределения шума. Обратите внимание, что оно не зависит от

ковариационной матрицы Σ . Докажите, что решение максимального правдоподобия для Σ имеет вид:

$$\Sigma = \frac{1}{N} \sum_{n=1}^N (t_n - \mathbf{W}_{\text{ML}}^T \boldsymbol{\varphi}(\mathbf{x}_n))(t_n - \mathbf{W}_{\text{ML}}^T \boldsymbol{\varphi}(\mathbf{x}_n))^T. \quad (4.63)$$

- 4.8** (*) Рассмотрим обобщение квадратичной функции потерь (4.35) для одиночной целевой переменной t на множество целевых переменных, описываемых вектором \mathbf{t} , которое задается как

$$\mathbb{E}[L(\mathbf{t}, \mathbf{f}(\mathbf{x}))] = \int \int \|\mathbf{f}(\mathbf{x}) - \mathbf{t}\|^2 p(\mathbf{x}, \mathbf{t}) d\mathbf{x} d\mathbf{t}. \quad (4.64)$$

Используя вариационное исчисление, докажите, что функция $\mathbf{f}(\mathbf{x})$, для которой минимизируются эти ожидаемые потери, задается формулой:

$$\mathbf{f}(\mathbf{x}) = \mathbb{E}_{\mathbf{t}}[\mathbf{t} | \mathbf{x}]. \quad (4.65)$$

- 4.9** (*) Путем возведения в квадрат в (4.64) выведите результат, аналогичный (4.39), и таким образом докажите, что функция $\mathbf{f}(\mathbf{x})$, минимизирующая ожидаемые квадратичные потери для вектора \mathbf{t} целевых переменных, также задается условным ожиданием \mathbf{t} в форме (4.65).
- 4.10** (**) Переформулируйте результат (4.65), предварительно расширив (4.64) по аналогии с (4.39).
- 4.11** (**) Следующее распределение

$$p(x | \sigma^2, q) = \frac{q}{2(\sigma^2)^{1/q} \Gamma(1/q)} \exp\left(-\frac{|x|^q}{2\sigma^2}\right) \quad (4.66)$$

является обобщением одномерного гауссова распределения. Здесь $\Gamma(x)$ – это гамма-функция, определяемая как

$$\Gamma(x) = \int_{-\infty}^{\infty} u^{x-1} e^{-u} du. \quad (4.67)$$

Докажите, что это распределение нормализовано таким образом, что

$$\int_{-\infty}^{\infty} p(x | \sigma^2, q) dx = 1, \quad (4.68)$$

и что оно сводится к гауссовому при $q = 2$. Рассмотрим регрессионную модель, в которой целевая переменная задается как $t = y(\mathbf{x}, \mathbf{w}) + \epsilon$ и является случайной шумовой переменной, взятой из распределения (4.66). Докажите, что функция логарифмического правдоподобия по \mathbf{w} и σ^2 для наблюдаемого набора входных векторов $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ и соответствующих целевых переменных $\mathbf{t} = (t_1, \dots, t_N)^T$ имеет вид

$$\ln p(\mathbf{t} | \mathbf{X}, \mathbf{w}, \sigma^2) = -\frac{1}{2\sigma^2} \sum_{n=1}^N |y(\mathbf{x}_n, \mathbf{w}) - t_n|^q - \frac{N}{q} \ln(2\sigma^2) + \text{const}, \quad (4.69)$$

где const обозначает члены, не зависящие ни от w , ни от σ^2 . Обратите внимание, что в качестве функции от w используется функция ошибки L_q , рассмотренная в разделе 4.2.

- 4.12** (**) Рассмотрим ожидаемые потери для задач регрессии с функцией потерь L_q , заданной в (4.40). Запишите условие, которому должна удовлетворять $y(x)$, чтобы минимизировать $\mathbb{E}[L_q]$. Докажите, что для $q = 1$ это решение представляет собой условную медиану, т. е. такую функцию $y(x)$, что масса вероятностей для $t < y(x)$ такая же, как и для $t > y(x)$. Также докажите, что минимальные ожидаемые потери L_q для $q \rightarrow 0$ являются условной модой, т. е. функцией $y(x)$, равной такому значению t , которое максимизирует $p(t | x)$ для каждого x .

Глава 5

Однослойные сети: классификация

В предыдущей главе был рассмотрен класс регрессионных моделей, в которых выходные переменные являются линейными функциями параметров модели и которые, следовательно, могут быть представлены в виде простых нейронных сетей с одним слоем весов и параметров смещения. Теперь рассмотрим задачи классификации, и в этой главе речь пойдет об аналогичном классе моделей, которые опять же могут быть выражены в виде однослойных нейронных сетей. Они дадут возможность ввести многие ключевые концепции классификации, прежде чем в последующих главах будут рассмотрены более сложные глубокие нейронные сети.

Задача классификации заключается в том, чтобы взять входной вектор $\mathbf{x} \in \mathbb{R}^D$ и сопоставить его с одним из K дискретных классов \mathcal{C}_k , где $k = 1, \dots, K$. При наиболее распространенном сценарии классы считаются непересекающимися, так что каждый входной сигнал относится к одному и только одному классу. Таким образом, пространство входов разбивается на области принятия решений (*decision regions*), границы которых называются границами принятия решений (*decision boundaries*) или поверхностями принятия решений (*decision surfaces*). В этой главе рассматриваются линейные модели классификации, т. е. поверхности принятия решений являются линейными функциями входного вектора \mathbf{x} и, следовательно, определяются $(D - 1)$ -мерными гиперплоскостями в D -мерном входном пространстве. Наборы данных, классы которых могут быть разделены именно линейными поверхностями принятия решений, называются линейно разделимыми (*linearly separable*). Линейные модели классификации могут быть также применены к наборам данных, которые не являются линейно разделяемыми, однако при этом не все входные данные могут быть классифицированы корректно.

В целом можно выделить три различных подхода к решению задач классификации. Самый простой включает в себя построение дискриминантной функции (*discriminant function*), которая напрямую причисляет каждый вектор \mathbf{x} к определенному классу. Более эффективный подход, напротив, моделирует условные распределения вероятностей $p(\mathcal{C}_k | \mathbf{x})$ на этапе вывода (*inference stage*) и в дальнейшем использует эти распределения для принятия оптимальных решений. Разделение выводов и решений обеспечивает множество положительных моментов (см. раздел 5.2.4). Существует два

различных подхода к определению условных вероятностей $p(\mathcal{C}_k | \mathbf{x})$. Один из них заключается в непосредственном их моделировании, например путем представления их в виде параметрических моделей и последующей оптимизации параметров с помощью обучающего набора. Такой подход называется *дискриминантной вероятностной моделью* (*discriminative probabilistic model*). В качестве альтернативы можно провести моделирование условных плотностей классов $p(\mathbf{x} | \mathcal{C}_k)$ вместе с априорными вероятностями $p(\mathcal{C}_k)$ для классов, а затем вычислить требуемые апостериорные вероятности при помощи теоремы Байеса:

$$p(\mathcal{C}_k | \mathbf{x}) = \frac{p(\mathbf{x} | \mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}. \quad (5.1)$$

Такая модель называется *генеративной вероятностной моделью* (*generative probabilistic model*), поскольку она предоставляет возможность генерировать выборки по каждой из условных плотностей классов $p(\mathbf{x} | \mathcal{C}_k)$. В этой главе будут рассмотрены примеры всех трех подходов: дискриминантные функции, генеративные вероятностные модели и дискриминантные вероятностные модели.

5.1. Дискриминантные функции

Дискриминантной называют функцию, которая принимает входной вектор \mathbf{x} и относит его к одному из K классов, обозначаемых \mathcal{C}_k . В этой главе основное внимание будет уделено *линейным дискриминантам* (*linear discriminants*), а именно тем, для которых поверхности принятия решений являются гиперплоскостями. Чтобы упростить обсуждение, для начала рассмотрим два класса, а затем рассмотрим расширение до $K > 2$ классов.

5.1.1. Два класса

Простейшее представление линейной дискриминантной функции дает линейная функция от входного вектора, так что

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0, \quad (5.2)$$

где \mathbf{w} называется *вектором весовых коэффициентов* (*weight vector*), а w_0 – это *смещение* (*bias*, не путать со смещением в статистическом значении). Входной вектор \mathbf{x} относится к классу \mathcal{C}_1 , если $y(\mathbf{x}) > 0$, и к классу \mathcal{C}_2 в противном случае. Соответствующая граница принятия решения определяется соотношением $y(\mathbf{x}) = 0$, которое соответствует $(D - 1)$ -мерной гиперплоскости в D -мерном пространстве входных данных. Рассмотрим две точки \mathbf{x}_A и \mathbf{x}_B , каждая из которых лежит на поверхности принятия решения. Поскольку $y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$, имеем $\mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$, и, следовательно, вектор \mathbf{w} ортогонален каждому вектору, лежащему в пределах поверхности принятия решений, а значит, \mathbf{w} определяет ориентацию поверхности принятия решений. Точно

так же если \mathbf{x} – это точка на поверхности принятия решений, то $y(\mathbf{x}) = 0$, и поэтому нормальное расстояние от начала координат до поверхности принятия решений определяется как

$$\frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}. \quad (5.3)$$

Таким образом, можно видеть, что параметр смещения w_0 определяет расположение поверхности принятия решений. Эти свойства проиллюстрированы для случая $D = 2$ на рис. 5.1. Поверхность принятия решений, показанная красным цветом, перпендикулярна \mathbf{w} , а ее смещение от начала координат контролируется параметром смещения w_0 . Кроме того, подписанное орто-гональное расстояние общей точки \mathbf{x} от поверхности принятия решений задается $y(\mathbf{x})/\|\mathbf{w}\|$.

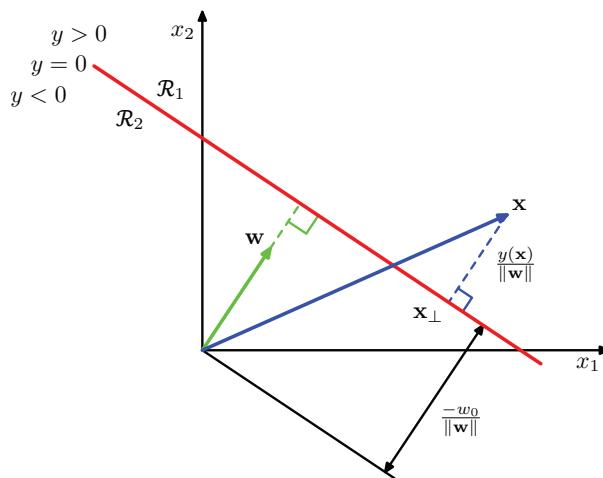


РИС. 5.1 Иллюстрация геометрии линейной дискриминантной функции в двух измерениях

Кроме того, обратите внимание, что значение $y(\mathbf{x})$ является знаковым показателем перпендикулярного расстояния r для точки \mathbf{x} от поверхности принятия решений. Чтобы убедиться в этом, рассмотрим произвольную точку \mathbf{x} , и пусть \mathbf{x}_\perp – это ее ортогональная проекция на поверхность принятия решений, так что

$$\mathbf{x} = \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|}. \quad (5.4)$$

Умножая обе стороны этого выражения на \mathbf{w}^T и добавляя w_0 , а также используя $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$ и $y(\mathbf{x}_\perp) = \mathbf{w}^T \mathbf{x}_\perp + w_0 = 0$, получаем

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}. \quad (5.5)$$

Этот результат проиллюстрирован на рис. 5.1.

Как и в моделях линейной регрессии (см. раздел 4.1.1), иногда бывает удобнее использовать более компактные обозначения, в которых вводится дополнительное фиктивное «входное» значение $x_0 = 1$, а затем определяются $\tilde{\mathbf{w}} = (w_0, \mathbf{w})$ и $\tilde{\mathbf{x}} = (x_0, \mathbf{x})$ так, что

$$y(\mathbf{x}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}. \quad (5.6)$$

В этом случае поверхности принятия решений представляют собой D -мерные гиперплоскости, проходящие через начало координат ($D + 1$)-мерного расширенного пространства входных данных.

5.1.2. Множественные классы

Теперь рассмотрим расширение линейных дискриминантных функций на $K > 2$ классов. Возникает соблазн построить дискриминант для K -классов, объединив несколько дискриминантных функций для двух классов. Однако это приводит к серьезным затруднениям (Duda and Hart, 1973), как будет показано ниже.

Рассмотрим модель с $K - 1$ классификаторами, каждый из которых решает задачу двух классов по отделению точек определенного класса \mathcal{C}_k от точек, не входящих в этот класс. Это называется классификатором «один против остальных» (*one-versus-the-rest*). Слева на рис. 5.2 показан пример с тремя классами, когда такой подход приводит к неоднозначно классифицированным областям входного пространства. Слева показан пример с двумя дискриминантными функциями, предназначенными для различения точек класса \mathcal{C}_k и точек, не относящихся к этому классу. Справа приведен пример с тремя дискриминантными функциями, каждая из которых используется для разделения пары классов \mathcal{C}_k и \mathcal{C}_j .

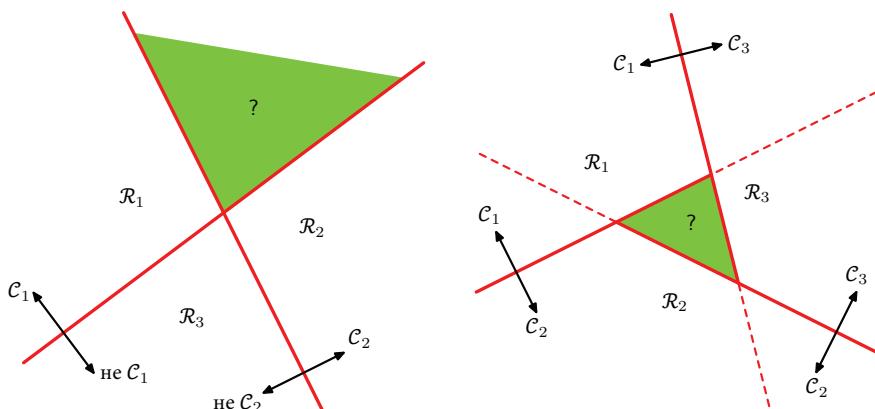


РИС. 5.2 Попытка построить дискриминант K -класса из набора дискриминантных функций двух классов приводит к появлению неоднозначных областей, как показано зеленым цветом

В качестве альтернативного варианта можно ввести $K(K - 1)/2$ бинарных дискриминантных функций, по одной на каждую возможную пару классов. Такой вариант известен как классификатор «один против одного» (*one-versus-one*). Каждая точка классифицируется в соответствии с большинством среди дискриминантных функций. Однако и в этом случае возникает проблема неоднозначных областей, как показано на правой диаграмме рис. 5.2.

Можно избежать этих трудностей, если рассмотреть один K -классовый дискриминант, который включает K линейных функций вида

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}, \quad (5.7)$$

и отнести точку \mathbf{x} к классу \mathcal{C}_k , если $y_k(\mathbf{x}) > y_j(\mathbf{x})$ для всех $j \neq k$. Таким образом, граница принятия решения между классами \mathcal{C}_k и \mathcal{C}_j задается значением $y_k(\mathbf{x}) = y_j(\mathbf{x})$ и, следовательно, соответствует $(D - 1)$ -мерной гиперплоскости, определяемой

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0. \quad (5.8)$$

Она имеет ту же форму, что и граница принятия решения для случая с двумя классами, рассмотренного в разделе 5.1.1, и поэтому для нее применимы аналогичные геометрические свойства.

Области принятия решений такого дискриминанта всегда односвязные и всегда являются выпуклыми. Чтобы убедиться в этом, рассмотрим две точки \mathbf{x}_A и \mathbf{x}_B , каждая из которых лежит внутри области принятия решений \mathcal{R}_k , как показано на рис. 5.3. Любая точка $\hat{\mathbf{x}}$, лежащая на прямой, соединяющей \mathbf{x}_A и \mathbf{x}_B , может быть выражена в виде

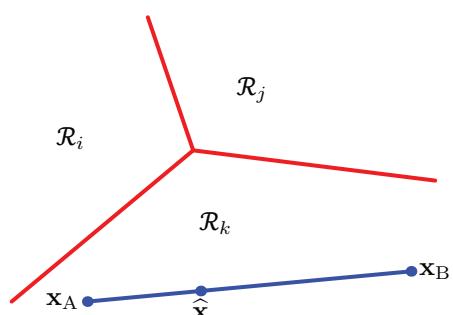
$$\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda) \mathbf{x}_B, \quad (5.9)$$

где $0 \leq \lambda \leq 1$. Из линейности дискриминантных функций следует, что

$$y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B). \quad (5.10)$$

Поскольку и \mathbf{x}_A и \mathbf{x}_B лежат внутри \mathcal{R}_k , из этого следует, что $y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A)$ и что $y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)$ при всех $j \neq k$, а это значит, что $y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}})$ и, следовательно, $\hat{\mathbf{x}}$ также лежит внутри \mathcal{R}_k . Таким образом, \mathcal{R}_k является односвязной и выпуклой.

РИС. 5.3 Иллюстрация областей принятия решений для линейного дискриминанта нескольких классов. Границы принятия решений показаны красным цветом



Обратите внимание, что для двух классов можно либо применить обсуждаемый здесь формализм, основанный на двух дискриминантных функциях $y_1(\mathbf{x})$ и $y_2(\mathbf{x})$, либо использовать более простую, но, по сути, эквивалентную формулировку, основанную на одной дискриминантной функции $y(\mathbf{x})$ (см. раздел 5.1.1).

5.1.3. Кодирование «1 из K»

Для задач регрессии целевая переменная \mathbf{t} – это просто вектор вещественных чисел, значения которых необходимо предсказать. В задачах классификации существуют различные способы использования целевых переменных для представления меток классов. Для задач двух классов наиболее удобным является двоичное представление, в котором имеется одна целевая переменная $t \in \{0, 1\}$ такая, что $t = 1$ представляет класс \mathcal{C}_1 , а $t = 0$ соответствует классу \mathcal{C}_2 . Значение t можно интерпретировать как вероятность того, что класс является \mathcal{C}_1 , при этом значения вероятности принимают только крайние значения 0 и 1. Для случаев классов $K > 2$ удобнее использовать схему кодирования «1 из K» (1-of- K), также известную как схема прямого кодирования (one-hot encoding scheme), в которой \mathbf{t} является вектором длины K , т. е. для класса \mathcal{C}_j все элементы t_k из \mathbf{t} равны нулю, кроме элемента t_j , который принимает значение 1. Например, если имеется $K = 5$ классов, то точке данных из класса 2 будет присвоен целевой вектор:

$$\mathbf{t} = (0, 1, 0, 0, 0)^T. \quad (5.11)$$

Опять же, значение t_k можно интерпретировать как вероятность того, что класс является \mathcal{C}_k , в котором вероятности принимают значения только 0 и 1.

5.1.4. Наименьшие квадраты для классификации

Минимизация функции ошибки по методу суммы квадратов (см. раздел 4.1.3) в моделях линейной регрессии приводит к простому решению в замкнутой форме для значений параметров. Поэтому возникает желание проверить, можно ли применить тот же формализм наименьших квадратов к задачам классификации. Рассмотрим общую задачу классификации с K классами и схемой бинарного кодирования «1 из K» для целевого вектора \mathbf{t} . Одним из аргументов в пользу использования метода наименьших квадратов в таком контексте является возможность аппроксимации условного ожидания $\mathbb{E}[\mathbf{t} | \mathbf{x}]$ целевых значений с учетом входного вектора. Для бинарной схемы кодирования это условное ожидание задается вектором апостериорных вероятностей классов (см. упражнение 5.1). К сожалению, эти вероятности обычно аппроксимируются достаточно плохо, и, более того, аппроксимации могут принимать значения вне диапазона (0, 1). Тем не менее изучить эти простые модели и разобраться в причинах возникновения этих ограничений очень полезно.

Каждый класс \mathcal{C}_k описывается собственной линейной моделью, так что

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}, \quad (5.12)$$

где $k = 1, \dots, K$. Их удобно сгруппировать вместе с помощью векторной нотации, так что

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}, \quad (5.13)$$

где $\tilde{\mathbf{W}}$ – это матрица, k -й столбец которой представляет собой $(D + 1)$ -мерный вектор $\tilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^T)^T$, а $\tilde{\mathbf{x}}$ – это соответствующий дополненный вектор входов $(1, \mathbf{x}^T)^T$ с фиктивным входом $x_0 = 1$. Новый входной сигнал \mathbf{x} относится к тому классу, для которого выходной сигнал $y_k = \tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}}$ является наибольшим.

Теперь определим матрицу параметров $\tilde{\mathbf{W}}$ посредством минимизации функции ошибки по сумме квадратов. Рассмотрим обучающий набор данных $\{\mathbf{x}_n, \mathbf{t}_n\}$, где $n = 1, \dots, N$, и определим матрицу \mathbf{T} , n -й строкой которой является вектор \mathbf{t}_n^T , а также матрицу $\tilde{\mathbf{X}}$, n -й строкой которой является $\tilde{\mathbf{x}}_n^T$. Тогда функция ошибки суммы квадратов может быть записана как

$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2} \text{Tr} \{ (\tilde{\mathbf{X}} \tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}} \tilde{\mathbf{W}} - \mathbf{T}) \}. \quad (5.14)$$

Обратив производную по $\tilde{\mathbf{W}}$ в ноль и сделав перестановку, получим решение для $\tilde{\mathbf{W}}$ в виде

$$\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{T} = \tilde{\mathbf{X}}^\dagger \mathbf{T}, \quad (5.15)$$

где $\tilde{\mathbf{X}}^\dagger$ – это псевдообратная матрица $\tilde{\mathbf{X}}$ (см. раздел 4.1.3). Затем получаем дискриминантную функцию в виде

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}} = \mathbf{T}^T (\tilde{\mathbf{X}}^\dagger)^T \tilde{\mathbf{x}}. \quad (5.16)$$

Интересным свойством решений по методу наименьших квадратов с некоторыми целевыми переменными является тот факт, что если каждый целевой вектор в обучающем множестве удовлетворяет некоторому линейному ограничению

$$\mathbf{a}^T \mathbf{t}_n + b = 0 \quad (5.17)$$

для некоторых постоянных \mathbf{a} и b , то прогноз модели для любого значения \mathbf{x} будет удовлетворять этому же ограничению (см. упражнение 5.3), так что

$$\mathbf{a}^T \mathbf{y}(\mathbf{x}) + b = 0. \quad (5.18)$$

Таким образом, если использовать схему кодирования «1 из K » для K классов, то сделанные моделью прогнозы будут обладать тем свойством, что элементы $\mathbf{y}(\mathbf{x})$ будут равны 1 для любого значения \mathbf{x} . Однако одного этого ограничения по суммированию недостаточно, чтобы результаты модели можно было интерпретировать как вероятности, поскольку они не обязательно будут находиться в интервале $(0, 1)$.

Метод наименьших квадратов дает точное решение в замкнутой форме для параметров дискриминантной функции. Однако даже в качестве дискриминантной функции (когда она используется для принятия решений напрямую и без какой-либо вероятностной интерпретации) она связана с некоторыми серьезными проблемами. Ранее было показано, что функция ошибки суммы квадратов может рассматриваться как отрицательное логарифмическое правдоподобие в предположении о гауссовом распределении шума (см. раздел 2.3.4). Если истинное распределение данных заметно отличается от гауссова, то метод наименьших квадратов может дать неверные результаты. В частности, метод наименьших квадратов очень чувствителен к наличию *выбросов (outliers)*, т. е. точек данных, расположенных на большом расстоянии от основной массы данных. Это показано на рис. 5.4. Здесь можно увидеть, что дополнительные точки данных на правом рисунке приводят к значительному изменению местоположения границы принятия решения, несмотря на то что эти точки были бы правильно классифицированы исходной границей принятия решения на левом рисунке. Функция ошибки суммы квадратов придает слишком большой вес точкам данных, которые находятся на большом расстоянии от границы принятия решения, даже если они правильно классифицированы. Выбросы могут возникать из-за редких событий или просто из-за ошибок в наборе данных. О методах, чувствительных к небольшому количеству точек данных, говорят, что они не обладают достаточной *надежностью (robustness)*. Для сравнения на рис. 5.4 также показаны результаты метода *логистической регрессии (logistic regression)* (см. раздел 5.4.3), который более устойчив к выбросам. Слева показаны данные двух классов (красные крестики и синие кружки), а также границы принятия решений, найденные методом наименьших квадратов (пурпурная кривая) и с помощью модели логистической регрессии (зеленая кривая). Справа показаны соответствующие результаты, полученные при добавлении дополнительных точек данных в правом нижнем углу диаграммы.

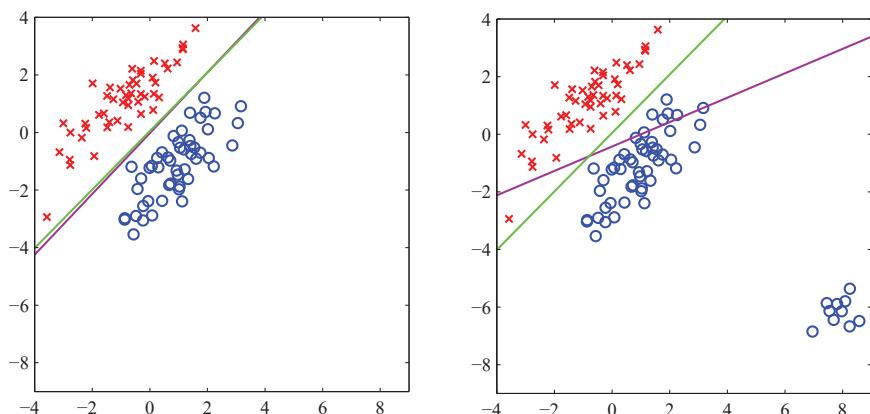


РИС. 5.4 Метод наименьших квадратов, в отличие от логистической регрессии, очень чувствителен к выбросам

Неудачный результат наименьших квадратов не должен нас удивлять, если вспомнить, что он соответствует максимальному правдоподобию в предположении гауссова условного распределения, тогда как бинарные векторы целей явно имеют распределение, далекое от гауссова. Использование более подходящих вероятностных моделей позволяет получить методы классификации, обладающие гораздо лучшими свойствами, чем метод наименьших квадратов, и которые также могут быть обобщены для создания гибких нелинейных нейросетевых моделей, как будет показано в последующих главах.

5.2. Теория принятия решений

При обсуждении линейной регрессии было наглядно показано, что процесс составления прогнозов в машинном обучении можно разбить на два этапа – вывод и принятие решения (см. раздел 4.2). Теперь же более подробно рассмотрим эту перспективу в контексте классификаторов.

Пусть на входе имеется вектор x и соответствующий вектор t целевых переменных, а целью является прогноз t по новому значению x . Для задач регрессии t будет состоять из непрерывных переменных и в общем случае будет вектором, поскольку может возникнуть необходимость предсказать несколько связанных величин. Для задач классификации t будет представлять метки классов. Кроме того, t будет вектором при наличии более двух классов. Совместное распределение вероятностей $p(x, t)$ дает полное представление о неопределенности, связанной с этими переменными. Определение $p(x, t)$ по набору обучающих данных является примером *вывода* (*inference*) и обычно представляет собой очень сложную задачу, решению которой посвящена большая часть этой книги. Однако на практике часто приходится делать определенный прогноз относительно значения t или, в более общем случае, предпринимать конкретные действия на основе имеющихся представлений о том, какие значения может принимать t ; и этот вопрос является предметом изучения теории принятия решений.

Рассмотрим, например, предыдущую задачу медицинской диагностики, в которой необходимо получить изображение поражения кожи у пациента и определить, есть ли у него онкологическое заболевание. В этом случае входной вектор x – это набор интенсивностей пикселей на изображении, а выходная переменная t будет отражать отсутствие онкологии, что будет обозначаться как класс C_1 , или наличие онкологии, что будет обозначаться как класс C_2 . Так, например, можно выбрать t в качестве двоичной переменной, чтобы $t = 0$ соответствовало классу C_1 , а $t = 1$ соответствовало классу C_1 . Позже будет показано, что такой выбор значений меток особенно удобен при работе с вероятностями. Тогда общая задача вывода состоит в определении совместного распределения $p(x, C_k)$ или, эквивалентно, $p(x, t)$, которое дает наиболее полное вероятностное описание переменных. Хотя это может быть очень полезной и информативной величиной, в конечном итоге необходимо

принять решение о том, назначать пациенту лечение или нет, и при этом важно, чтобы этот выбор был оптимальным в соответствии с некоторыми соответствующими критериями (Duda and Hart, 1973). Это и есть *этап принятия решения (decision step)*, а задача теории принятия решений состоит в определении того, как принимать оптимальные решения с учетом соответствующих вероятностей. В дальнейшем будет показано, что этап принятия решения обычно весьма прост и даже тривиален, если решить задачу вывода. Здесь представлено введение в ключевые идеи теории принятия решений, что необходимо для изучения остальной части книги. Дополнительную информацию, а также более подробное изложение можно найти в работах (Berger, 1985) и (Bather, 2000).

До проведения более детального анализа необходимо рассмотреть неформальные предположения о том, какую роль могут играть вероятности в принятии решений. Получив изображение кожи \mathbf{x} нового пациента, необходимо определить, к какому из двух классов отнести это изображение. Поэтому в зависимости от полученного изображения можно определить вероятности двух классов, которые обозначаются $p(\mathcal{C}_k | \mathbf{x})$. Используя теорему Байеса, эти вероятности можно выразить в виде

$$p(\mathcal{C}_k | \mathbf{x}) = \frac{p(\mathbf{x} | \mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}. \quad (5.19)$$

Обратите внимание, что любая из величин, фигурирующих в теореме Байеса, может быть получена из совместного распределения $p(\mathbf{x}, \mathcal{C}_k)$ путем маргинализации или вычисления с условиями относительно соответствующих переменных. Теперь можно интерпретировать $p(\mathcal{C}_k)$ как априорную вероятность для класса \mathcal{C}_k , а $p(\mathcal{C}_k | \mathbf{x})$ – как соответствующую апостериорную вероятность. Таким образом, $p(\mathcal{C}_1)$ представляет собой вероятность того, что человек болен онкологическим заболеванием до получения изображения. Точно так же $p(\mathcal{C}_1 | \mathbf{x})$ – это апостериорная вероятность, скорректированная с помощью теоремы Байеса в свете информации с изображения. Если наша цель заключается в минимизации вероятности отнесения \mathbf{x} к неправильному классу, то, руководствуясь интуицией, следует выбрать класс с большей апостериорной вероятностью. Теперь посмотрим, верна ли эта идея, а также обсудим более общие критерии для принятия решений.

5.2.1. Коэффициент ошибок классификации

Предположим, что целью нашей работы является сокращение числа ошибочных классификаций до возможного минимума. Для этого понадобится правило, которое позволит отнести каждое значение \mathbf{x} к одному из доступных классов. Такое правило разделит пространство входных данных на области \mathcal{R}_k , называемые *областями принятия решений (decision regions)*, по одной для каждого класса, при этом все точки из \mathcal{R}_k будут отнесены к классу \mathcal{C}_k . Границы между такими областями называются *границами принятия*

решений (decision boundaries) или *поверхностями принятия решений (decision surfaces)*). Обратите внимание, что каждая область принятия решений не обязательно должна быть непрерывной, а может состоять из некоторого количества непересекающихся областей. Чтобы найти оптимальное правило принятия решений, рассмотрим для начала случай с двумя классами, как, например, в задаче об онкологических заболеваниях. Ошибка возникает в том случае, когда входной вектор, принадлежащий классу \mathcal{C}_1 , приписывается классу \mathcal{C}_2 , или наоборот. Вероятность того, что это произойдет, определяется как

$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}. \end{aligned} \quad (5.20)$$

Мы вольны выбрать правило принятия решений, которое отнесет каждую точку \mathbf{x} к одному из двух классов. Вполне очевидно, что для минимизации $p(\text{mistake})$ [$p(\text{ошибки})$] необходимо сделать так, чтобы каждая точка \mathbf{x} была отнесена к тому классу, который имеет меньшее значение интеграла в (5.20). Таким образом, если $p(\mathbf{x}, \mathcal{C}_1) > p(\mathbf{x}, \mathcal{C}_2)$ для данного значения \mathbf{x} , то следует отнести эту точку \mathbf{x} к классу \mathcal{C}_1 . Из правила произведения вероятностей следует, что $p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k | \mathbf{x})p(\mathbf{x})$. Поскольку коэффициент $p(\mathbf{x})$ является общим для обоих условий, можно переформулировать этот результат следующим образом: минимальная вероятность совершить ошибку достигается в том случае, если каждое значение \mathbf{x} будет отнесено к классу, для которого апостериорная вероятность $p(\mathcal{C}_k | \mathbf{x})$ является наибольшей. Этот результат проиллюстрирован для двух классов и одной входной переменной x на рис. 5.5. Здесь значения $\mathbf{x} \geq \hat{x}$ относятся к классу \mathcal{C}_2 , и, следовательно, принадлежат области принятия решений \mathcal{R}_2 , тогда как точки $\mathbf{x} < \hat{x}$ относятся к классу \mathcal{C}_1 и принадлежат к \mathcal{R}_1 . Ошибки возникают в синей, зеленой и красной областях, так что для $x < \hat{x}$ ошибки связаны с тем, что точки из класса \mathcal{C}_2 неправильно классифицируются как \mathcal{C}_1 (представленные суммой красной и зеленой областей). И наоборот, для точек в области $x \geq \hat{x}$ ошибки связаны с тем, что точки из класса \mathcal{C}_1 неправильно классифицируются как \mathcal{C}_2 (представлены синей областью). При изменении положения \hat{x} границы принятия решения, как показано красной двунаправленной стрелкой в (a), суммарная площадь синей и зеленой областей остается постоянной, в то время как размер красной области изменяется. Оптимальный выбор \hat{x} находится в точке пересечения кривых $p(\mathbf{x}, \mathcal{C}_1)$ и $p(\mathbf{x}, \mathcal{C}_2)$, как показано на (b) и соответствует $\hat{x} = x_0$, поскольку в этом случае красная область исчезает. Это эквивалентно правилу принятия решений с минимальным коэффициентом ошибочной классификации, которое относит каждое значение \mathbf{x} к классу с большей апостериорной вероятностью $p(\mathcal{C}_k | \mathbf{x})$.

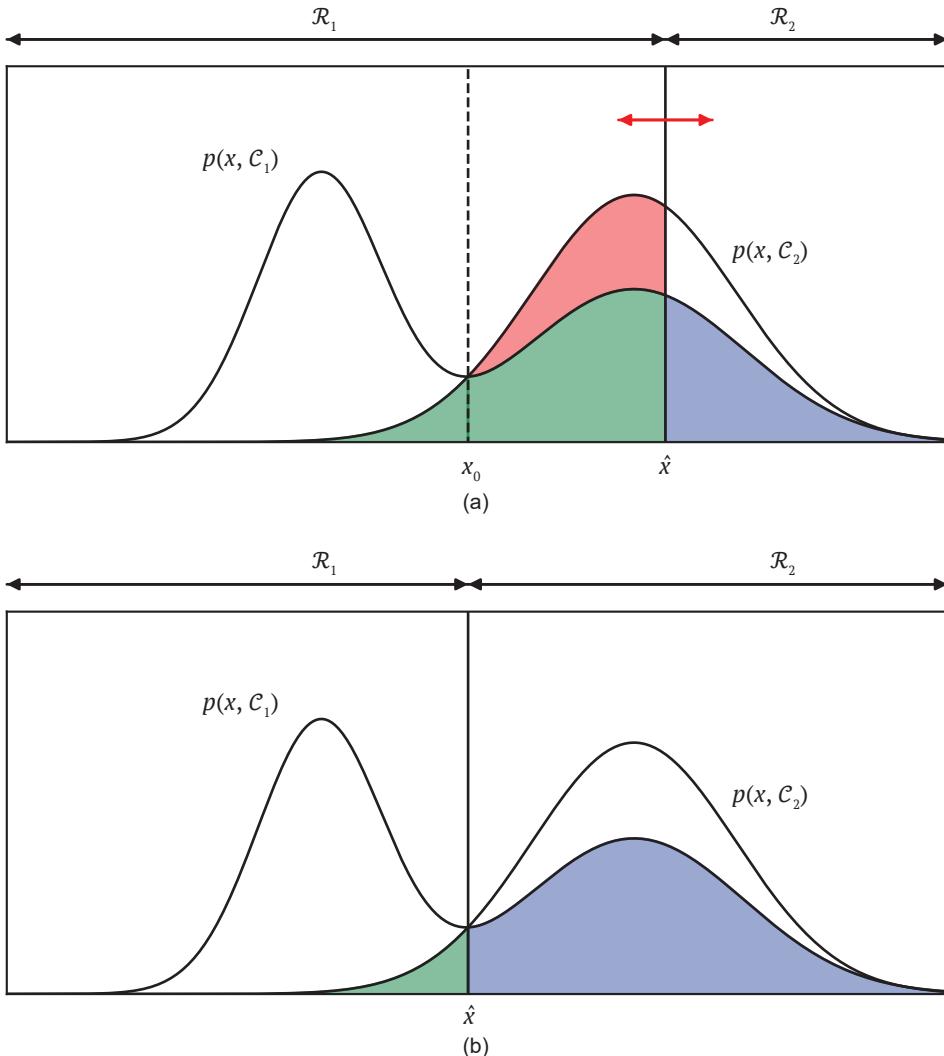


РИС. 5.5 Схематическое изображение совместных вероятностей $p(\mathbf{x}, \mathcal{C}_k)$ для каждого из двух классов, построенное в зависимости от \mathbf{x} , вместе с границей принятия решения $\mathbf{x} = \hat{\mathbf{x}}$

Для более общего случая с K классами гораздо проще максимизировать вероятность правильного решения [$p(\text{correct}) - p(\text{правильное})$], для чего используется выражение

$$\begin{aligned} p(\text{correct}) &= \sum_{k=1}^K p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) \\ &= \sum_{k=1}^K \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}, \end{aligned} \quad (5.21)$$

значение которого максимально при выборе областей \mathcal{R}_k таким образом, чтобы каждая точка \mathbf{x} была отнесена к тому классу, для которого $p(\mathbf{x}, \mathcal{C}_k)$ наибольшее. Опять же, используя правило произведения $p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k | \mathbf{x})p(\mathbf{x})$ и принимая во внимание, что коэффициент $p(\mathbf{x})$ является общим для всех членов, можно убедиться, что для каждого \mathbf{x} следует назначить класс с наибольшей апостериорной вероятностью $p(\mathcal{C}_k | \mathbf{x})$.

5.2.2. Ожидаемые потери

Для многих приложений поставленная задача будет более сложной, чем просто минимизация числа ошибочных классификаций. Рассмотрим еще раз проблему медицинской диагностики. В случае, если пациенту, у которого нет онкологического заболевания, неверно поставлен диагноз «рак», последствия могут заключаться в том, что он будет испытывать определенный стресс, плюс возникнет необходимость в проведении дополнительных исследований. И наоборот, если пациенту с онкологией ставят диагноз «здоров», результатом может стать преждевременная смерть из-за отсутствия лечения. Таким образом, последствия этих двух типов ошибок кардинально отличаются. Однозначно лучше совершать меньше ошибок второго рода, даже если это будет связано с увеличением количества ошибок первого рода.

Формализовать эти вопросы можно с помощью функции потерь (*loss function*), также называемой функцией стоимости (*cost function*), которая представляет собой единую общую меру потерь, возникающих при принятии любого из возможных решений или действий. В этом случае нашей целью является минимизация общих потерь. Обратите внимание, что некоторые авторы вместо этого рассматривают функцию выгоды (*utility function*), значение которой пытаются максимизировать. Это эквивалентные понятия, если считать, что выгода – это просто отрицательная оценка потерь. Однако на протяжении всей этой книги будет использоваться обозначение функции потерь. Предположим, что для нового значения \mathbf{x} истинным классом является \mathcal{C}_k , а мы причисляем \mathbf{x} к классу \mathcal{C}_j (где j может быть равно k , а может и не быть). При этом возникает определенный уровень потерь, который обозначается как L_{kj} , что можно рассматривать в качестве элементов k, j матрицы потерь. Например, в примере с онкологическими заболеваниями можно получить матрицу потерь в виде, показанном на рис. 5.6. Строки соответствуют истинному классу, а столбцы – назначению класса, которое было определено с помощью критерия принятия решения. Эта специальная матрица потерь говорит о том, что при правильном решении нет никаких потерь, а при диагностировании онкологии у здорового пациента потери равны 1, в то время как при диагностировании больного раком пациента как здорового потери равны 100.

РИС. 5.6 Пример матрицы потерь с элементами L_{kj} для задачи диагностики онкологии

	Здоровые	Онкология
Здоровые	0	1
Онкология	100	0

Оптимальным решением будет то, которое минимизирует функцию потерь. Однако функция потерь зависит от истинного класса, который неизвестен. Для заданного входного вектора \mathbf{x} наша неопределенность в отношении истинного класса выражается через совместное распределение вероятностей $p(\mathbf{x}, \mathcal{C}_k)$, и потому необходимо минимизировать средние потери, где среднее значение вычисляется относительно этого распределения и задается как

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}. \quad (5.22)$$

Каждый \mathbf{x} может быть независимо отнесен к одной из областей принятия решений \mathcal{R}_j . Нашей целью является выбор областей \mathcal{R}_j для минимизации ожидаемых потерь (5.22), а это означает, что для каждого \mathbf{x} необходимо минимизировать $\sum_k L_{kj} p(\mathbf{x}, \mathcal{C}_k)$. Как и раньше, для исключения общего множителя $p(\mathbf{x})$ можно использовать правило произведения $p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k | \mathbf{x})p(\mathbf{x})$. Таким образом, правило принятия решений, минимизирующее ожидаемые потери, относит каждый новый \mathbf{x} к тому классу j , для которого количество

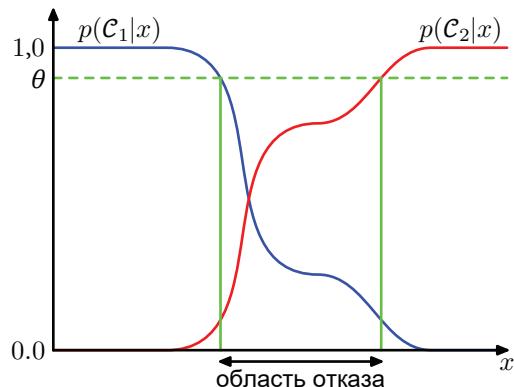
$$\sum_k L_{kj} p(\mathcal{C}_k | \mathbf{x}) \quad (5.23)$$

является минимальным. После выбора значений элементов матрицы потерь L_{kj} это становится очевидной тривиальной задачей.

5.2.3. Опция отказа

Ранее уже было замечено, что ошибки классификации возникают в тех областях пространства входных данных, где наибольшая из апостериорных вероятностей $p(\mathcal{C}_k | \mathbf{x})$ значительно меньше единицы или, что эквивалентно, где совместные распределения $p(\mathbf{x}, \mathcal{C}_k)$ имеют сравнимые значения. Это области, в которых существует относительная неопределенность относительно принадлежности к определенному классу. В некоторых приложениях целесообразно избегать принятия решений в трудных случаях с расчетом на получение более низкого коэффициента ошибок в примерах, по которым принимается классификационное решение. Такой способ известен как *опция отказа* (*reject option*). Например, в гипотетическом примере диагностики онкологических заболеваний может быть уместным использование автоматической системы для классификации тех изображений, для которых нет сомнений в правильности определения класса, в то время как для классификации более неоднозначных случаев необходимо обратиться за проведением биопсии. Для этого необходимо ввести порог θ и отбрасывать те входные данные \mathbf{x} , для которых наибольшая из апостериорных вероятностей $p(\mathcal{C}_k | \mathbf{x})$ меньше или равна θ . На рис. 5.7 это представлено для двух классов и одной непрерывной входной переменной \mathbf{x} . Обратите внимание, что при $\theta = 1$ будут отвергнуты все образцы, тогда как при наличии K классов значение $\theta < 1/K$ будет гарантией того, что ни один образец не будет отвергнут. Получается, что доля отклоненных примеров определяется значением θ .

РИС. 5.7 Иллюстрация опции отказа. Входные данные x с большей из двух апостериорных вероятностей, меньшей или равной некоторому порогу θ , будут отклонены



Критерий отказа можно легко расширить для минимизации ожидаемых потерь, если задана матрица потерь и учитываются потери, возникающие при принятии решения об отказе (см. упражнение 5.10).

5.2.4. Вывод и принятие решения

Задача классификации состоит из двух отдельных этапов: *этапа вывода (inference stage)*, на котором на обучающих данных обучается модель для $p(\mathcal{C}_k | \mathbf{x})$, и *последующего этапа принятия решения (decision stage)*, на котором эти апостериорные вероятности используются для оптимального распределения по классам. В качестве альтернативы можно решить обе задачи вместе и просто получить функцию, которая сопоставляет входные данные \mathbf{x} непосредственно с принимаемыми решениями. Такая функция называется *дискриминантной (discriminant function)*.

На самом деле можно выделить три различных подхода к решению проблем принятия решений, каждый из которых используется в практических приложениях. Ниже приведены эти подходы в порядке убывания сложности.

(а) Сначала решается задача вывода, которая состоит в определении распределения плотностей $p(\mathbf{x} | \mathcal{C}_k)$ для каждого класса \mathcal{C}_k в отдельности. Для этого отдельно определяются априорные вероятности классов $p(\mathcal{C}_k)$. Затем используется теорема Байеса в форме

$$p(\mathcal{C}_k | \mathbf{x}) = \frac{p(\mathbf{x} | \mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} \quad (5.24)$$

для получения апостериорных вероятностей классов $p(\mathcal{C}_k | \mathbf{x})$. Как всегда, знаменатель в теореме Байеса можно найти по величинам в числителе, воспользовавшись формулой

$$p(\mathbf{x}) = \sum_k p(\mathbf{x} | \mathcal{C}_k)p(\mathcal{C}_k). \quad (5.25)$$

Точно так же можно произвести прямое моделирование совместного распределения $p(\mathbf{x} | \mathcal{C}_k)$, а затем нормализовать его для получения апостериор-

ных вероятностей. Найдя апостериорные вероятности, можно определить принадлежность к классу для каждого нового входа \mathbf{x} с помощью теории принятия решений. Подходы, которые явно или неявно моделируют распределение входов и выходов, называются *генеративными моделями (generative models)*, поскольку с их помощью можно генерировать синтетические точки данных в пространстве входов.

(б) Вначале решается задача вывода, которая заключается в определении апостериорных вероятностей классов $p(\mathcal{C}_k | \mathbf{x})$, а затем используется теория принятия решений для отнесения каждого нового \mathbf{x} к одному из классов. Подходы, которые моделируют апостериорные вероятности напрямую, называются *дискриминативными моделями (discriminative models)*.

(с) Нужно найти функцию $f(\mathbf{x})$, называемую дискриминантной функцией, которая отображает каждый новый входной \mathbf{x} непосредственно на метку класса. Например, для задач с двумя классами функция $f(\cdot)$ может быть двоичной и такой, что $f = 0$ представляет класс \mathcal{C}_1 , а $f = 1$ представляет класс \mathcal{C}_2 . В этом случае вероятности не играют никакой роли.

Рассмотрим отличительные особенности этих трех альтернатив. Подход (а) является наиболее критичным, поскольку предполагает нахождение совместного распределения как по \mathbf{x} , так и по \mathcal{C}_k . Для многих приложений \mathbf{x} будет иметь высокие показатели размерности, и, следовательно, может возникнуть необходимость в большом обучающем наборе для определения плотностей распределения по классам с достаточной точностью. Обратите внимание на то, что априорные вероятности классов $p(\mathcal{C}_k)$ часто могут быть оценены просто по доле точек данных обучающего набора в каждом из классов. Однако одним из преимуществ подхода (а) является то, что он также позволяет определить маргинальную плотность данных $p(\mathbf{x})$ из (5.25). Это может быть полезно для выявления новых точек данных с низкой вероятностью по модели, для которых предсказания могут обладать низкой точностью. Этот процесс известен как выявление выбросов (outlier detection) или выявление новизны (novelty detection) (Bishop, 1994; Tarassenko, 1995).

Однако если задача состоит только в принятии классификационных решений, то поиск совместного распределения $p(\mathbf{x}, \mathcal{C}_k)$ может оказаться слишком расточительным в плане затраченных вычислительных ресурсов и чрезмерно требовательным к данным, тогда как на самом деле нужно получить только апостериорные вероятности $p(\mathcal{C}_k | \mathbf{x})$, а их можно получить непосредственно с помощью подхода (б). Действительно, плотности распределения по классам могут содержать значительное количество структур, которые мало влияют на апостериорные вероятности, как показано на рис. 5.8. Обратите внимание, что левая мода в распределении плотности по классам $p(\mathbf{x} | \mathcal{C}_1)$, показанная синим цветом на левом графике, не влияет на апостериорные вероятности. Вертикальная зеленая линия на правом графике показывает границу принятия решения по \mathbf{x} , которая дает минимальный процент ошибочной классификации при условии, что предшествующие вероятности классов, $p(\mathcal{C}_1)$ и $p(\mathcal{C}_2)$, равны. В последнее время наблюдается большой интерес к изучению

относительных достоинств генеративного и дискриминативного подходов к машинному обучению, а также к поиску способов их сочетания (Jebara, 2004; Lasserre, Bishop and Minka, 2006).

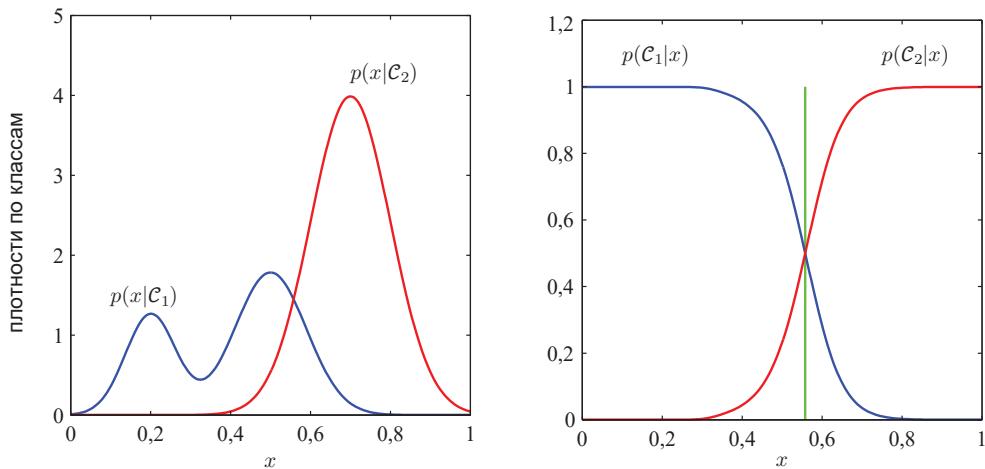


РИС. 5.8 Пример распределения плотностей по классам для двух классов с одной входной переменной x (левый график) вместе с соответствующими апостериорными вероятностями (правый график)

Подход (с) еще проще: на основании обучающих данных определяется дискриминантная функция $f(x)$, которая привязывает каждое значение x непосредственно к метке класса, тем самым объединяя этапы вывода и принятия решения в одну задачу обучения. В примере на рис. 5.8 это соответствует нахождению значения x , показанного вертикальной зеленой линией, поскольку именно эта граница принятия решения дает минимальную вероятность ошибочной классификации.

Однако в варианте (с) нет доступа к апостериорным вероятностям $p(\mathcal{C}_k | \mathbf{x})$. Существует множество весомых причин для того, чтобы вычислить апостериорные вероятности, даже если впоследствии они будут использоваться для принятия решений. К ним относятся:

- **минимизация риска.** Рассмотрим задачу, в которой элементы матрицы потерь периодически подвергаются пересмотру (как, например, в финансовых приложениях). При наличии данных об апостериорных вероятностях можно практически полностью пересмотреть критерий принятия решения о минимальном риске, соответствующим образом изменив (5.23). Если же имеется только дискриминантная функция, то любое изменение матрицы потерь потребует возврата к обучающим данным и повторного решения задачи вывода;
- **опция отказа.** Апостериорные вероятности позволяют определить критерий отклонения, который минимизирует процент ошибочной класси-

ификации, или, в более общем случае, ожидаемые потери, для заданной доли отвергнутых точек данных;

- **компенсация априорных вероятностей классов.** Рассмотрим наш пример с диагностикой онкологических заболеваний (см. раздел 2.1.1) и предположим, что для создания автоматизированной системы диагностики было собрано большое количество изображений от населения в целом, которые используются в качестве обучающих данных. Поскольку раковые заболевания среди населения в целом встречаются редко, может оказаться, что, предположим, только 1 из каждого 1000 изображений соответствует признакам онкологического заболевания. Если бы такой набор данных использовался для обучения адаптивной модели, то из-за малого количества примеров из класса онкологических заболеваний могли бы возникнуть серьезные затруднения. Например, классификатор, который отнесет каждую точку к классу здоровых, достигнет точности 99,9 %, и избежать этого будет непросто. Кроме того, даже большой набор данных будет содержать очень небольшое количество примеров изображений кожи, соответствующих онкологическим заболеваниям, поэтому алгоритм обучения не будет иметь широкого спектра примеров таких изображений и, следовательно, вряд ли будет эффективно проводить обобщение. Сбалансированный набор данных с равным количеством примеров из каждого класса позволил бы нам найти более точную модель. Однако в этом случае необходимо компенсировать влияние изменений, внесенных в обучающие данные. Предположим, что с помощью такого модифицированного набора данных были найдены модели для апостериорных вероятностей. Из теоремы Байеса (5.24) следует, что апостериорные вероятности пропорциональны предшествующим вероятностям, которые можно интерпретировать как доли точек в каждом классе. Поэтому достаточно взять апостериорные вероятности, полученные из искусственно сбалансированного набора данных, разделить на доли классов в этом наборе данных, а затем умножить на доли классов в совокупности людей, к которым планируется применить модель. Наконец, необходимо произвести нормализацию, чтобы новые апостериорные вероятности в сумме равнялись единице. Обратите внимание, что эта процедура не может быть применена, если сначала была получена дискриминантная функция, а не определены апостериорные вероятности;
- **объединение моделей.** Для сложных приложений может потребоваться разбиение проблемы на ряд более мелких задач, каждая из которых может быть решена в отдельном модуле. Например, в случае с нашей гипотетической задачей медицинской диагностики можно получить информацию, скажем, из анализов крови, а также по изображениям кожи. Вместо объединения всей этой разнородной информации в одно огромное пространство входных данных, возможно, было бы более эффективным построить одну систему для интерпретации изображений

и другую – для интерпретации данных о крови. Если каждая из двух моделей дает апостериорные вероятности для соответствующих классов, то результаты можно будет систематически объединять при помощи правил вероятности. Один из простых способов решения этой задачи – считать, что для каждого класса в отдельности распределения входных данных для изображений, обозначаемых \mathbf{x}_I , и данных крови, обозначаемых \mathbf{x}_B , независимы, так что

$$p(\mathbf{x}_I, \mathbf{x}_B | \mathcal{C}_k) = p(\mathbf{x}_I | \mathcal{C}_k)p(\mathbf{x}_B | \mathcal{C}_k). \quad (5.26)$$

Это пример *свойства условной независимости* (*conditional independence property*) (см. раздел 11.2), поскольку независимость сохраняется при условии, что распределение обусловлено принадлежностью к классу \mathcal{C}_k . Тогда апостериорная вероятность с учетом данных по изображению и крови определяется как

$$\begin{aligned} p(\mathcal{C}_k | \mathbf{x}_I, \mathbf{x}_B) &\propto p(\mathbf{x}_I, \mathbf{x}_B | \mathcal{C}_k)p(\mathcal{C}_k) \\ &\propto p(\mathbf{x}_I | \mathcal{C}_k)p(\mathbf{x}_B | \mathcal{C}_k)p(\mathcal{C}_k) \\ &\propto \frac{p(\mathcal{C}_k | \mathbf{x}_I)p(\mathcal{C}_k | \mathbf{x}_B)}{p(\mathcal{C}_k)}. \end{aligned} \quad (5.27)$$

Таким образом, необходимо получить априорные вероятности классов $p(\mathcal{C}_k)$, которые несложно оценить по доле точек данных в каждом классе, а затем нормализовать полученные апостериорные вероятности, чтобы их сумма равнялась единице. Особое предположение об условной независимости (5.26) является примером *наивной модели Байеса* (*naive Bayes model*) (см. раздел 11.2.3). Обратите внимание, что для этой модели совместное маргинальное распределение $p(\mathbf{x}_I, \mathbf{x}_B)$, как правило, не факторизуется. В последующих главах будет рассмотрен вопрос о построении моделей для объединения данных, которые не требуют предположения об условной независимости (5.26). Еще одно преимущество использования моделей с выводом вероятностей, а не решений заключается в том, что их легко сделать дифференцируемыми относительно любых настраиваемых параметров (таких как весовые коэффициенты в примере с полиномиальной регрессией), что позволяет компоновать их и совместно обучать с помощью градиентных методов оптимизации (см. главу 7).

5.2.5. Точность классификатора

Самым простым показателем эффективности классификатора является доля правильно классифицированных точек тестового набора. Однако здесь уже говорилось о том, что различные типы ошибок могут иметь разные последствия в виде матрицы потерь, поэтому задача состоит не только в минимизации числа ошибочных классификаций. Изменяя расположение границы принятия решения, можно добиться компромисса между различ-

ными видами ошибок, например, с целью минимизации ожидаемых потерь. Поскольку речь в данном случае идет о настолько важном явлении, введем некоторые дополнительные определения и термины для более точного описания работы классификатора.

Вернемся к примеру диагностики онкологических заболеваний (см. раздел 2.1.1). Для каждого пациента определяется «истинная метка», отмечающая наличие или отсутствие у него злокачественных новообразований, а также прогноз, сделанный классификатором. Если для конкретного пациента классификатор предсказывает наличие онкологического заболевания и это на самом деле является истинной меткой, то такое предсказание называется *истинно положительным* (*true positive*). Однако если у человека нет рака, то такой прогноз называется *ложноположительным* (*false positive*). Точно так же, если классификатор сообщает об отсутствии онкологического заболевания у пациента, и это верно, то такой прогноз называется *истинно отрицательным* (*true negative*), в противном случае он является *ложноотрицательным* (*false negative*). Ложноположительные результаты также известны как *ошибки первого типа* (*type 1 errors*), в то время как ложноотрицательные результаты называются *ошибками второго типа* (*type 2 errors*). Если N – это общее число участников обследования, то N_{TP} – это число истинно положительных результатов, N_{FP} – число ложноположительных результатов, N_{TN} – число истинно отрицательных результатов, а N_{FN} – число ложноотрицательных результатов, где

$$N = N_{\text{TP}} + N_{\text{FP}} + N_{\text{TN}} + N_{\text{FN}}. \quad (5.28)$$

Это можно представить в виде *матрицы неточностей* (*confusion matrix*), как показано на рис. 5.9. Здесь строки соответствуют истинному классу, а столбцы – присвоению класса с помощью нашего критерия принятия решения. Элементы матрицы показывают количество истинно отрицательных, ложноположительных, ложноотрицательных и истинно положительных результатов. Точность, измеряемая долей правильных классификаций, определяется как

$$\text{Accuracy (точность)} = \frac{N_{\text{TP}} + N_{\text{TN}}}{N_{\text{TP}} + N_{\text{FP}} + N_{\text{TN}} + N_{\text{FN}}}. \quad (5.29)$$

Как видно, при сильном дисбалансе классов показатели точности могут вводить в заблуждение. Например, в нашем примере с диагностикой онкологии, где только 1 человек из 1000 может быть болен раком, наивный классификатор, который просто считает, что никто не болен раком, достигнет точности 99,9 %, но будет совершенно бесполезен.

РИС. 5.9 Матрица неточностей для задачи диагностики рака

	Здоровые	Онкология
Здоровые	N_{TN}	N_{FP}
Онкология	N_{FN}	N_{TP}

С помощью этих чисел можно определить несколько других величин, среди которых наиболее часто встречаются такие, как:

$$\text{Точность (precision)} = \frac{N_{\text{TR}}}{N_{\text{TP}} + N_{\text{FP}}}, \quad (5.30)$$

$$\text{Отзыв (recall)} = \frac{N_{\text{TR}}}{N_{\text{TP}} + N_{\text{FN}}}, \quad (5.31)$$

$$\text{Частота ложноположительных результатов} = \frac{N_{\text{FP}}}{N_{\text{FP}} + N_{\text{TN}}}, \quad (5.32)$$

$$\text{Частота ложных срабатываний} = \frac{N_{\text{FP}}}{N_{\text{FP}} + N_{\text{TP}}}. \quad (5.33)$$

В нашем примере с диагностикой онкологических заболеваний точность представляет собой оценку вероятности того, что у человека с положительным результатом теста действительно есть рак, в то время как отзыв – это оценка вероятности того, что человек, у которого есть рак, правильно определен с помощью диагностики. Коэффициент ложноположительных результатов – это оценка вероятности того, что у здорового человека будет обнаружен рак, в то время как коэффициент ложных обнаружений – это доля положительных результатов тестирования у тех, кто на самом деле не болен раком.

Изменяя местоположение границы принятия решения, можно изменить соотношение между двумя видами ошибок. Чтобы лучше понять суть этого баланса, вернемся к рис. 5.5, но теперь обозначим различные области, как показано на рис. 5.10. Помеченные области можно соотнести с различными показателями истинных и ложных ошибок следующим образом:

$$N_{\text{FP}}/N = E, \quad (5.34)$$

$$N_{\text{TP}}/N = D + E, \quad (5.35)$$

$$N_{\text{FN}}/N = B + C, \quad (5.36)$$

$$N_{\text{TN}}/N = A + C, \quad (5.37)$$

где неявно рассматривается предел $N \rightarrow \infty$ для связи числа наблюдений с вероятностями.

5.2.6. ROC-кривая

Вероятностный классификатор выдает апостериорную вероятность, которая может быть преобразована в решение путем задания порогового значения. Варьируя значение этого порогового параметра, можно уменьшить количество ошибок первого типа за счет увеличения ошибок второго типа, или

наоборот. Для лучшего понимания этого соотношения полезно построить так называемую кривую операционной характеристики приемника (receiver operating characteristic curve, или ROC-кривую) (Fawcett, 2006), название которой берет свое начало от методов измерения эффективности работы радиарных приемников. Это график зависимости частоты истинных положительных результатов от частоты ложных положительных результатов, как показано на рис. 5.11. Верхняя синяя кривая на рисунке представляет собой лучший классификатор, чем нижняя красная кривая. Здесь пунктирная кривая представляет производительность простого случайного классификатора.

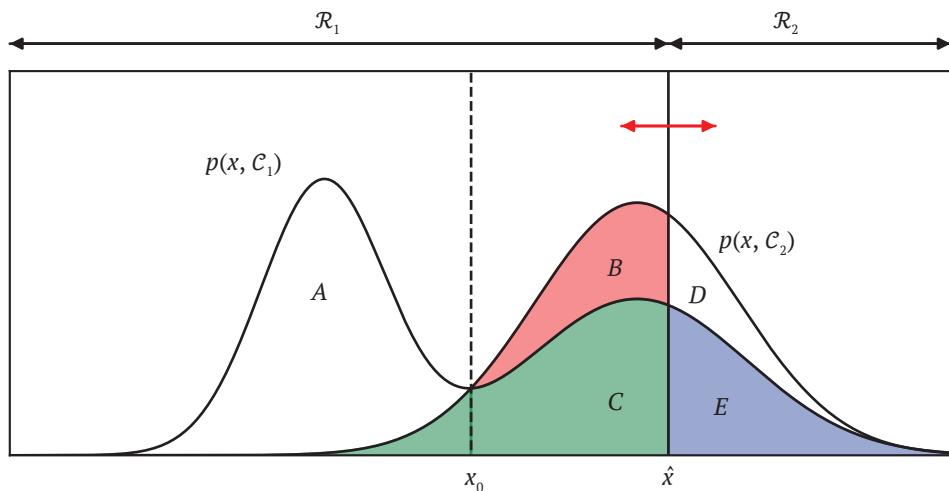
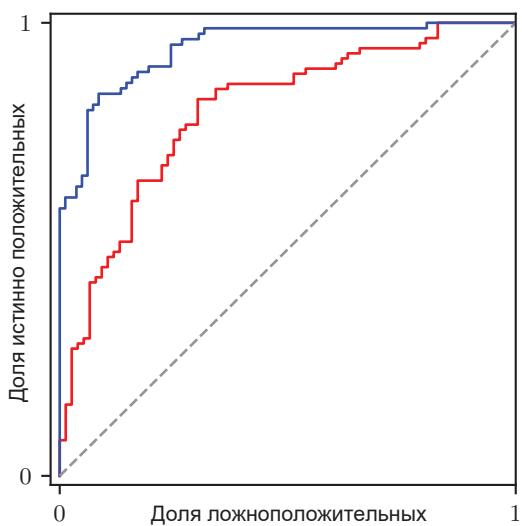


РИС. 5.10 Как и на рис. 5.5 с различными размеченными областями, в задаче классификации онкологии область \mathcal{R}_1 отнесена к классу здоровых, а область \mathcal{R}_2 – к классу больных раком

РИС. 5.11 Характеристика оператора-приемника (ROC) – это график соотношения истинно положительных и ложноположительных результатов как компромисс между ошибками первого и второго типа в задаче классификации



При перемещении границы принятия решения на рис. 5.10 от $-\infty$ до ∞ прослеживается ROC-кривая, которую можно построить путем сопоставления кумулятивной доли верных обнаружений злокачественных новообразований по оси y с кумулятивной долей неверных обнаружений по оси x . Обратите внимание, что конкретная матрица неточностей представляет собой одну точку на ROC-кривой. Наилучший возможный классификатор будет представлен точкой в левом верхнем углу ROC-диаграммы. Левый нижний угол представляет простой классификатор, который относит каждую точку к классу нормы и, следовательно, не имеет истинно положительных, но и не имеет ложноположительных результатов. Точно так же правый верхний угол представляет классификатор, который относит все точки к классу онкологических заболеваний и поэтому не имеет ложноотрицательных, но также не имеет и истинно отрицательных результатов. На рис. 5.11 классификаторы, представленные синей кривой, превосходят классификаторы, представленные красной кривой, при любом выборе, скажем, коэффициента ложноположительных результатов. Однако не исключено, что эти кривые пересекаются, и в этом случае выбор лучшего классификатора будет зависеть от выбора операционной точки.

В качестве базового варианта рассмотрим случайный классификатор, который просто относит каждую точку данных к онкологическим заболеваниям с вероятностью ρ и к норме с вероятностью $1 - \rho$. При изменении значения ρ получится ROC-кривая, выраженная диагональной прямой линией, как показано на рис. 5.11. Любой классификатор, расположенный ниже диагональной линии, показывает худшие результаты, чем просто случайное угадывание.

Иногда полезно получить одно число, которое характеризует всю ROC-кривую. Один из подходов заключается в измерении *площади под кривой* (*area under the curve, AUC*). Значение AUC, равное 0,5, представляет собой случайное угадывание, а значение 1,0 соответствует идеальному классификатору.

Другим показателем является *F-оценка* (*F-score*), которая представляет собой среднее геометрическое значение от точности и отзыва и определяется как

$$F = \frac{2 \times \text{точность} \times \text{отзыв}}{\text{точность} + \text{отзыв}} \quad (5.38)$$

$$= \frac{2N_{\text{TP}}}{2N_{\text{TP}} + N_{\text{FP}} + N_{\text{FN}}}. \quad (5.39)$$

Разумеется, можно также объединить матрицу неточностей на рис. 5.9 с матрицей потерь на рис. 5.6 для вычисления ожидаемых потерь путем точечного умножения элементов и суммирования полученных произведений.

Хотя ROC-кривая может быть расширена более чем на два класса, она стремительно становится слишком громоздкой при увеличении количества классов.

5.3. Генеративные классификаторы

Теперь перейдем к вероятностному представлению о классификации и рассмотрим способы получения моделей с линейными границами принятия решений на основе простых предположений о распределении данных. Ранее в разделе 5.2.4 уже обсуждалось различие между дискриминантным и генеративным подходами к классификации. Здесь же речь пойдет о генеративном подходе, при котором моделируются плотности $p(\mathbf{x} | \mathcal{C}_k)$, обусловленные классами, а также априорные вероятности классов $p(\mathcal{C}_k)$, которые затем используются для вычисления апостериорных вероятностей $p(\mathcal{C}_k | \mathbf{x})$ с помощью теоремы Байеса.

Для начала рассмотрим задачи с двумя классами. Апостериорная вероятность для класса \mathcal{C}_1 может быть записана как

$$\begin{aligned} p(\mathcal{C}_1 | \mathbf{x}) &= \frac{p(\mathbf{x} | \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} | \mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x} | \mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \exp(-a)} = \sigma(a), \end{aligned} \quad (5.40)$$

где определено

$$a = \ln \frac{p(\mathbf{x} | \mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x} | \mathcal{C}_2)p(\mathcal{C}_2)}, \quad (5.41)$$

а $\sigma(a)$ представляет собой логистическую сигмоидную функцию, определяемую как

$$\sigma(a) = \frac{1}{1 + \exp(-a)}, \quad (5.42)$$

что показано на рис. 5.12. Термин «сигмоид» (*sigmoid*) означает S-образную форму. Этот тип функции иногда также называют «сжимающей функцией» (*squashing function*), поскольку она отображает всю вещественную ось на конечный интервал. Логистическая сигмоидная функция уже упоминалась в предыдущих главах. Она играет важную роль во многих алгоритмах классификации. Эта функция удовлетворяет следующему свойству симметрии:

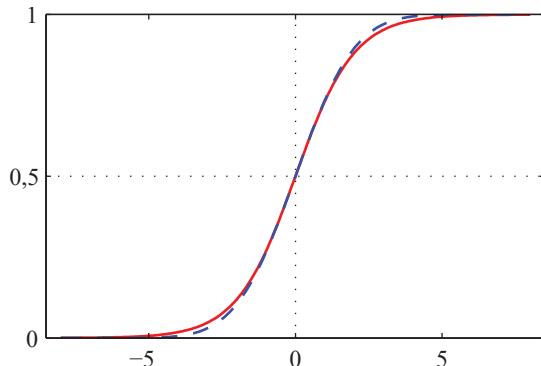
$$\sigma(-a) = 1 - \sigma(a), \quad (5.43)$$

в чем легко можно удостовериться на практике. Инверсия логистической сигмоидной функции задается выражением

$$a = \ln \left(\frac{\sigma}{1 - \sigma} \right) \quad (5.44)$$

и известна как функция *logit*. Она представляет собой логарифм отношения вероятностей $\ln[p(\mathcal{C}_1 | \mathbf{x})/p(\mathcal{C}_2 | \mathbf{x})]$ для двух классов, также известный как *логарифм отношения шансов (log odds)*.

РИС. 5.12 Красным показан график логистической сигмоидной функции $\sigma(a)$, определяемой по (5.42), вместе с масштабированной функцией пробита $\Phi(\lambda_a)$ для $\lambda^2 = \pi/8$ (пунктир синего цвета), где $\Phi(a)$ определяется по (5.86). Масштабный коэффициент $\pi/8$ выбран так, чтобы производные двух кривых были равны при $a = 0$



Обратите внимание, что в (5.40) на самом деле переписываются апостериорные вероятности в эквивалентной форме, поэтому появление логистической сигмоидной формы может показаться ненатуральным. Однако это имеет значение в том случае, если $a(\mathbf{x})$ является ограниченной функциональной формой. В ближайшее время будут рассмотрены ситуации, когда $a(\mathbf{x})$ является линейной функцией по \mathbf{x} , и в этом случае апостериорные вероятности подчиняются обобщенной линейной модели.

Если количество классов $K > 2$, справедливо уравнение

$$\begin{aligned} p(\mathcal{C}_k | \mathbf{x}) &= \frac{p(\mathbf{x} | \mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x} | \mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \end{aligned} \quad (5.45)$$

которое называют *нормализованным экспоненциалом (normalized exponential)* и которое можно рассматривать в качестве обобщения логистической сигмоидной формы для нескольких классов. Здесь значения a_k определяются как

$$a_k = \ln(p(\mathbf{x} | \mathcal{C}_k)p(\mathcal{C}_k)). \quad (5.46)$$

Нормализованная экспоненциальная функция также известна как функция *softmax*, поскольку она представляет собой слаженную версию функции «*max*», потому что если $a_k \gg a_j$ для всех $j \neq k$, то $p(\mathcal{C}_k | \mathbf{x}) \approx 1$, а $p(\mathcal{C}_j | \mathbf{x}) \approx 0$.

Теперь рассмотрим последствия выбора конкретных форм плотностей распределения для классов. Сначала обратимся к непрерывным входным переменным \mathbf{x} , а затем кратко остановимся на дискретных входных переменных.

5.3.1. Непрерывные входные данные

Предположим, что плотности распределения по классам являются гауссовыми. Далее рассмотрим результирующую форму для апостериорных вероятностей. Для начала допустим, что все классы имеют одну и ту же ковариационную матрицу Σ . Таким образом, плотность для класса \mathcal{C}_k определяется как

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right\}. \quad (5.47)$$

Пусть для начала имеется два класса. Из (5.40) и (5.41) следует, что

$$p(\mathcal{C}_1 | \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0), \quad (5.48)$$

где было задано

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \quad (5.49)$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \quad (5.50)$$

Теперь видно, что квадратичные члены в \mathbf{x} от экспонент гауссовых плотностей аннулированы (из-за допущения об общих ковариационных матрицах), что приводит к линейной функции \mathbf{x} в аргументе логистической сигмоидной формы. Этот результат проиллюстрирован для двумерного входного пространства \mathbf{x} на рис. 5.13. На графике слева показаны плотности распределения по классам для двух классов (обозначены красным и синим). Справа показана соответствующая апостериорная вероятность $p(\mathcal{C}_1 | \mathbf{x})$, которая задается логистической сигмоидной формой линейной функции от \mathbf{x} . Поверхность на правом графике окрашена с долей красных чернил, заданной $p(\mathcal{C}_1 | \mathbf{x})$, и долей синих чернил, заданной $p(\mathcal{C}_2 | \mathbf{x}) = 1 - p(\mathcal{C}_1 | \mathbf{x})$. Результирующие границы принятия решений соответствуют поверхностям, вдоль которых

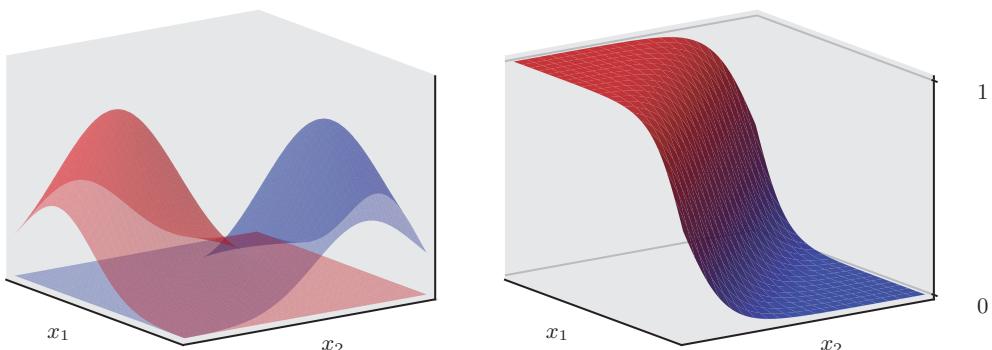


РИС. 5.13 Плотности распределения по классам для двух классов

апостериорные вероятности $p(\mathcal{C}_k | \mathbf{x})$ постоянны, поэтому они будут задаваться линейными функциями от \mathbf{x} , и, следовательно, границы принятия решений линейны во входном пространстве. Априорные вероятности $p(\mathcal{C}_k)$ вводятся только через параметр смещения w_0 , так что изменения в значениях априорных вероятностей приводят к параллельным сдвигам границ принятия решений и (в более общем случае) параллельных контуров постоянной апостериорной вероятности.

Для общего случая K классов апостериорные вероятности задаются в (5.45), где из (5.46) и (5.47) следует, что

$$a_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}, \quad (5.51)$$

где определено

$$\mathbf{w}_k = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k, \quad (5.52)$$

$$w_{k0} = -\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \ln p(\mathcal{C}_k). \quad (5.53)$$

Как видно, $a_k(\mathbf{x})$ снова являются линейными функциями от \mathbf{x} вследствие исключения квадратичных членов, обусловленных общими ковариациями. Результирующие границы принятия решений, соответствующие минимальному уровню ошибочной классификации, возникают при равенстве двух апостериорных вероятностей (двух наибольших) и поэтому определяются линейными функциями от \mathbf{x} . Таким образом, в результате вновь получается обобщенная линейная модель.

Если снять допущение об общей ковариационной матрице и позволить каждой условной плотности $p(\mathbf{x} | \mathcal{C}_k)$ иметь собственную ковариационную матрицу $\boldsymbol{\Sigma}_k$, то прежние исключения больше не возникнут, и в результате будут получены квадратичные функции от \mathbf{x} , что приведет к *квадратичному дискриминанту* (*quadratic discriminant*). Границы линейного и квадратичного решений показаны на рис. 5.14. На левом графике показаны распределения плотностей по классам для трех классов с гауссовым распределением (красный, зеленый и синий цвета), при этом красный и синий классы имеют одинаковую ковариационную матрицу. На правом графике показаны соответствующие апостериорные вероятности, где каждая точка на изображении окрашена с помощью пропорций красного, синего и зеленого цветов, соответствующих апостериорным вероятностям для соответствующих трех классов. Также показаны границы принятия решений. Обратите внимание, что граница между красным и синим классами, которые имеют одинаковую ковариационную матрицу, является линейной, в то время как границы между другими парами классов являются квадратичными.

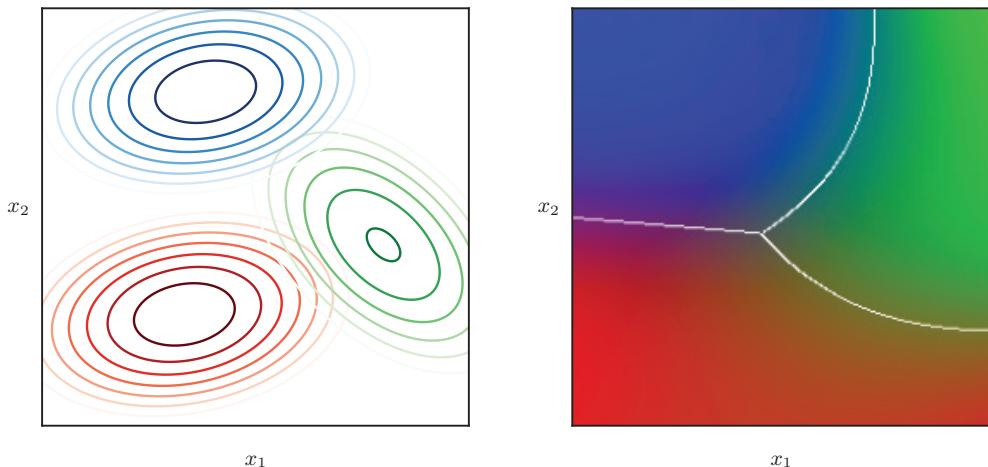


РИС. 5.14 Границы линейного и квадратичного решений

5.3.2. Решение методом максимального правдоподобия

После определения параметрической функциональной формы для распределения плотности по классам $p(\mathbf{x} | \mathcal{C}_k)$ можно определить значения параметров вместе с априорными вероятностями классов $p(\mathcal{C}_k)$ с помощью метода максимального правдоподобия. Для этого необходим набор данных, состоящий из наблюдений \mathbf{x} и соответствующих им меток классов.

Для начала допустим, что имеется два класса, каждый из которых имеет гауссово распределение плотности по классам с общей ковариационной матрицей, и набор данных $\{\mathbf{x}_n, t_n\}$, где $n = 1, \dots, N$. Здесь $t_n = 1$ обозначает класс \mathcal{C}_1 , а $t_n = 0$ обозначает класс \mathcal{C}_2 .

Обозначим априорную вероятность класса $p(\mathcal{C}_1) = \pi$, так что $p(\mathcal{C}_2) = 1 - \pi$. Для точки данных \mathbf{x}_n из класса \mathcal{C}_1 задано $t_n = 1$ и, следовательно,

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n | \mathcal{C}_1) = \pi \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}).$$

Аналогично для класса \mathcal{C}_2 задано $t_n = 0$ и, следовательно,

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n | \mathcal{C}_2) = (1 - \pi) \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_2, \boldsymbol{\Sigma}).$$

Таким образом, функция правдоподобия имеет вид:

$$p(\mathbf{t}, \mathbf{X} | \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^N [\pi \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1 - \pi) \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}, \quad (5.54)$$

где $\mathbf{t} = (t_1, \dots, t_N)^T$. Как обычно, для максимизации функции правдоподобия удобно использовать логарифм. Сначала рассмотрим максимизацию отно-

сительно π . Члены логарифмической функции правдоподобия, зависящие от π , имеют вид:

$$\sum_{n=1}^N \{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)\}. \quad (5.55)$$

Установливая производную по π равной нулю и производя перестановку, получаем

$$\pi = \frac{1}{N} \sum_{n=1}^N t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}, \quad (5.56)$$

где N_1 обозначает общее количество точек данных в классе \mathcal{C}_1 , а N_2 – общее количество точек данных в классе \mathcal{C}_2 . Таким образом, оценка максимального правдоподобия для π – это просто доля точек в классе \mathcal{C}_1 , как и ожидалось. Этот результат удобно обобщить на случай нескольких классов, где оценка максимального правдоподобия для априорной вероятности, связанной с классом \mathcal{C}_k , опять же определяется долей точек обучающего набора, отнесенных к этому классу (см. упражнение 5.13).

Теперь рассмотрим задачу максимизации относительно $\boldsymbol{\mu}_1$. И снова из функции логарифмического правдоподобия можно отобрать те члены, которые зависят от $\boldsymbol{\mu}_1$:

$$\sum_{n=1}^N t_n \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_{n=1}^N t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const.} \quad (5.57)$$

Обращая производную по $\boldsymbol{\mu}_1$ в ноль и производя перестановку, получаем

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^N t_n \mathbf{x}_n, \quad (5.58)$$

что является просто средним значением всех входных векторов \mathbf{x}_n , отнесенных к классу \mathcal{C}_1 . Аналогичным образом можно получить соответствующий результат для $\boldsymbol{\mu}_2$:

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^N (1 - t_n) \mathbf{x}_n, \quad (5.59)$$

что опять же представляет собой среднее значение всех входных векторов \mathbf{x}_n , отнесенных к классу \mathcal{C}_2 .

Наконец, рассмотрим решение максимального правдоподобия для общей ковариационной матрицы $\boldsymbol{\Sigma}$. Выбрав из функции логарифмического правдоподобия члены, зависящие от $\boldsymbol{\Sigma}$, получаем:

$$\begin{aligned}
& -\frac{1}{2} \sum_{n=1}^N t_n \ln |\Sigma| - \frac{1}{2} \sum_{n=1}^N t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) \\
& - \frac{1}{2} \sum_{n=1}^N (1 - t_n) \ln |\Sigma| - \frac{1}{2} \sum_{n=1}^N (1 - t_n) (\mathbf{x}_n - \boldsymbol{\mu}_2)^T \Sigma^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2) \\
& = -\frac{N}{2} \ln |\Sigma| - \frac{N}{2} \text{Tr}\{\Sigma^{-1} \mathbf{S}\},
\end{aligned} \tag{5.60}$$

где определено

$$\mathbf{S} = \frac{N_1}{N} \mathbf{S}_1 + \frac{N_2}{N} \mathbf{S}_2, \tag{5.61}$$

$$\mathbf{S}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T, \tag{5.62}$$

$$\mathbf{S}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T. \tag{5.63}$$

Используя стандартный результат определения максимального правдоподобия для гауссова распределения, получаем $\Sigma = \mathbf{S}$, что представляет собой средневзвешенное значение ковариационных матриц, связанных с каждым из двух классов по отдельности.

Этот результат несложно распространить на задачу о K -классах для получения соответствующих решений максимального правдоподобия по параметрам, в которых условная плотность каждого класса является гауссовой с общей ковариационной матрицей (см. упражнение 5.14). Обратите внимание, что подход, при котором классам соответствуют гаусссы распределения, не является достаточно устойчивым к отклонениям, поскольку оценка максимального правдоподобия для гауссова распределения не является устойчивой (см. раздел 5.1.4).

5.3.3. Дискретные параметры

Теперь рассмотрим дискретные значения параметров x_i . Для упрощения сначала рассмотрим двоичные значения параметров $x_i \in \{0, 1\}$ и далее перейдем к более общим дискретным параметрам. Если имеется D входов, то общее распределение будет соответствовать таблице из 2^D чисел для каждого класса и иметь $2^D - 1$ независимых переменных (из-за ограничения на суммирование). Поскольку с увеличением числа параметров рост происходит экспоненциально, следует найти более ограниченное отображение. Здесь будет использовано наивное предположение Байеса (см. раздел 11.2.3), в котором значения признаков рассматриваются в качестве независимых и обусловленных классом \mathcal{C}_k . Таким образом, получаются распределения, обусловленные классом, в виде

$$p(\mathbf{x} | \mathcal{C}_k) = \prod_{i=1}^D \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i}, \quad (5.64)$$

которые содержат D независимых параметров для каждого класса. Подстановка в (5.46) дает

$$a_k(\mathbf{x}) = \sum_{i=1}^D \{x_i \ln \mu_{ki} + (1 - x_i) \ln (1 - \mu_{ki})\} + \ln p(\mathcal{C}_k), \quad (5.65)$$

что опять же является линейными функциями входных значений x_i . Для $K = 2$ классов можно также в качестве альтернативы рассмотреть логистическую сигмоидную формулу, представленную в (5.40). Аналогичные результаты (см. упражнение 5.16) получаются для дискретных переменных, которые принимают $L > 2$ состояний.

5.3.4. Экспоненциальное семейство

Как уже было отмечено, в случае как гауссова распределения, так и дискретных входов апостериорные вероятности классов задаются обобщенными линейными моделями с логистическими сигмоидными функциями активации (класса $K = 2$) или softmax-функциями активации (класса $K > 2$). Это частные случаи более общего результата, полученного в предположении, что условные плотности $p(\mathbf{x} | \mathcal{C}_k)$ являются членами подмножества экспоненциального семейства распределений (см. раздел 3.4), заданного как

$$p(\mathbf{x} | \boldsymbol{\lambda}_k, s) = \frac{1}{s} h\left(\frac{1}{s} \mathbf{x}\right) g(\boldsymbol{\lambda}_k) \exp\left\{\frac{1}{s} \boldsymbol{\lambda}_k^T \mathbf{x}\right\}. \quad (5.66)$$

Здесь масштабный параметр s является общим для всех классов.

Для задачи с двумя классами подставим это выражение для плотности распределения по классам в (5.41) и убедимся, что апостериорная вероятность класса по-прежнему определяется логистической сигмоидной функцией для линейной функции $a(\mathbf{x})$, которая задается как

$$a(\mathbf{x}) = (\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)^T \mathbf{x} + \ln g(\boldsymbol{\lambda}_1) - \ln g(\boldsymbol{\lambda}_2) + \ln p(\mathcal{C}_1) - \ln p(\mathcal{C}_2). \quad (5.67)$$

Точно так же для задачи с K -классами подставим выражение для плотности распределения по классам в (5.46), чтобы получить

$$a_k(\mathbf{x}) = \boldsymbol{\lambda}_k^T \mathbf{x} + \ln g(\boldsymbol{\lambda}_k) + \ln p(\mathcal{C}_k), \quad (5.68)$$

и в этом случае она также является линейной функцией от \mathbf{x} .

5.4. Дискриминационные классификаторы

Для задачи классификации с двумя классами, как уже выяснилось, апостериорная вероятность класса \mathcal{C}_1 может быть записана в виде логистической сигмоидной формы для линейной функции \mathbf{x} при широком выборе распределений по классам $p(\mathbf{x} | \mathcal{C}_k)$ экспоненциального семейства. Точно так же для случая с множеством классов апостериорная вероятность класса \mathcal{C}_k задается преобразованием softmax для линейных функций \mathbf{x} . Для конкретных вариантов распределения плотностей по классам $p(\mathbf{x} | \mathcal{C}_k)$ используется максимальное правдоподобие с определением параметров плотностей, а также априорных вероятностей классов $p(\mathcal{C}_k)$, далее с помощью теоремы Байеса находятся апостериорные вероятности классов. Такая модель представляет собой пример генеративного моделирования, поскольку с ее помощью можно сгенерировать синтетические данные, взяв значения \mathbf{x} из маргинального распределения $p(\mathbf{x})$ или из любого из распределений плотности по классам $p(\mathbf{x} | \mathcal{C}_k)$.

В качестве альтернативы можно использовать функциональную форму обобщенной линейной модели в явном виде и определять ее параметры непосредственно с помощью метода максимального правдоподобия. При таком непосредственном подходе выполняется максимизация функции правдоподобия, определяемой через условное распределение $p(\mathcal{C}_k | \mathbf{x})$, что представляет собой одну из форм *дискриминативного* вероятностного моделирования. Одним из преимуществ дискриминативного подхода, как будет показано далее, заключается в сокращении числа обучаемых параметров, которые необходимо определить. Кроме того, это может привести к улучшению эффективности прогнозирования, особенно если предполагаемые формы распределения плотностей по классам представляют собой слабое приближение к истинным распределениям.

5.4.1. Функции активации

Для линейной регрессии (см. главу 4) предсказание модели $y(\mathbf{x}, \mathbf{w})$ задается линейной функцией параметров

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x} + w_0, \quad (5.69)$$

что позволяет получить непрерывное выходное значение в диапазоне $(-\infty, \infty)$. Однако для задач классификации необходимо предсказывать дискретные метки классов, или, в более общем случае, апостериорные вероятности, лежащие в диапазоне $(0, 1)$. В этом случае необходимо рассматривать обобщение этой модели, где линейная функция \mathbf{w} и w_0 преобразуются с помощью нелинейной функции $f(\cdot)$ следующим образом:

$$y(\mathbf{x}, \mathbf{w}) = f(\mathbf{w}^T \mathbf{x} + w_0). \quad (5.70)$$

В литературе по машинному обучению $f(\cdot)$ известна как *функция активации* (*activation function*), в то время как в литературе по статистике ее обратная величина называется *функцией связи* (*link function*). Поверхности принятия решений соответствуют константе: $y(x) = \text{constant}$, так что $w^T x = \text{constant}$, и, следовательно, поверхности принятия решений являются линейными функциями от x , даже если функция $f(\cdot)$ нелинейна. По этой причине класс моделей, характеризуемых в (5.70), называют *обобщенными линейными моделями* (*generalized linear models*) (McCullagh and Nelder, 1989). Но, в отличие от моделей, используемых для регрессии, такие модели уже не являются линейными в части параметров из-за нелинейной функции $f(\cdot)$. В результате аналитические и вычислительные свойства этих моделей будут сложнее, чем у моделей линейной регрессии. Тем не менее эти модели все еще относительно просты по сравнению с гораздо более гибкими нелинейными моделями, которые будут рассмотрены в последующих главах.

5.4.2. Фиксированные базисные функции

До сих пор в этой главе рассматривались модели классификации, работающие непосредственно с исходным входным вектором x . В то же время если сначала произвести фиксированное нелинейное преобразование входных данных с помощью вектора базисных функций $\varphi(x)$, то в этом случае будут одинаково эффективны все алгоритмы. Полученные границы принятия решений будут линейными в пространстве признаков φ , и они будут соответствовать нелинейным границам принятия решений в исходном пространстве x , как показано на рис. 5.15. На левом графике показано исходное входное пространство (x_1, x_2) с точками данных двух классов, обозначенных красным и синим. В этом пространстве определены две «гауссовые» базисные функции $\varphi_1(x)$ и $\varphi_2(x)$, центры которых показаны зелеными крестиками,

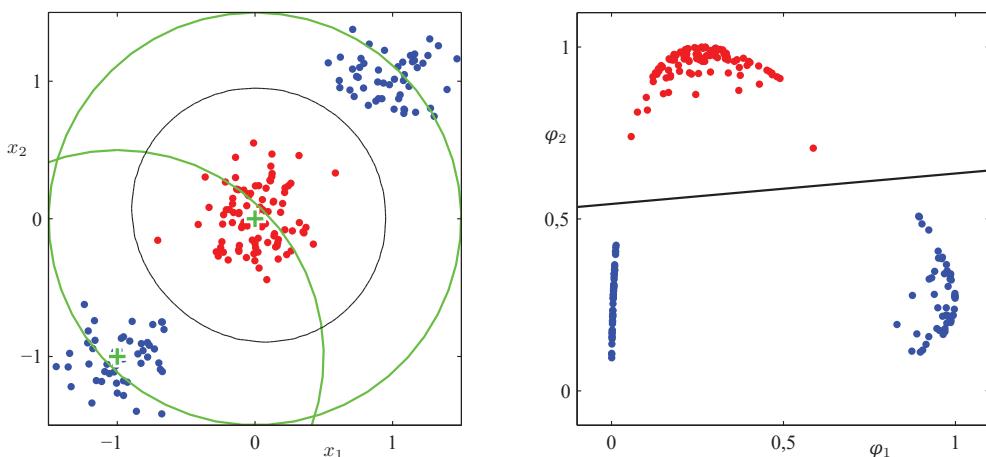


РИС. 5.15 Иллюстрация роли нелинейных базисных функций в линейных моделях классификации

а контуры – зелеными кружками. На правом графике показано соответствующее пространство признаков (φ_1, φ_2) вместе с линейной границей принятия решения, полученной с помощью модели логистической регрессии в форме (см. раздел 5.4.3). Это соответствует нелинейной границе принятия решения в исходном пространстве входных данных, показанной черной кривой на левом графике. Классы, линейно разделяемые в пространстве признаков $\varphi(\mathbf{x})$, не обязательно должны быть линейно разделяемыми в исходном пространстве наблюдений \mathbf{x} .

Обратите внимание, что, как и при анализе линейных моделей регрессии, одна из базисных функций обычно задается постоянной, например $\varphi_0(\mathbf{x}) = 1$, так что соответствующий параметр w_0 играет роль смещения.

Во многих практических задачах в пространстве \mathbf{x} между классами проходит значительное перекрытие условных плотностей $p(\mathbf{x} | \mathcal{C}_k)$. Это соответствует апостериорным вероятностям $p(\mathcal{C}_k | \mathbf{x})$, которые, по крайней мере для некоторых значений \mathbf{x} , не равны 0 или 1. В таких случаях оптимальное решение достигается посредством точного моделирования апостериорных вероятностей и последующего применения стандартной теории принятия решений (см. раздел 5.2). Обратите внимание, что нелинейные преобразования $\varphi(\mathbf{x})$ не могут устраниТЬ такое перекрытие классов, хотя они могут повысить уровень перекрытия или создать перекрытие там, где его не было в исходном пространстве наблюдений. Вместе с тем правильный выбор нелинейности может облегчить процесс моделирования апостериорных вероятностей. Однако такие модели с фиксированными базисными функциями имеют существенные ограничения (см. раздел 6.1), и в последующих главах вопрос об устранении этих ограничений будет решаться путем адаптации самих базисных функций к данным.

5.4.3. Логистическая регрессия

Для начала рассмотрим задачу классификации по двум классам. При обсуждении генеративных подходов в разделе 5.3 было отмечено, что при достаточно широких допущениях апостериорная вероятность класса \mathcal{C}_1 может быть записана в виде логистической сигмоидной функции по линейной функции вектора признаков φ так, что

$$p(\mathcal{C}_1 | \varphi) = y(\varphi) = \sigma(\mathbf{w}^T \varphi), \quad (5.71)$$

при этом $p(\mathcal{C}_2 | \varphi) = 1 - p(\mathcal{C}_1 | \varphi)$. Здесь $\sigma(\cdot)$ – это логистическая сигмоидная функция, определенная в (5.42). В терминологии статистики эта модель известна как логистическая регрессия (*logistic regression*), однако следует отметить, что речь идет о модели для классификации, а не для непрерывной переменной.

Для M -мерного пространства признаков φ эта модель обладает M настраиваемыми параметрами. Для сравнения: если использовать метод максимального правдоподобия для гауссовых распределений плотностей по классам, то потребовалось бы $2M$ параметров для средних значений и $M(M + 1)/2$ параметров для (совместной) ковариационной матрицы. В сочетании с априор-

ным распределением класса $p(\mathcal{C}_1)$ это дает в общей сложности $M(M + 5)/2 + 1$ параметров, которые растут квадратично с увеличением M , в отличие от линейной зависимости числа параметров от M в логистической регрессии. При больших значениях M есть явное преимущество в работе с моделью логистической регрессии напрямую.

Теперь воспользуемся методом максимального правдоподобия для определения параметров модели логистической регрессии. Для этого возьмем производную логистической сигмоидной функции, которую удобно выразить в виде самой сигмоидной функции (см. упражнение 5.18):

$$\frac{d\sigma}{da} = \sigma(1 - \sigma). \quad (5.72)$$

Для набора данных $\{\boldsymbol{\varphi}_n, t_n\}$, где $\boldsymbol{\varphi}_n = \boldsymbol{\varphi}(\mathbf{x}_n)$ и $t_n \in \{0, 1\}$, при $n = 1, \dots, N$, функция правдоподобия может быть записана как

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n}, \quad (5.73)$$

где $\mathbf{t} = (t_1, \dots, t_N)^T$ и $y_n = p(\mathcal{C}_1 | \boldsymbol{\varphi}_n)$. Как обычно, мы можем определить функцию ошибки, взяв отрицательный логарифм правдоподобия, что позволяет получить функцию ошибки *перекрестной энтропии* (*cross-entropy*):

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}, \quad (5.74)$$

где $y_n = \sigma(a_n)$ и $a_n = \mathbf{w}^T \boldsymbol{\varphi}_n$. Взяв градиент функции ошибки по \mathbf{w} (см. упражнение 5.19), получим

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \boldsymbol{\varphi}_n, \quad (5.75)$$

где использовано (5.72). Как видно, коэффициент, включающий производную логистической сигмоидной функции, был аннулирован, что привело к упрощенной форме для градиента логарифмического правдоподобия. В частности, вклад в градиент от точки данных n определяется «ошибкой» $y_n - t_n$ между целевым значением и предсказанием модели, умноженным на вектор базисных функций $\boldsymbol{\varphi}_n$. Более того, сравнение с (4.12) показывает, что она имеет точно такую же форму, как и градиент функции ошибки суммы квадратов для модели линейной регрессии (см. раздел 4.1.3).

Решение максимального правдоподобия равно $\nabla E(\mathbf{w}) = 0$. Однако из (5.75) видно, что оно уже не соответствует набору линейных уравнений из-за нелинейности $y(\cdot)$, и поэтому у этого уравнения нет решения в замкнутой форме. Одним из подходов к поиску решения максимального правдоподобия может быть использование стохастического градиентного спуска (см. главу 7), в котором ∇E_n – это n -й член в правой части (5.75). Стохастический градиентный спуск является основным подходом к обучению сильно нелинейных нейрон-

ных сетей, о которых пойдет речь в последующих главах. Однако уравнение максимального правдоподобия лишь «слегка» нелинейно, и на самом деле функция ошибки (5.74) для определения модели в (5.71) является выпуклой функцией параметров, что позволяет минимизировать функцию ошибки с помощью простого алгоритма *итеративного пересчета наименьших квадратов* (*iterative reweighted least squares*, или *IRLS*) (Bishop, 2006). Однако такой подход непросто обобщить применительно к более сложным моделям, таким как глубокие нейронные сети.

Обратите внимание, что максимальное правдоподобие может демонстрировать значительную избыточную подгонку для линейно разделяемых наборов данных. Это происходит вследствие того, что решение с максимальным правдоподобием имеет место при разделении двух классов гиперплоскостью со значением $\sigma = 0,5$, что соответствует $\mathbf{w}^T \boldsymbol{\varphi} = 0$, и величина \mathbf{w} стремится к бесконечности. В этом случае логистическая сигмоидная функция становится бесконечно крутой в пространстве признаков, что соответствует ступенчатой функции Хевисайда, так что каждой обучающей точке из каждого класса k присваивается апостериорная вероятность $p(\mathcal{C}_k | \mathbf{x}) = 1$ (см. упражнение 5.20). Более того, обычно существует множество таких решений, поскольку любая разделяющая гиперплоскость приводит к одинаковым апостериорным вероятностям в точках обучающих данных. Максимальное правдоподобие не дает возможности отдать предпочтение одному такому решению перед другим, и на практике найденное решение будет зависеть от выбора алгоритма оптимизации и инициализации параметров. Следует отметить, что проблема возникнет даже при большом количестве точек данных по сравнению с количеством параметров в модели при условии, что набор обучающих данных линейно разделим. Сингулярности можно избежать добавлением регуляризующего члена в функцию ошибки (см. главу 9).

5.4.4. Логистическая регрессия для нескольких классов

При изучении генеративных моделей для классификации по нескольким классам (см. раздел 5.3) было отмечено, что для большого класса распределений из экспоненциального семейства апостериорные вероятности задаются преобразованием линейных функций признаковых переменных по методу softmax, так что

$$p(\mathcal{C}_k | \boldsymbol{\varphi}) = y_k(\boldsymbol{\varphi}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \quad (5.76)$$

где предварительные активации a_k задаются как

$$a_k = \mathbf{w}_k^T \boldsymbol{\varphi}. \quad (5.77)$$

Здесь максимальное правдоподобие используется для раздельного определения распределения плотностей по классам и априорных вероятностей классов, и затем с помощью теоремы Байеса находятся соответствующие

апостериорные вероятности, тем самым определяя в неявном виде параметры $\{\mathbf{w}_k\}$. Далее речь пойдет об использовании метода максимального правдоподобия для непосредственного определения параметров $\{\mathbf{w}_k\}$ этой модели. Для этого потребуются производные y_k по всем предварительным активациям a_j (см. упражнение 5.21). Они определяются как

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j), \quad (5.78)$$

где I_{kj} – это элементы единичной матрицы.

Далее запишем функцию правдоподобия. Это проще всего сделать с использованием схемы кодирования «1 из K », где целевой вектор \mathbf{t}_n для вектора признаков $\boldsymbol{\varphi}_n$, принадлежащего классу \mathcal{C}_k , представляет собой двоичный вектор со всеми нулевыми элементами, кроме элемента k , который равен единице. Тогда функция правдоподобия задается как

$$p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^N \prod_{k=1}^K p(\mathcal{C}_k | \boldsymbol{\varphi}_n)^{t_{nk}} = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}}, \quad (5.79)$$

где $y_{nk} = y_k(\boldsymbol{\varphi}_n)$, а \mathbf{T} – это матрица $N \times K$ целевых переменных с элементами t_{nk} . Взятие отрицательного логарифма дает

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}, \quad (5.80)$$

что называют функцией ошибки *перекрестной энтропии* (*cross-entropy*) для задач классификации по нескольким классам.

Теперь возьмем градиент функции ошибки относительно одного из векторов параметров \mathbf{w}_j . Используя результат (5.78) для производных функции softmax (см. упражнение 5.22), получим

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{n=1}^N (y_{nk} - t_{nj}) \varphi_n, \quad (5.81)$$

где используется $\sum_k t_{nk} = 1$. И вновь можно оптимизировать параметры с помощью стохастического градиентного спуска (см. главу 7).

И снова наблюдается та же форма градиента, что и для функции ошибки по сумме квадратов в линейной модели и для ошибки перекрестной энтропии в модели логистической регрессии, а именно произведение ошибки $(y_{nj} - t_{nj})$ на активацию базисной функции φ_n . Это примеры более общего результата, который будет рассмотрен позже (см. раздел 5.4.6).

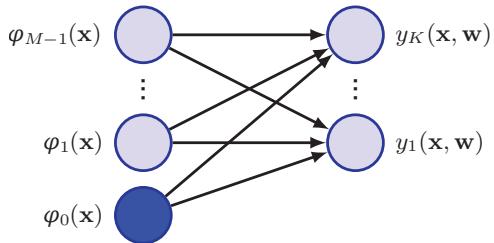
Линейные модели классификации можно представить в виде однослойных нейронных сетей, как показано на рис. 5.16. Здесь каждая базисная функция представлена узлом, при этом сплошной узел представляет «смещенную» базисную функцию φ_0 , в то время как каждый выход y_1, \dots, y_N также представлен узлом. Связи между узлами представляют собой соответствующие параметры веса и смещения. Если рассматривать производную функции

ошибки по весу w_{ik} , который связывает базисную функцию $\varphi_i(\mathbf{x})$ с выходной величиной t_k , то из (5.81) следует, что

$$\frac{\partial E(\mathbf{w}_1, \dots, \mathbf{w}_K)}{\partial w_{ij}} = \sum_{n=1}^N (y_{nk} - t_{nk})\varphi_i(\mathbf{x}_n). \quad (5.82)$$

Сравнивая это с рис. 5.16, можно убедиться, что для каждой точки данных n этот градиент имеет вид выхода базисной функции на входе весовой связи с «ошибкой» ($y_{nk} - t_{nk}$) на выходе.

РИС. 5.16 Представление модели линейной классификации для нескольких классов в виде нейронной сети с одним слоем связей



5.4.5. Пробит-регрессия

Ранее было отмечено, что для широкого диапазона распределения по классам, описываемого экспоненциальным семейством, результирующие апостериорные вероятности классов задаются логистическим (или softmax) преобразованием по линейной функции переменных признаков. Однако не все варианты распределения плотности по классам приводят к такой простой форме для апостериорных вероятностей, и это повод для изучения других типов дискриминативных вероятностных моделей. Рассмотрим случай для двух классов, снова оставаясь в рамках обобщенных линейных моделей, так что

$$p(t = 1 | a) = f(a), \quad (5.83)$$

где $a = \mathbf{w}^T \boldsymbol{\varphi}$, а $f(\cdot)$ – это функция активации.

Одним из способов аргументировать альтернативный выбор функции связи может быть использование пороговой модели шума, которая выглядит следующим образом. Для каждого входа φ_n производится оценка $a_n = \mathbf{w}^T \boldsymbol{\varphi}_n$, и затем устанавливается целевое значение в соответствии с

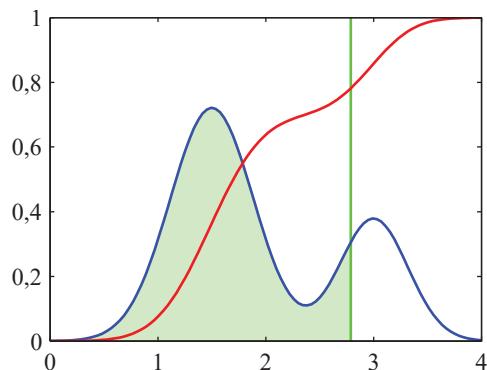
$$\begin{cases} t_n = 1, & \text{если } a_n \geq \theta, \\ t_n = 0 & \text{в ином случае.} \end{cases} \quad (5.84)$$

Если значение θ берется из плотности вероятности $p(\theta)$, то соответствующая функция активации будет задана кумулятивной функцией распределения:

$$f(a) = \int_{-\infty}^a p(\theta) d\theta, \quad (5.85)$$

как показано на рис. 5.17. Обратите внимание, что значение синей кривой в любой точке, например обозначенной вертикальной зеленой линией, соответствует наклону красной кривой в той же точке. И наоборот, значение красной кривой в этой точке соответствует площади под синей кривой, обозначенной заштрихованной зеленой областью. В стохастической пороговой модели метка класса принимает значение $t = 1$, если значение $a = \mathbf{w}^T \varphi$ превышает порог, в противном случае она принимает значение $t = 0$. Это эквивалентно функции активации, заданной кумулятивной функцией распределения $f(a)$.

РИС. 5.17 Схематический пример плотности вероятности $p(\theta)$ (синяя кривая), которая в этом примере задана смесью двух гауссовых распределений, и ее кумулятивной функции распределения $f(a)$ (красная кривая)



В качестве конкретного примера предположим, что плотность $p(\theta)$ задается гауссовым распределением с нулевым средним значением и единичной дисперсией. Соответствующая кумулятивная функция распределения имеет вид

$$\Phi(a) = \int_{-\infty}^a \mathcal{N}(\theta | 0, 1) d\theta, \quad (5.86)$$

и она известна как функция пробита (*probit function*). Она имеет сигмоидную форму и сравнивается с логистической сигмоидной функцией на рис. 5.12. Обратите внимание, что использование гауссова распределения с общими средним значением и дисперсией не меняет модель, поскольку это эквивалентно перемасштабированию линейных коэффициентов \mathbf{w} . Многие числовые пакеты могут оценивать близкую функцию, определяемую как

$$\text{erf}(a) = \frac{2}{\sqrt{\pi}} \int_0^a \exp(-\theta^2/2) d\theta, \quad (5.87)$$

которую называют функцией *erf* (*erf function*) или функцией ошибки (*error function*). Не путать с функцией ошибки модели машинного обучения (см. упражнение 5.23). Она связана с функцией пробита следующим образом:

$$\Phi(a) = \frac{1}{2} \left\{ 1 + \frac{1}{\sqrt{2}} \text{erf}(a) \right\}. \quad (5.88)$$

Обобщенная линейная модель, основанная на функции активации пробита, известна как *пробит-регрессия (probit regression)*. Параметры этой модели можно определить с помощью метода максимального правдоподобия, что является простым продолжением рассмотренных ранее концепций. На практике результаты, полученные с помощью пробит-регрессии, похожи на результаты логистической регрессии.

Одна из проблем, которая может возникнуть в практических приложениях, – это *выбросы (outliers)*, которые могут появиться, например, из-за ошибок в измерении входного вектора x или из-за неправильной маркировки целевого значения t . Поскольку такие точки могут лежать далеко за пределами идеальной границы принятия решения, они могут серьезно искажить результаты классификатора. Модели логистической и пробит-регрессии в этом плане ведут себя по-разному, поскольку хвосты логистической сигмоидной функции асимптотически затухают как $\exp(-x)$ для $|x| \rightarrow \infty$, тогда как для функции активации пробита они затухают как $\exp(-x^2)$, и поэтому пробит-модель может быть значительно более чувствительна к выбросам.

5.4.6. Канонические функции связей

Для модели линейной регрессии с гауссовым распределением шума функция ошибки, соответствующая отрицательному логарифму правдоподобия, задается в (4.11). Если взять производную по вектору параметров w от вклада в функцию ошибки для точки данных n , то она примет вид «ошибки» $y_n - t_n$, умноженной на вектор признаков φ_n , где $y_n = w^T \varphi_n$. Точно так же для комбинации функции активации логистической сигмоидной функции, функции ошибки перекрестной энтропии (5.74) и для функции активации softmax с функцией ошибки перекрестной энтропии для нескольких классов (5.80) вновь получается такая же простая форма. Теперь приведем общий результат предположения об условном распределении целевой переменной из экспоненциального семейства и соответствующего выбора функции активации, известной как *каноническая функция связи (canonical link function)*.

Здесь вновь используется ограниченная форма (3.169) распределений экспоненциального семейства. Обратите внимание, что в данном случае допущение о распределении экспоненциального семейства применяется к целевой переменной t , в отличие от раздела 5.3.4, где это допущение применялось к входному вектору x . Поэтому рассмотрим условные распределения целевой переменной вида

$$p(t|\eta, s) = \frac{1}{s} h\left(\frac{t}{s}\right) g(\eta) \exp\left\{-\frac{\eta t}{s}\right\}. \quad (5.89)$$

Используя ту же линию аргументации, которая привела к выводу результата (3.172), получаем, что условное среднее значение t , которое обозначим через y , задается как

$$y \equiv \mathbb{E}[t|\eta] = -s \frac{d}{d\eta} \ln g(\eta). \quad (5.90)$$

Таким образом, y и η необходимо связать, и эту связь можно обозначить через $\eta = \psi(y)$.

Следуя рекомендациям публикации (Nelder and Wedderburn, 1972), определим *обобщенную линейную модель* (*generalized linear model*) как такую, для которой y является нелинейной функцией линейной комбинации входных (или признаковых) переменных, так что

$$y = f(\mathbf{w}^T \boldsymbol{\varphi}), \quad (5.91)$$

где $f(\cdot)$ в литературе по машинному обучению известна как функция активации, а $f^{-1}(\cdot)$ в статистике называется функцией связи.

Теперь для этой модели рассмотрим функцию логарифмического правдоподобия, которая в качестве функции η имеет вид

$$\ln p(\mathbf{t}|\eta, s) = \sum_{n=1}^N \ln p(t_n|\eta, s) = \sum_{n=1}^N \left\{ \ln g(\eta_n) + \frac{\eta_n t_n}{s} \right\} + \text{const}, \quad (5.92)$$

где подразумевается, что все наблюдения имеют общий масштабный параметр (который соответствует дисперсии шума, например для гауссова распределения), и поэтому s не зависит от n . Тогда производная логарифмического правдоподобия по параметрам модели \mathbf{w} определяется как

$$\begin{aligned} \nabla_{\mathbf{w}} \ln p(\mathbf{t}|\eta, s) &= \sum_{n=1}^N \left\{ \frac{d}{d\eta_n} \ln g(\eta_n) + \frac{t_n}{s} \right\} \frac{d\eta_n}{dy_n} \frac{dy_n}{da_n} \nabla_{\mathbf{w}} a_n \\ &= \sum_{n=1}^N \frac{1}{s} \{t_n - y_n\} \psi'(y_n) f'(a_n) \varphi_n, \end{aligned} \quad (5.93)$$

где $a_n = \mathbf{w}^T \boldsymbol{\varphi}_n$, и здесь используется $y_n = f(a_n)$ вместе с результатом (5.90) для $\mathbb{E}[t|\eta]$. Теперь становится ясно, что можно значительно упростить задачу, если выбрать определенную форму для функции связи $f^{-1}(y)$, которая задается следующим образом:

$$f^{-1}(y) = \psi(y), \quad (5.94)$$

что дает $f(\psi(y)) = y$ и, следовательно, $f'(\psi)\psi'(y) = 1$. Также, поскольку $a = f^{-1}(y)$, получается $a = \psi$ и, следовательно, $f'(a)\psi'(y) = 1$. В этом случае градиент функции ошибки сводится к

$$\nabla \ln E(\mathbf{w}) = \frac{1}{s} \sum_{n=1}^N \{y_n - t_n\} \varphi_n. \quad (5.95)$$

Как было замечено, существует естественная связь между выбором функции ошибки и выбором функции активации выходных модулей. Хотя этот результат был получен в контексте моделей однослойных сетей, те же соображения применимы и к глубоким нейронным сетям, о которых пойдет речь в последующих главах.

Упражнения

- 5.1** (*) Рассмотрим задачу классификации с K классами и целевым вектором \mathbf{t} , в которой используется схема двоичного кодирования «1 из K ». Докажите, что условное ожидание $E[\mathbf{t} | \mathbf{x}]$ задается апостериорной вероятностью $p(\mathcal{C}_k | \mathbf{x})$.
- 5.2** (**) При заданном наборе точек данных $\{\mathbf{x}_n\}$ можно определить выпуклую оболочку как набор всех точек \mathbf{x} , заданных следующим образом:

$$\mathbf{x} = \sum_n \alpha_n \mathbf{x}_n, \quad (5.96)$$

где $\alpha_n \geq 0$ и $\sum_n \alpha_n = 1$. Рассмотрим второе множество точек $\{\mathbf{y}_n\}$ вместе с соответствующей им выпуклой оболочкой. По определению эти два множества точек будут линейно разделимы, если существуют такие вектор $\hat{\mathbf{w}}$ и скаляр w_0 , что $\hat{\mathbf{w}}^T \mathbf{x}_n + w_0 > 0$ для всех \mathbf{x}_n и $\hat{\mathbf{w}}^T \mathbf{y}_n + w_0 < 0$ для всех \mathbf{y}_n . Докажите, что если их выпуклые оболочки пересекаются, то эти два набора точек не могут быть линейно разделяемы; и наоборот, если они линейно разделяемы, то их выпуклые оболочки не пересекаются.

- 5.3** (**) Рассмотрим минимизацию функции ошибки суммы квадратов (5.14) и предположим, что все целевые векторы в обучающем множестве удовлетворяют линейному ограничению

$$\mathbf{a}^T \mathbf{t}_n + b = 0, \quad (5.97)$$

где \mathbf{t}_n соответствует n -й строке матрицы \mathbf{T} в (5.14). Докажите, что, как следствие этого ограничения, элементы модельного предсказания $\mathbf{y}(\mathbf{x})$, заданного решением по методу наименьших квадратов (5.16), также удовлетворяют этому ограничению, так что

$$\mathbf{a}^T \mathbf{y}(\mathbf{x}) + b = 0. \quad (5.98)$$

Для этого предположим, что одна из базисных функций $\varphi_0(\mathbf{x}) = 1$, так что соответствующий параметр w_0 играет роль смещения.

- 5.4** (**) Расширьте результат упражнения 5.3, чтобы доказать, что если целевые векторы одновременно удовлетворяют нескольким линейным ограничениям, то эти же ограничения будут удовлетворять и предсказанию по методу наименьших квадратов линейной модели.
- 5.5** (*) Используйте определение (5.38), а также (5.30) и (5.31), чтобы вывести результат (5.39) для F-оценки.
- 5.6** (**) Пусть a и b – это два неотрицательных числа. Докажите, что если $a \leq b$, то $a \leq (ab)^{1/2}$. Используйте этот результат для доказательства того, что если области принятия решений в задаче классификации двух клас-

сов выбраны таким образом, чтобы минимизировать вероятность ошибочной классификации, то эта вероятность будет удовлетворять условию

$$p(\text{ошибки}) \leq \int \{p(\mathbf{x}, \mathcal{C}_1)p(\mathbf{x}, \mathcal{C}_2)\}^{1/2} d\mathbf{x}. \quad (5.99)$$

- 5.7** (*) С учетом матрицы потерь с элементами L_{kj} , ожидаемый риск минимизируется, если для каждого \mathbf{x} выбрать класс, который минимизирует (5.23). Подтвердите, что, когда матрица потерь задана как $L_{kj} = 1 - I_{kj}$, где I_{kj} – это элементы единичной матрицы, это сводится к критерию выбора класса с наибольшей апостериорной вероятностью. Какова интерпретация этой формы матрицы потерь?
- 5.8** (*) Выполните критерий минимизации ожидаемых потерь при наличии общей матрицы потерь и общих априорных вероятностей для классов.
- 5.9** (*) Пусть среднее значение апостериорных вероятностей по набору из N точек данных имеет вид:

$$\frac{1}{N} \sum_{n=1}^N p(\mathcal{C}_k | \mathbf{x}_n). \quad (5.100)$$

Используя предел $N \rightarrow \infty$, докажите, что эта величина приближается к априорной вероятности класса $p(\mathcal{C}_k)$.

- 5.10** (**) Рассмотрим задачу классификации, в которой потери, понесенные при классификации входного вектора из класса \mathcal{C}_k как принадлежащего классу \mathcal{C}_j , задаются матрицей потерь L_{kj} и для которой потери, понесенные при выборе опции отказа, равны λ . Найдите критерий принятия решения, который даст минимальные ожидаемые потери. Докажите, что он сводится к критерию отказа, рассмотренному в разделе 5.2.3, когда матрица потерь задана $L_{kj} = 1 - I_{kj}$. Какова связь между λ и порогом отклонения θ ?
- 5.11** (*) Докажите, что логистическая сигмоидная функция (5.42) удовлетворяет свойству $\sigma(-a) = 1 - \sigma(a)$ и что ее обратная величина задается как $\sigma^{-1}(y) = \ln \{y/(1-y)\}$.
- 5.12** (*) Используя (5.40) и (5.41), выведите результат (5.48) для апостериорной вероятности класса в генеративной модели с двумя классами и гауссовыми распределениями плотности, а также проверьте результаты (5.49) и (5.50) для параметров \mathbf{w} и w_0 .
- 5.13** (*) Рассмотрим генеративную модель классификации для K классов, определяемую априорными вероятностями классов $p(\mathcal{C}_k) = \pi_k$ и общими условными плотностями классов $p(\boldsymbol{\varphi} | \mathcal{C}_k)$, где $\boldsymbol{\varphi}$ – это вектор входных признаков. Пусть задан обучающий набор данных $\{\boldsymbol{\varphi}_n, \mathbf{t}_n\}$, где $n = 1, \dots, N$, а \mathbf{t}_n – это двоичный целевой вектор длины K , использующий схему кодирования «1 из K », так что он имеет компоненты $t_{nj} = I_{jk}$, если точка

данных n относится к классу \mathcal{C}_k . Допуская, что точки данных берутся из этой модели независимо друг от друга, докажите, что решение максимального правдоподобия для априорных вероятностей имеет вид:

$$\pi_k = \frac{N_k}{N}, \quad (5.101)$$

где N_k – это количество точек данных, отнесенных к классу \mathcal{C}_k .

- 5.14** (**) Рассмотрим модель классификации из упражнения 5.13 и теперь предположим, что условные плотности классов заданы гауссовыми распределениями с общей ковариационной матрицей, так что

$$p(\boldsymbol{\varphi} | \mathcal{C}_k) = \mathcal{N}(\boldsymbol{\varphi} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}). \quad (5.102)$$

Докажите, что решение максимального правдоподобия для среднего значения гауссова распределения для класса \mathcal{C}_k имеет вид:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N t_{nk} \boldsymbol{\varphi}_n, \quad (5.103)$$

что представляет собой среднее значение векторов признаков, отнесенных к классу \mathcal{C}_k . Аналогичным образом докажите, что решение максимального правдоподобия для общей ковариационной матрицы имеет вид:

$$\boldsymbol{\Sigma} = \sum_{k=1}^K \frac{N_k}{N} \mathbf{S}_k, \quad (5.104)$$

где

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^N t_{nk} (\boldsymbol{\varphi}_n - \boldsymbol{\mu}_k)(\boldsymbol{\varphi}_n - \boldsymbol{\mu}_k)^T. \quad (5.105)$$

Таким образом, $\boldsymbol{\Sigma}$ задается средневзвешенным значением ковариаций данных, связанных с каждым классом, где весовые коэффициенты задаются априорными вероятностями классов.

- 5.15** (**) Выполните решение максимального правдоподобия для параметров $\{\mu_{ki}\}$ вероятностного классификатора наивного Байеса с дискретными бинарными признаками, который был представлен в разделе 5.3.3.

- 5.16** (**) Рассмотрите задачу классификации с K классами, для которых вектор признаков $\boldsymbol{\varphi}$ имеет M компонент, каждая из которых может принимать L дискретных состояний. Пусть значения компонент представлены схемой двоичного кодирования «1 из L ». Далее предположим, что в зависимости от класса \mathcal{C}_k M компонент $\boldsymbol{\varphi}$ независимы, так что плотность распределения по классам факторизуется относительно компонент вектора признаков. Докажите, что величины a_k , заданные

в (5.46), входящие в аргумент функции softmax, описывающей апостериорные вероятности классов, являются линейными функциями компонент φ (см. раздел 11.2.3). Обратите внимание, что это пример наивной модели Байеса.

- 5.17** (**) Выведите решение максимального правдоподобия для параметров вероятностного классификатора наивного Байеса, описанного в упражнении 5.16.
- 5.18** (*) Убедитесь в справедливости соотношения (5.72) для производной логистической сигмоидной функции, определенной в (5.42).
- 5.19** (*) Используя результат (5.72) для производной логистической сигмоидной функции, докажите, что производная функции ошибки (5.74) для модели логистической регрессии имеет вид (5.75).
- 5.20** (*) Докажите, что для линейно разделяемого набора данных решение с максимальным правдоподобием логистической регрессионной модели можно получить при нахождении вектора w , граница принятия решения которого $w^T \varphi(x) = 0$ разделяет классы, с последующим стремлением величины w к бесконечности.
- 5.21** (*) Докажите, что производные функции активации softmax (5.76), где a_k определены в (5.77), определяются в (5.78).
- 5.22** (*) Используя результат (5.78) для производных функции активации softmax, докажите, что градиенты ошибки перекрестной энтропии (5.80) определяются в (5.81).
- 5.23** (*) Докажите, что функция пробита (5.86) и функция erf (5.87) связаны между собой (5.88).
- 5.24** (**) Допустим, требуется аппроксимировать логистическую сигмоидную функцию $\sigma(a)$, определенную в (5.42), масштабированной функцией пробита $\Phi(\lambda a)$, где $\Phi(a)$ определяется в (5.86). Докажите, что при выборе λ таким образом, чтобы производные этих двух функций были равны при $a = 0$, $\lambda^2 = \pi/8$.

Глава 6

Глубокие нейронные сети

В последние годы нейронные сети превратились в самую важную технологию машинного обучения для практических приложений, и поэтому значительная часть этой книги посвящена их изучению. В предыдущих главах уже был заложен базовый фундамент для изучения этих технологий. В частности, ранее было отмечено, что модели линейной регрессии, включающие линейные комбинации фиксированных нелинейных базисных функций, могут быть выражены в виде нейронных сетей с одним слоем весовых параметров и параметров смещения (см. главу 4).

Точно так же модели классификации на основе линейных комбинаций базисных функций можно рассматривать в качестве однослойных нейронных сетей (см. главу 5). Это позволяет ввести в этой главе несколько важных концепций, прежде чем перейти к обсуждению более сложных многослойных сетей.

При достаточном количестве правильно подобранных базисных функций такие линейные модели позволяют аппроксимировать любое нелинейное преобразование от входов к выходам с любой желаемой точностью, и поэтому может показаться, что их достаточно для решения любых практических задач. Однако эти модели имеют ряд серьезных ограничений, и поэтому знакомство с нейронными сетями начнется с изучения этих ограничений и объяснения необходимости использования базисных функций, которые сами обучаются на основе данных. Это естественным образом приводит к обсуждению нейронных сетей, имеющих более одного слоя обучаемых параметров (раздел 6.3.6). Они известны как *сети прямого распространения* (*feed-forward networks*) или *многослойные перцептроны* (*multilayer perceptrons*). Далее будут рассмотрены преимущества большого количества таких слоев обработки, что позволит сформировать ключевую концепцию глубоких нейронных сетей, доминирующих в настоящее время в области машинного обучения.

6.1. Ограничения фиксированных базисных функций

Модели линейных базисных функций для классификации (см. главу 5) основаны на линейных комбинациях базисных функций $\varphi_j(\mathbf{x})$ и имеют вид:

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^M w_j \varphi_j(\mathbf{x}) + w_0\right), \quad (6.1)$$

где $f(\cdot)$ – это нелинейная функция активации выхода. Линейные модели регрессии имеют ту же форму, но $f(\cdot)$ заменяется тождеством (см. главу 4). Эти модели допускают произвольный набор нелинейных базисных функций $\{\varphi_i(\mathbf{x})\}$, и в силу общности этих базисных функций такие модели в принципе могут обеспечить решение любой задачи регрессии или классификации. Это справедливо в том банальном смысле, что если одна из базисных функций соответствует желаемому преобразованию входа в выход, то обучаемый линейный слой просто копирует значение этой базисной функции на выход модели.

В более общем случае можно ожидать, что достаточно большой и разнообразный набор базисных функций позволит аппроксимировать любую желаемую функцию с произвольной точностью. Поэтому может показаться, что такие линейные модели представляют собой универсальный инструментарий для решения задач машинного обучения. К сожалению, у линейных моделей есть ряд существенных недостатков, которые связаны с предположением, что базисные функции $\varphi_i(\mathbf{x})$ фиксированы и не зависят от обучающих данных. Для того чтобы лучше понимать суть этих недостатков, для начала рассмотрим поведение линейных моделей при увеличении числа входных переменных.

6.1.1. Проклятие размерности

Рассмотрим простую регрессионную модель для одной входной переменной (см. раздел 1.2), заданную полиномом порядка M в форме

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M. \quad (6.2)$$

Теперь рассмотрим ситуацию с увеличением числа входов. Если имеется D входных переменных $\{x_1, \dots, x_D\}$, то общий многочлен с коэффициентами до порядка 3 будет иметь вид:

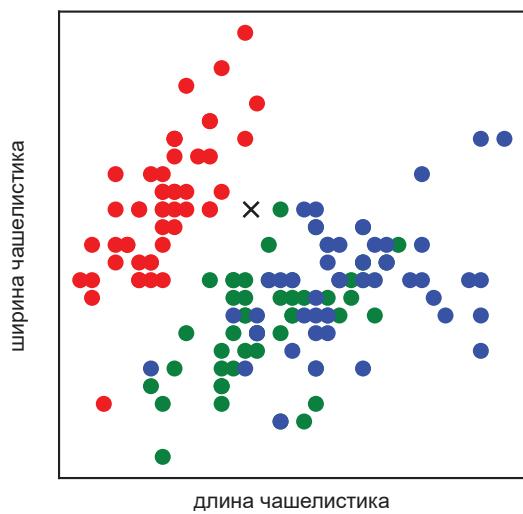
$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^D w_i x_i + \sum_{i=1}^D \sum_{j=1}^D w_{ij} x_i x_j + \sum_{i=1}^D \sum_{j=1}^D \sum_{k=1}^D w_{ijk} x_i x_j x_k. \quad (6.3)$$

С увеличением D рост числа независимых коэффициентов составит $\mathcal{O}(D^3)$, тогда как для полинома порядка M рост числа коэффициентов равен $\mathcal{O}(D^M)$ (Bishop, 2006). Как видно, в пространствах большей размерности полиномы могут стремительно становиться громоздкими и бесполезными с практической точки зрения.

Серьезные трудности, которые могут возникнуть в пространствах с большим количеством измерений, иногда называют *проклятием размерности*.

(*curse of dimensionality*) (Bellman, 1961). Оно не ограничивается полиномиальной регрессией, а является довольно распространенным явлением. Рассмотрим использование линейных моделей для решения задач классификации. На рис. 6.1 показана диаграмма из массива данных Iris, включающего 50 наблюдений каждого из трех видов цветков ириса. В каждом наблюдении есть четыре переменные, представляющие собой измерения длины и ширины чашелистика, длины и ширины лепестка. В этом примере рассматриваются только переменные длины и ширины чашелистика. С учетом этих 150 наблюдений в качестве обучающих данных целью нашей работы является классификация новой тестовой точки, например обозначенной на рис. 6.1 крестиком (\times), путем отнесения ее к одному из трех видов. Можно видеть, что крестик находится рядом с несколькими красными точками, и это позволяет предположить, что он принадлежит к красному классу. Однако рядом также есть несколько зеленых точек, поэтому вполне можно предположить, что он может принадлежать к зеленому классу. Менее вероятным кажется его принадлежность к синему классу. Интуиция подсказывает, что принадлежность крестика должна в большей степени определяться близлежащими точками из обучающего набора и в меньшей – более удаленными, и эта интуиция вполне резонна.

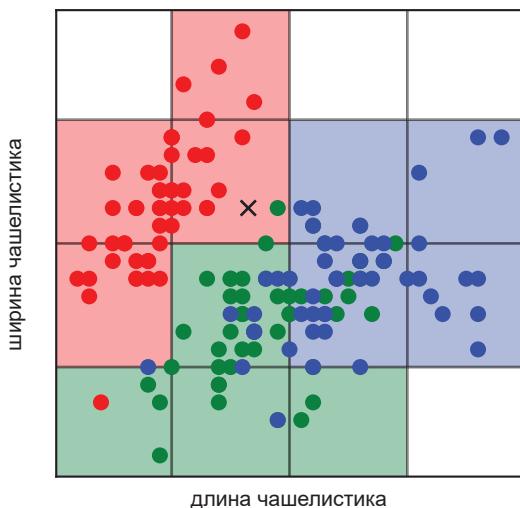
РИС. 6.1 Схема данных Iris, где красные, зеленые и синие точки обозначают три вида цветков ириса, а оси – измерения длины и ширины чашелистика соответственно



Один из очень простых способов преобразовать эту интуицию в алгоритм обучения состоит в разбиении входного пространства на регулярные ячейки, как показано на рис. 6.2. Получив тестовую точку и желая предсказать ее класс, необходимо сначала определить, к какой ячейке она относится, а затем найти все обучающие точки данных, которые попадают в эту же ячейку. Идентичность тестовой точки предсказывается как класс с наибольшим чис-

лом обучающих точек в той же ячейке, что и тестовая точка (при этом равные доли разбиваются случайным образом). Это можно рассматривать в качестве модели базисной функции, в которой для каждой ячейки сетки существует базисная функция $\varphi_i(x)$, при этом она просто возвращает ноль, если x лежит вне ячейки сетки, а в противном случае возвращает наиболее вероятный класс по всем точкам обучающих данных, которые попадают в ячейку. Выход модели определяется суммой выходов всех базисных функций. Впрочем, как вскоре будет показано, этот упрощенный подход обладает рядом серьезных недостатков.

РИС. 6.2 Иллюстрация простого подхода к решению задач классификации, когда входное пространство делится на ячейки и любая новая тестовая точка относится к классу с наибольшим числом членов в той же ячейке, что и тестовая точка



С этим упрощенным подходом связано множество проблем, но одна из самых серьезных становится очевидной при расширении его применения на задачи с большим числом входных переменных, соответствующих входным пространствам более высокой размерности. Истоки проблемы показаны на рис. 6.3. Если разделить область пространства на регулярные ячейки, то количество таких ячеек растет экспоненциально с увеличением размерности пространства. Проблема с экспоненциально большим числом ячеек заключается в необходимости экспоненциально большого количества обучающих данных для гарантии того, что ячейки не будут пустыми. На рис. 6.2 уже было показано, что некоторые ячейки не содержат обучающих точек. Следовательно, тестовая точка в таких ячейках не может быть классифицирована. Поэтому не стоит надеяться на применение такой техники в пространстве с большим числом переменных. Трудности как с примером полиномиальной регрессии, так и с примером классификации данных Iris возникают из-за выбора базисных функций безотносительно к решаемой задаче. Чтобы обойти проклятие размерности, необходимо более тщательно подходить к выбору базисных функций (см. раздел 6.1.4).

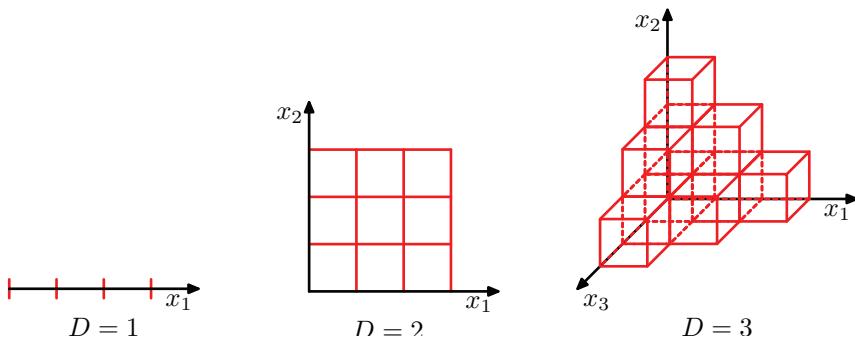


РИС. 6.3 Иллюстрация проклятия размерности: число областей регулярной сетки *растет экспоненциально с увеличением размерности D пространства*. Для наглядности показано только подмножество кубических областей для $D = 3$

6.1.2. Пространства большой размерности

Однако для начала обратимся к свойствам пространств с более высокой размерностью. Наши геометрические представления, сформированные в ходе жизни в трехмерном пространстве, здесь могут серьезно подвести. В качестве простого примера рассмотрим гиперсферу радиуса $r = 1$ в пространстве размерности D и зададимся вопросом, какова доля объема гиперсферы, лежащая между радиусами $r = 1 - \epsilon$ и $r = 1$. Эту долю можно оценить исходя из предпосылки, что объем $V_D(r)$ гиперсферы с радиусом r в пространстве размерности D должен масштабироваться как r^D , и поэтому запишем

$$V_D(r) = K_D r^D, \quad (6.4)$$

где постоянная K_D зависит только от D (см. упражнение 6.1). Искомая доля определяется следующим образом:

$$\frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = 1 - (1 - \epsilon)^D, \quad (6.5)$$

которая показана на рис. 6.4 как функция от различных значений D . Здесь показано, что при больших D эта доля стремится к 1 даже при малых значениях. Получается примечательный результат: в пространствах высокой размерности большая часть объема гиперсферы сосредоточена в тонкой оболочке вблизи поверхности!

В качестве еще одного примера, имеющего непосредственное отношение к машинному обучению, рассмотрим поведение гауссова распределения в пространстве высокой размерности. Если преобразовать декартовы координаты в полярные, а затем проинтегрировать направленные переменные, то получится выражение для плотности $p(r)$ как функции радиуса r от начала координат (см. упражнение 6.3). Таким образом, $p(r)\delta r$ – это масса вероятности внутри тонкой оболочки толщиной δr , расположенной по радиусу r . Это распределение для различных значений D показано на рис. 6.5, здесь

показано, что при больших D вероятностная мера гауссова распределения сосредоточена в тонкой оболочке по определенному радиусу.

РИС. 6.4 График доли объема гиперсферы радиуса $r = 1$, лежащей в диапазоне от $r = 1 - \epsilon$ до $r = 1$ для различных значений размерности D

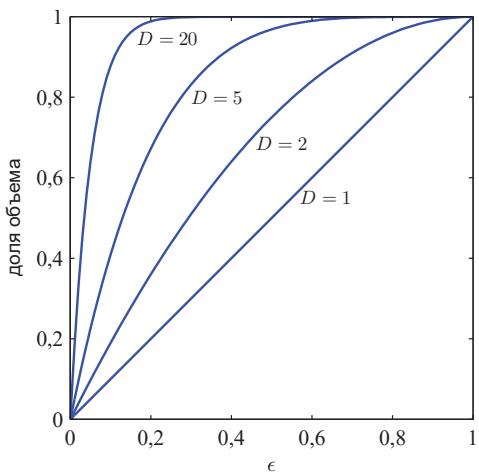
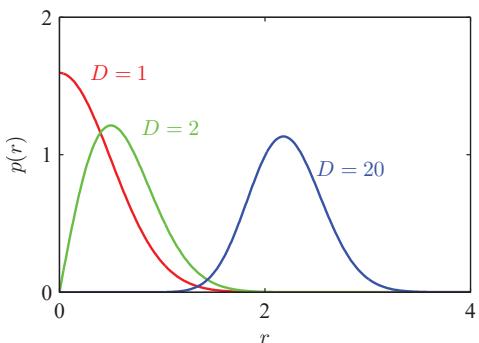


РИС. 6.5 График плотности вероятности по радиусу r гауссова распределения для различных значений размерности D



В этой книге широко используются иллюстративные примеры с одной или двумя переменными, поскольку это позволяет наиболее просто представить эти пространства в графической форме. Однако читателям следует предупредить, что не все интуитивные представления, развитые в пространствах с низкой размерностью, будут обобщены для ситуаций с большим числом измерений.

Наконец, хотя здесь и говорилось о проклятии размерности, работа в пространствах высокой размерности может иметь и свои преимущества. Рассмотрим ситуацию, показанную на рис. 6.6. Здесь показано, что набор данных, в котором каждая точка данных состоит из пары значений (x_1, x_2) , является линейно разделимым, но в случае, когда наблюдается только значение x_1 , классы существенно пересекаются. Поэтому задача классификации становится гораздо проще в пространстве более высокой размерности.

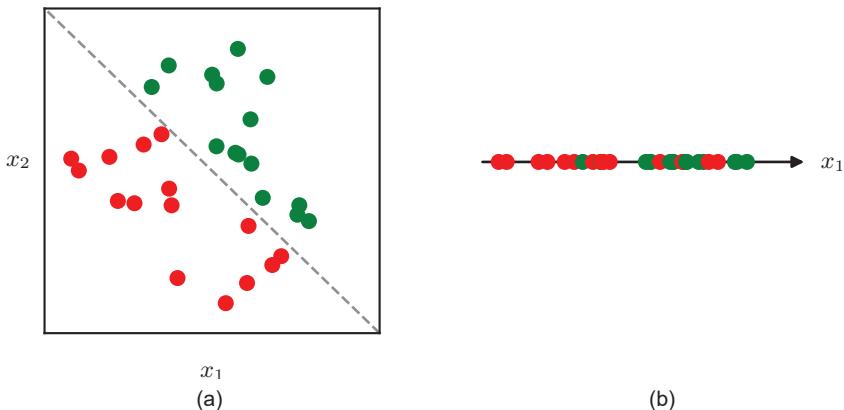


РИС. 6.6 Иллюстрация набора данных в двумерном пространстве (x_1, x_2) , где точки данных из двух классов (зеленые и красные кружки) могут быть разделены линейной поверхностью принятия решений, как показано в (а). Если измеряется только переменная x_1 , классы больше не разделяются, как показано в (б)

6.1.3. Многообразие данных

На примере полиномиальной регрессионной модели и классификатора на основе решетки на рис. 6.2 было показано, что с ростом размерности число базисных функций быстро увеличивается, что делает такие методы непрактичными для приложений с несколькими десятками переменных, не говоря уже о миллионах входных данных, которые часто возникают, например, при обработке изображений. Проблема состоит в том, что базисные функции фиксированы заранее и не зависят ни от данных, ни даже от конкретной решаемой задачи. Необходимо найти способ создания базисных функций для конкретного приложения.

Хотя проклятие размерности, безусловно, поднимает серьезные вопросы для приложений машинного обучения, оно не мешает создавать эффективные методы для работы с пространствами высокой размерности. Одна из причин этого заключается в том, что реальные данные, как правило, ограничены областью пространства данных с меньшей эффективной размерностью. Рассмотрим изображения на рис. 6.7. Каждое изображение – это точка в пространстве высокой размерности, при этом его размерность определяется количеством пикселей. Поскольку объекты могут находиться в разных вертикальных и горизонтальных положениях в пределах изображения, а также располагаться в разных ориентациях, существует три степени свободы изменения между изображениями, и набор изображений в первом приближении будет располагаться в трехмерном *многообразии* (*manifold*), встроенном в пространство высокой размерности. Из-за сложных взаимосвязей между положением или ориентацией объекта и интенсивностью пикселей это многообразие крайне нелинейно.

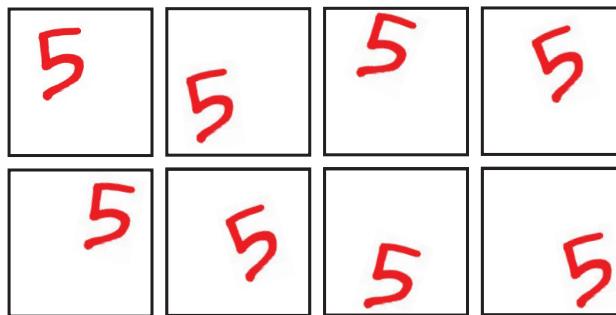


РИС. 6.7 Примеры изображений рукописных цифр с различным расположением и ориентацией цифр. Данные находятся в нелинейном трехмерном многообразии внутри пространства изображений высокой размерности

На самом деле количество пикселей – это лишь побочный эффект процесса создания изображения, поскольку они представляют собой измерения непрерывного процесса. Захват одного и того же изображения с более высоким разрешением увеличивает размерность D пространства данных, не меняя того факта, что изображения по-прежнему находятся в трехмерном многообразии. Если связать локализованные базисные функции с многообразием данных, а не со всем пространством данных высокой размерности, то можно ожидать, что количество необходимых базисных функций будет расти экспоненциально с размерностью многообразия, а не с размерностью пространства данных. Поскольку многообразие обычно имеет гораздо меньшую размерность, чем пространство данных, это является огромным плюсом. По сути, нейронные сети обучаются набору базисных функций, которые адаптированы к многообразиям данных. Более того, для конкретной задачи далеко не все направления внутри многообразия могут иметь значение. Например, если требуется определить только ориентацию, а не положение объекта на рис. 6.7, то в этом многообразии можно использовать только одну степень свободы, а не три. Нейронные сети также способны определить важные направления на многообразии для предсказания желаемых результатов.

Еще один способ убедиться в ограниченных возможностях реальных данных – обратиться к задаче генерации случайных изображений. На рис. 6.8 приведены примеры реальных изображений, а также синтетические изображения с тем же разрешением, которые были получены путем независимой случайной выборки интенсивности красного, зеленого и синего цветов для каждого пикселя из равномерного распределения. Как видно, ни одно из синтетических изображений совершенно не похоже на реальные. Причина кроется в том, что в этих случайных изображениях отсутствуют те самые устойчивые корреляции между пикселями, которые наблюдаются в реальных изображениях. Например, вероятность того, что два соседних пикселя на естественном изображении имеют одинаковый или очень похожий цвет, гораздо выше, чем для двух соседних пикселей в случайных примерах. Каждое из изображений на рис. 6.8 соответствует точке в пространстве высокой

размерности, однако реальные изображения охватывают лишь малую часть этого пространства.

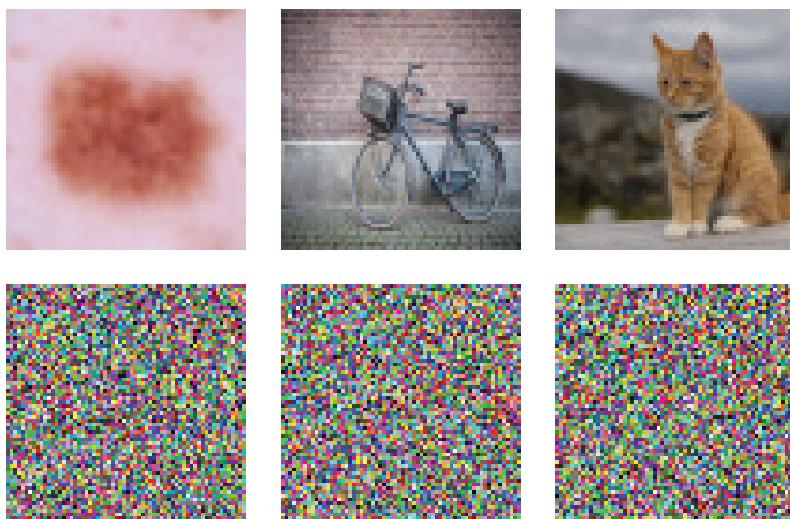


РИС. 6.8 В верхнем ряду показаны примеры естественных изображений размером 64×64 пикселя, в нижнем – случайно сгенерированные изображения того же размера, полученные путем извлечения значений пикселей из равномерного распределения вероятности по возможным цветам пикселей

6.1.4. Базисные функции на основе данных

Ранее было отмечено, что простые базисные функции, выбираемые независимо от решаемой задачи, могут привести к значительным ограничениям, особенно в пространствах высокой размерности. Если все же необходимо использовать базисные функции в таких ситуациях, то одним из подходящих методов является использование экспертных знаний для ручной настройки базисных функций с учетом специфики каждой конкретной задачи. В течение многих лет это был основной метод в машинном обучении. Базисные функции, часто также называемые *характеристиками* (*features*), определялись путем совмещения знаний о предметной области и метода проб и ошибок. Однако этот подход был малоэффективен и в конце концов был вытеснен методами с использованием данных, где базисные функции определяются на основе обучающих данных. Знания о предметной области по-прежнему играют роль в современном машинном обучении, но уже на более качественном уровне, когда дело доходит до создания сетевых архитектур, где они помогают зафиксировать соответствующее *индуктивное смещение* (*inductive bias*), как будет показано в последующих главах (см. раздел 9.1).

Поскольку данные в пространстве высокой размерности могут быть ограничены многообразием низкой размерности, не обязательно использовать базисные функции для плотного заполнения всего входного пространства.

Вместо этого можно использовать базисные функции, которые будут непосредственно связаны с многообразием данных. Одним из способов решения этой задачи может быть привязка одной базисной функции к каждой точке данных в обучающем наборе, что обеспечит автоматическую адаптацию базисных функций к базовому многообразию данных. Примером такой модели являются *радиальные базисные функции* (*radial basis functions*) (Broomhead and Lowe, 1988), свойством которых является зависимость каждой базисной функции только от радиального расстояния (обычно евклидова) от центрального вектора. Если центрами базиса являются значения входных данных $\{\mathbf{x}_n\}$, то для каждой точки данных существует одна базисная функция $\varphi_n(\mathbf{x})$, которая, таким образом, охватывает все многообразие данных. Типичным вариантом радиальной базисной функции будет

$$\varphi_n(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_n\|^2}{s^2}\right), \quad (6.6)$$

где s – это параметр измерения ширины базисной функции. Создание такой модели не требует больших усилий, однако основная проблема этого метода заключается в его вычислительной громоздкости для больших наборов данных. Кроме того, такая модель нуждается в тщательной регуляризации для предотвращения чрезмерной подгонки.

Еще один схожий подход, называемый *машиной опорных векторов* (*support vector machine*, или *SVM*) (Vapnik, 1995; Scholkopf and Smola, 2002; Bishop, 2006), справляется с этой проблемой путем определения базисных функций, сосредоточенных вокруг каждой из точек обучающих данных, а затем автоматического выбора подмножества этих функций в процессе обучения. В итоге эффективное число базисных функций в результирующих моделях обычно намного меньше числа точек обучения, хотя зачастую оно все равно остается относительно большим и, как правило, увеличивается с ростом размера обучающего набора. Машины опорных векторов также не обеспечивают вероятностных результатов и не способны к естественному обобщению более чем для двух классов. Такие методы, как радиальные базисные функции и машины опорных векторов, уже вытеснены глубокими нейронными сетями, которые гораздо лучше справляются с эффективным использованием огромных массивов данных. Более того, как будет показано далее, нейронные сети способны к обучению глубоким *иерархическим* представлениям, что крайне важно для достижения высокой точности предсказания в более сложных задачах.

6.2. Многослойные сети

В предыдущем разделе было отмечено, что для применения линейных моделей вида (6.1) к задачам с большими наборами данных в пространствах высокой размерности необходимо определить набор базисных функций, настроенных на конкретную задачу. Ключевая идея нейронных сетей заключается

в определении базисных функций $\varphi_j(\mathbf{x})$, которые сами обладают обучаемыми параметрами, и последующей настройке этих параметров вместе с коэффициентами $\{w_{ji}\}$ в процессе обучения. Затем вся модель оптимизируется путем минимизации функции ошибки с помощью градиентных методов оптимизации, таких как стохастический градиентный спуск (см. главу 7), где функция ошибки определяется совместно для всех параметров модели.

Несомненно, существует множество способов построения параметрических нелинейных базисных функций. Одно из ключевых требований к ним заключается в возможности дифференцировать их по обучаемым параметрам, чтобы иметь возможность применять градиентную оптимизацию. Наиболее удачным выбором является использование базисных функций в форме (6.1), где каждая базисная функция сама является нелинейной функцией линейной комбинации входов, а коэффициенты в линейной комбинации являются обучаемыми параметрами. Обратите внимание, что такую конструкцию можно рекурсивно расширить для получения иерархической модели со многими слоями (см. раздел 6.3), что является основой для создания глубоких нейронных сетей.

Рассмотрим базовую модель нейронной сети с двумя слоями обучаемых параметров. Сначала необходимо построить M линейных комбинаций входных переменных x_1, \dots, x_D в форме

$$a_j^{(1)} = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad (6.7)$$

где $j = 1, \dots, M$, а надстрочный индекс (1) указывает на то, что соответствующие параметры находятся в первом «слое» сети. В дальнейшем под параметрами $w_{ji}^{(1)}$ будут подразумеваться *веса (weights)*, под параметрами $w_{j0}^{(1)}$ – *смещения (biases)*, а величины $a_j^{(1)}$ будут называться *предварительными активациями (pre-activations)* (см. главу 4). Затем каждая из величин a_j преобразуется с помощью дифференцируемой нелинейной функции активации (*activation function*) $h(\cdot)$ для получения

$$z_j^{(1)} = h(a_j^{(1)}), \quad (6.8)$$

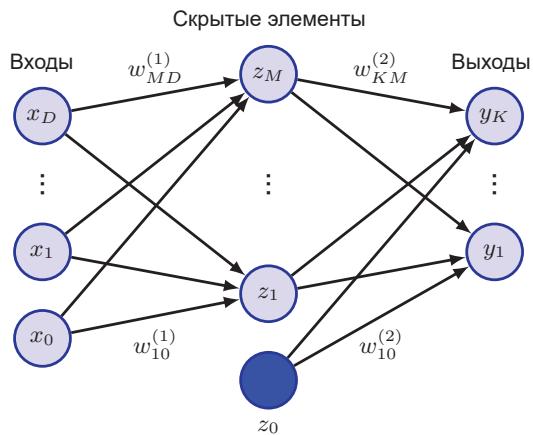
которые представляют собой выходы базисных функций в (6.1). В контексте нейронных сетей эти базисные функции называются *скрытыми элементами (hidden units)*. Ниже будут рассмотрены различные варианты выбора нелинейной функции $h(\cdot)$, но здесь следует отметить, что если производную $h'(\cdot)$ можно вычислить, то общая функция сети будет дифференцируемой. Согласно (6.1), эти значения снова линейно объединяются для получения

$$a_k^{(2)} = \sum_{j=1}^M w_{kj}^{(2)} z_j^{(1)} + w_{k0}^{(2)}, \quad (6.9)$$

где $k = 1, \dots, K$, а K – это общее количество выходов. Это преобразование соответствует второму слою сети, и вновь $w_{k0}^{(2)}$ – это параметры смещения. Наконец, $\{a_k^{(2)}\}$ преобразуются с помощью соответствующей функции активации выход-

ногого звена $f(\cdot)$, чтобы получить набор выходов сети y_k . Двухслойная нейронная сеть может быть представлена в виде схемы, как показано на рис. 6.9. Здесь входные, скрытые и выходные переменные представлены узлами, а весовые параметры – связями между узлами. Параметры смещения обозначены связями, исходящими из дополнительных входных и скрытых переменных x_0 и z_0 , которые сами обозначены сплошными узлами. Стрелки обозначают направление потока информации в сети при прямом распространении.

РИС. 6.9 Сетевая диаграмма двухслойной нейронной сети



6.2.1. Матрицы параметров

Как уже упоминалось в контексте моделей линейной регрессии (см. раздел 4.1.1), параметры смещения в (6.7) могут быть включены в набор весовых параметров путем определения дополнительной входной переменной x_0 , значение которой ограничено при $x_0 = 1$, так что (6.7) принимает вид:

$$a_j = \sum_{i=0}^D w_{ji}^{(1)} x_i. \quad (6.10)$$

Аналогичным образом можно включить смещения второго слоя в веса второго слоя, так что общая функция сети примет вид:

$$y_k(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=0}^M w_{kj}^{(2)} h\left(\sum_{i=0}^D w_{ji}^{(1)} x_i\right)\right). \quad (6.11)$$

Другой вариант записи, который будет удобен в дальнейшем, заключается в обозначении входов в виде вектора-столбца $\mathbf{x} = (x_1, \dots, x_N)^T$ и последующем объединении весовых параметров и параметров смещения в (6.11) в матрицы для получения

$$y(\mathbf{x}, \mathbf{w}) = f(\mathbf{W}^{(2)} h(\mathbf{W}^{(1)} \mathbf{x})), \quad (6.12)$$

где $f(\cdot)$ и $h(\cdot)$ оцениваются отдельно для каждого элемента вектора.

6.2.2. Универсальная аппроксимация

Способность двухслойной сети моделировать широкий спектр функций показана на рис. 6.10. Здесь также показано, как отдельные скрытые элементы совместными усилиями работают над аппроксимацией конечной функции. В каждом случае $N = 50$ точек данных (синие точки) были равномерно отобраны по x на интервале $(-1, 1)$ и оценены соответствующие значения $f(x)$. Затем эти точки данных используются для обучения двухслойной сети с тремя скрытыми блоками с функциями активации \tanh и линейными выходными блоками. Полученные функции сети показаны красными кривыми, а выходы трех скрытых блоков – тремя пунктирными кривыми. Роль скрытых элементов в простой задаче классификации показана на рис. 6.11. Пунктирные синие линии на этом рисунке показывают контуры $z = 0,5$ для каждого из скрытых блоков, а красная линия – поверхность принятия решения $y = 0,5$ для сети. Для сравнения: зеленые линии обозначают оптимальную границу принятия решения на основе вычисленных распределений, использованных для генерации данных.

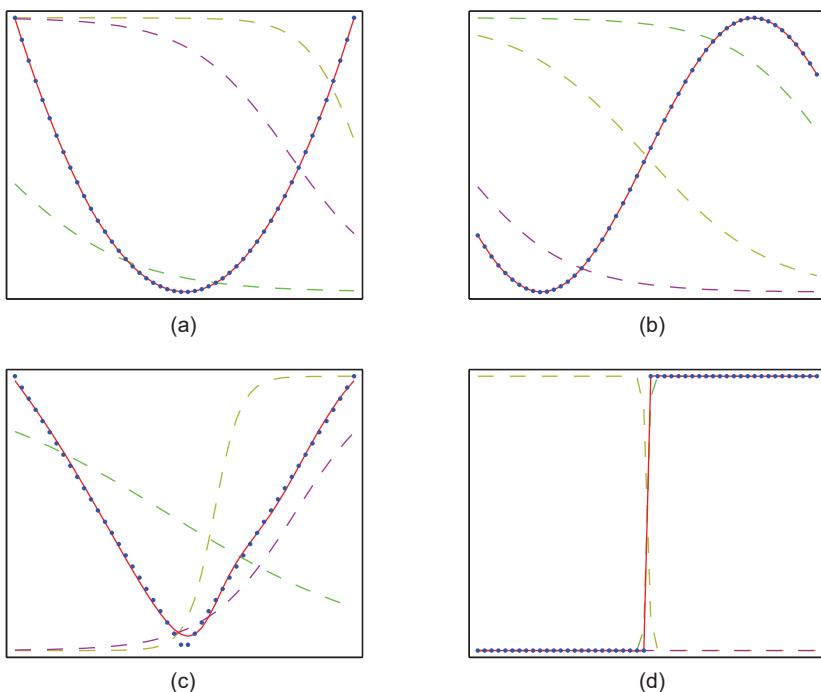
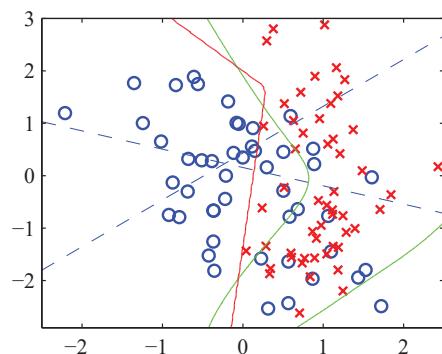


РИС. 6.10 Иллюстрация способности двухслойной нейронной сети аппроксимировать четыре различные функции: (a) $f(x) = x^2$, (b) $f(x) = \sin(x)$, (c) $f(x) = |x|$ и (d) $f(x) = H(x)$, где $H(x)$ – это ступенчатая функция Хевисайда

Аппроксимационные свойства двухслойных сетей с прямой передачей данных широко изучались в 1980-х годах. Различные теоремы указывают

на то, что для широкого диапазона функций активации такие сети могут аппроксимировать любую функцию, заданную на непрерывном подмножестве \mathbb{R}^D , с произвольной точностью (Funahashi, 1989; Cybenko, 1989; Hornik, Stinchcombe and White, 1989; Leshno et al., 1993). Аналогичный результат имеет место для функций перехода из любого дискретного пространства конечной размерности в любое другое. Поэтому считается, что нейронные сети представляют из себя универсальные аппроксиматоры.

РИС. 6.11 Пример решения простой задачи классификации по двум классам с использованием синтетических данных с помощью нейронной сети, имеющей два входа, два скрытых блока с функциями активации $tanh$ и один выход с функцией активации логистической сигмоидной функции



Несмотря на обнадеживающий характер таких теорем, в действительности они говорят лишь о существовании сети, способной представить искомую функцию. В некоторых случаях для этого может потребоваться сеть с экспоненциально большим числом скрытых элементов. Кроме того, ничего не говорится о способности такой сети быть обнаруженной алгоритмом обучения. Далее, в соответствии с известной теоремой об отсутствии бесплатного обеда (no free lunch theorem) будет показано, что найти по-настоящему универсальный алгоритм машинного обучения невозможно (см. раздел 9.1.2). И, наконец, хотя сети с двумя слоями весов являются универсальными аппроксиматорами, на практике можно получить гораздо больше преимуществ, если использовать сети с более чем двумя слоями, которые могут обучаться иерархическим внутренним представлениям. Все эти факторы свидетельствуют в пользу перехода к технологиям глубокого обучения.

6.2.3. Функции активации скрытых элементов

Как было отмечено ранее, функции активации для выходных блоков определяются типом моделируемого распределения. Однако для скрытых блоков единственным требованием является возможность их дифференцирования, что оставляет широкий простор для их применения. В большинстве случаев все скрытые элементы в сети будут иметь одну и ту же функцию активации, хотя фактически нет причин, по которым в разных частях сети нельзя было бы использовать разные варианты.

Самый простой вариант функции активации скрытых элементов – это функция тождества, которая подразумевает, что все скрытые элементы ста-

новятся линейными. Однако для любой такой сети всегда можно найти эквивалентную сеть без скрытых элементов. Это следует из того факта, что композиция последовательных линейных преобразований сама является линейным преобразованием, и потому ее возможности отображения не превышают таковые у одного линейного слоя. Однако если число скрытых элементов меньше числа входных или выходных элементов, то преобразования, которые может генерировать такая сеть, не являются наиболее общими линейными преобразованиями от входов к выходам, поскольку при сокращении размерности в скрытых элементах происходит потеря информации. Рассмотрим сеть с N входами, M скрытыми элементами и K выходами, в которой все функции активации линейны. Такая сеть имеет $M(N + K)$ параметров, в то время как линейное преобразование входов непосредственно в выходы имело бы NK параметров. Если M мало по отношению к N или K или к обоим параметрам, это приводит к тому, что двухслойная линейная сеть имеет меньше параметров, чем прямое линейное отображение, и соответствует преобразованию неполного ранга. Такие «узкие места» сети линейных элементов соответствуют стандартной технике анализа данных, называемой *анализом главных компонентов (principal component analysis)* (см. главу 16). В общем случае интерес к использованию многослойных сетей с линейными элементами ограничен, поскольку общая функция, вычисляемая такой сетью, все равно остается линейной.

Простой нелинейной дифференцируемой функцией является логистическая сигмоидная функция. Она задается как

$$\sigma(a) = \frac{1}{1 + \exp(-a)}, \quad (6.13)$$

а ее график изображен на рис. 5.12. Она широко использовалась в первые годы работы по созданию многослойных нейронных сетей, и на ее создание отчасти повлияли исследования свойств биологических нейронов. Близкородственной функцией является \tanh , которая определяется как

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}, \quad (6.14)$$

а ее график показан на рис. 6.12(а). Эта функция отличается от логистической сигмоидной линейным преобразованием входных и выходных значений, поэтому для любой сети с логистическими сигмоидными функциями активации скрытых элементов существует эквивалентная сеть с функциями активации \tanh (см. упражнение 6.4). Однако во время обучения сети они не обязательно эквивалентны, поскольку для оптимизации по градиенту необходима инициализация весов и смещений сети, поэтому если функции активации изменены, то схема инициализации должна быть скорректирована соответствующим образом. «Жесткая» версия функции \tanh (*'hard' tanh*) (Collobert, 2004) задается как

$$h(a) = \max(-1, \min(1, a)), \quad (6.15)$$

а ее график представлен на рис. 6.12b.

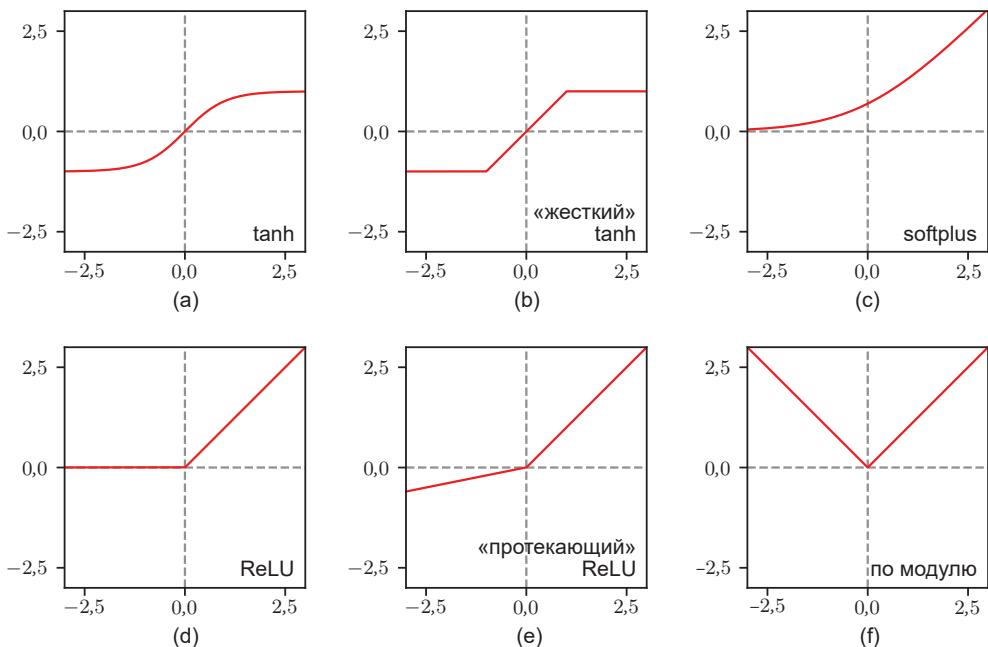


РИС. 6.12 Разнообразие нелинейных функций активации

Основной недостаток логистической сигмоидной и тангенциальной функций активации заключается в том, что их градиенты экспоненциально стремятся к нулю при больших положительных или больших отрицательных значениях на входе (см. раздел 7.4.2). Подробнее об этой проблеме «исчезающих градиентов» будет рассказано позже, а пока следует отметить, что, как правило, удобнее использовать функции активации с ненулевыми градиентами, по крайней мере, когда входные данные имеют большое положительное значение.

Одним из таких вариантов является функция активации *softplus* (*softplus activation function*) (см. упражнение 6.7), которая имеет вид

$$h(a) = \ln(1 + \exp(a)) \quad (6.16)$$

и которая показана на рис. 6.12c. Для значения $a \gg 1$ получается $h(a) \simeq a$, и поэтому градиент остается ненулевым даже при больших и положительных значениях входной функции активации, что помогает снять проблему исчезающих градиентов.

Еще более простым вариантом функции активации является выпрямленный линейный элемент (*rectified linear unit*, или *ReLU*), который определяется как

$$h(a) = \max(0, a) \quad (6.17)$$

и который показан на рис. 6.12d. Эмпирически это одна из самых эффективных функций активации, и поэтому она имеет широкое применение на практике. Обратите внимание, что, строго говоря, производная функции ReLU при $a = 0$ не определяется, но на практике этим можно смело пренебречь. Функцию softplus (6.16) можно рассматривать как сглаженную версию ReLU, поэтому ее также иногда называют *soft ReLU* (см. упражнение 6.5).

Хотя ReLU имеет ненулевой градиент для положительных входных значений, для отрицательных входных значений это не так. Это может означать, что некоторые скрытые элементы не получают «сигнала ошибки» во время обучения. Модификация ReLU, которая призвана избежать этой проблемы, называется «протекающим ReLU» (*leaky ReLU*) и определяется как

$$h(a) = \max(0, a) + \alpha \min(0, a), \quad (6.18)$$

где $0 < \alpha < 1$. Эта функция показана на рис. 6.12e. В отличие от ReLU она имеет ненулевой градиент для входных значений $a < 0$, что гарантирует наличие сигнала для проведения обучения. Вариант этой функции активации предполагает использование $\alpha = -1$, и тогда $h(a) = |a|$, что показано на рис. 6.12f. Другой вариант позволяет каждому скрытому элементу иметь свое собственное значение α_j , которое можно определить в процессе обучения сети путем оценки градиентов относительно $\{\alpha_j\}$ вместе с градиентами относительно весов и смещений.

Введение ReLU позволило значительно повысить эффективность обучения по сравнению с применявшимися ранее сигмоидными функциями активации (Krizhevsky, Sutskever and Hinton, 2012). Помимо возможности эффективного обучения более глубоких сетей, она гораздо менее чувствительна к случайной инициализации весов. Кроме того, она лучше подходит для реализаций с низкой точностью, например 8-битной фиксированной против 64-битной плавающей точки, и для ее оценки требуются незначительные вычислительные ресурсы. Во многих практических приложениях по умолчанию используются именно модули ReLU, если только не поставлена цель исследовать влияние различных вариантов функции активации.

6.2.4. Симметрии весового пространства

Одним из свойств сетей прямого распространения является способность нескольких различных вариантов весового вектора w приводить к одной и той же функции сопоставления (mapping function) между входами и выходами (Chen, Lu and Hecht-Nielsen, 1993). Рассмотрим двухслойную сеть, показанную на рис. 6.9, с M скрытыми элементами, которые характеризуются функциями активации \tanh и полной связностью в обоих слоях. Если изменить знак всех весов и смещений, подаваемых на определенный скрытый элемент, то для данной точки входных данных знак предварительной активации скрытого элемента изменится на противоположный, и, соответственно

но, изменится и активация, поскольку \tanh – это нечетная функция, так что $\tanh(-a) = -\tanh(a)$. Это преобразование можно полностью скорректировать путем изменения знака всех весов, выходящих из данного скрытого элемента. Таким образом, при изменении знаков определенной группы весов (и смещения) представленная сетью функция отображения входа-выхода остается неизменной, и теперь можно найти два разных вектора весов, которые приводят к одной и той же функции сопоставления. Для M скрытых единиц существует M таких симметрий «перестановки знаков», и получается, что любой заданный весовой вектор будет одним из множества 2^M эквивалентных весовых векторов.

Подобным же образом представьте, что значения всех весов (и смещения), ведущих как в определенный скрытый элемент, так и из него, заменяются на соответствующие значения весов (и смещения), связанные с другим скрытым элементом. И в этом случае функция сопоставления входа и выхода сети остается неизменной, но она соответствует другому выбору весового вектора. Для M скрытых элементов любой заданный весовой вектор будет относиться к множеству $M \times (M - 1) \times \dots \times 2 \times 1 = M!$ эквивалентных весовых векторов, связанных с этой симметрией чередования, соответствующих $M!$ различным расположениям скрытых элементов. Таким образом, общий коэффициент симметрии весового пространства сети составит $M!2^M$. Для сетей с более чем двумя слоями весов общий уровень симметрии будет определяться произведением таких коэффициентов, по одному на каждый слой скрытых элементов.

Как выяснилось, эти факторы учитывают все возможные симметрии в пространстве весов (за исключением возможных случайных симметрий, связанных с конкретным выбором значений весов). Более того, существование этих симметрий не является особым свойством функции \tanh , а применимо к широкому спектру функций активации (Kurkova and Kainen, 1994). В целом эти симметрии в пространстве весов не имеют большого практического значения, поскольку обучение сети направлено на поиск конкретного значения параметров, и поэтому возможность существования других (эквивалентных) значений не играет особой роли. Однако симметрии весового пространства действительно оказываются полезными при использовании байесовских методов для оценки распределения вероятностей в сетях разного размера (Bishop, 2006).

6.3. Глубокие сети

Развитие нейронных сетей было обусловлено стремлением сделать базисные функции линейной модели регрессии или классификации управляемыми обучаемыми параметрами. В результате появилась двухслойная модель сети, показанная на рис. 6.9. В течение многих лет это была наиболее широко используемая архитектура, главным образом потому, что эффективное обуче-

ние сетей с более чем двумя слоями оказалось затруднительным. Однако расширение нейронных сетей до более чем двух слоев, известных как глубокие нейронные сети, предоставляет множество преимуществ, о чём будет рассказано ниже, а последние достижения в области методов обучения нейронных сетей оказались весьма эффективными для сетей с большим количеством слоев (см. главу 7).

Архитектуру двухслойной сети (6.12) можно быстро дополнить до любого конечного числа слоёв L , при этом слой $l = 1, \dots, L$ вычисляет следующую функцию:

$$\mathbf{z}^{(l)} = h^{(l)}(\mathbf{W}^{(l)} \mathbf{z}^{(l-1)}), \quad (6.19)$$

где $h^{(l)}$ обозначает функцию активации, относящуюся к слою l , а $\mathbf{W}^{(l)}$ – соответствующую матрицу весов и параметров смещения. Кроме того, $\mathbf{z}^{(0)} = \mathbf{x}$ обозначает входной вектор, а $\mathbf{z}^{(L)} = \mathbf{y}$ – выходной вектор.

Обратите внимание, что в литературе наблюдается некоторая путаница в терминологии подсчета количества слоев в таких сетях. Так, сеть на рис. 6.9 иногда описывается как трехслойная (в ней подсчитывается количество слоев элементов, и при этом входы тоже рассматриваются как элементы), а иногда как однослойная (подсчитывается количество слоев скрытых элементов). Авторы этой книги рекомендуют использовать терминологию, в которой вариант на рис. 6.9 называется двухслойной сетью, поскольку для определения свойств сети важно именно количество слоев с обучаемыми весами.

Как уже было отмечено, сеть с двумя слоями обучаемых параметров, представленная на рис. 6.9, обладает универсальными возможностями аппроксимации. Однако сети с более чем двумя слоями иногда могут представлять заданную функцию с гораздо меньшим количеством параметров, чем двухслойная сеть. В работе (Montúfar et al., 2014) показано, что функция сети делит входное пространство на множество областей, и это множество экспоненциально по отношению к глубине сети, но полиномиально по отношению к ширине скрытых слоев. Для представления той же функции с помощью двухслойной сети потребовалось бы экспоненциальное число скрытых элементов.

6.3.1. Иерархические представления

Интересный результат, но более убедительная причина для изучения глубоких нейронных сетей заключается в том, что архитектура таких сетей координирует особую форму индуктивного смещения, а именно привязку выходов к входному пространству через иерархическое представление (см. главу 10). Хорошим примером является задача распознавания объектов на изображениях. Связь между пикселями изображения и таким высокоуровневым концептом, как «кот», очень сложна и нелинейна, и для двухслойной сети это было бы чрезвычайно сложной задачей. Но глубокие нейронные сети могут научиться обнаруживать низкоуровневые признаки, такие как края, в первых слоях, а затем объединять их в последующих слоях для формирования

признаков более высокого уровня, таких как глаза или усы, которые, в свою очередь, могут быть объединены в последующих слоях для распознавания наличия кота. Это можно рассматривать как *композиционное индуктивное смещение (compositional inductive bias)*, при котором объекты более высокого уровня, такие как коты, состоят из объектов более низкого уровня, таких как глаза, которые, в свою очередь, имеют элементы более низкого уровня, такие как края. Мы также можем представить это в обратном порядке, рассмотрев процесс создания изображения, начиная с низкоуровневых характеристик, таких как края, затем комбинируя их для формирования простых форм, таких как окружности, а затем, в свою очередь, комбинируя их для формирования объектов более высокого уровня, таких как коты. На каждом этапе существует множество способов объединения различных компонентов, что приводит к экспоненциальному росту числа возможностей с увеличением глубины.

6.3.2. Распределенные представления

Нейронные сети также могут воспользоваться преимуществами другой композиционной формы, называемой *распределенным представлением (distributed representation)*. Концептуально эту форму представления можно трактовать в качестве элементов скрытого слоя, каждый из которых соответствует «признаку» на данном уровне сети: высокое значение активации указывает на присутствие соответствующего признака, а низкое – на его отсутствие. При наличии M единиц в этом слое такая сеть может представлять M различных признаков. Однако потенциально сеть может научиться другому представлению, в котором признаки будут представлены комбинациями скрытых блоков, что позволит скрытому слою с M блоками представлять 2^M различных признаков, при этом их количество будет расти экспоненциально. Рассмотрим, например, сеть для обработки изображений лиц. На каждом конкретном изображении лица могут быть или не быть очки, может быть или не быть шляпа, может быть или не быть борода, что приводит к восьми различным комбинациям. Хотя это можно представить восемью элементами, каждый из которых «включается» при обнаружении соответствующей комбинации, более компактно это можно представить всего тремя элементами, по одному на каждый атрибут. Они могут присутствовать независимо друг от друга (хотя статистически их присутствие, скорее всего, будет в той или иной степени взаимосвязано). Подробнее о видах внутренних представлений, которые сети глубокого обучения открывают для себя в процессе обучения, будет рассказано позже (см. главу 10).

6.3.3. Обучение представлений

Последовательные слои глубокой нейронной сети можно рассматривать в качестве преобразователей данных, которые облегчают решение поставленной задачи или нескольких задач. Например, нейронная сеть, которая успешно учится классифицировать поражения кожи как доброкачественные или зло-

качественные, должна научиться преобразовывать исходные данные изображения в новое пространство (см. раздел 1.1.1), представленное выходами последнего слоя скрытых элементов таким образом, чтобы последний слой сети мог различать два класса. Этот последний слой можно рассматривать как простой линейный классификатор, и поэтому в представлении последнего скрытого слоя два класса должны быть хорошо разделены линейной поверхностью. Эта способность обнаружить нелинейное преобразование данных, которое облегчает решение последующих задач, называется *обучением представлениям* (*representation learning*) (Bengio, Courville and Vincent, 2012). Эта способность, иногда также называемая *встраиваемым пространством* (*embedding space*), задается выходами одного из скрытых слоев сети, так что любой входной вектор – либо из обучающего набора, либо из нового набора данных – может быть преобразован в это представление путем прямого распространения по сети.

Обучение представлениям является особенно эффективным инструментом, поскольку дает возможность использовать неразмеченные данные. Зачастую бывает несложно собрать большое количество неразмеченных данных, но гораздо сложнее бывает определить соответствующие метки. Например, видеокамера на автомобиле может собрать большое количество изображений городской среды во время движения по городу, однако получение этих изображений и идентификация соответствующих объектов, таких как пешеходы и дорожные знаки, потребует дорогостоящей и трудоемкой обработки данных человеком.

Обучение на основе немаркированных данных называется *неконтролируемым обучением* (или *обучением без учителя*, *unsupervised learning*), и для этого существует множество различных алгоритмов. Например, нейронная сеть может быть обучена получению изображений на входе и созданию таких же изображений на выходе. Для превращения такого процесса в нетривиальную задачу сеть может действовать скрытые слои с меньшим количеством элементов, чем число пикселей в изображении, тем самым заставляя сеть обучаться определенному способу сжатия изображений. В этом случае требуются только немаркированные данные, поскольку каждое изображение в обучающем наборе выступает в качестве как входного, так и целевого вектора. Такие сети называют *автокодировщиками* (*autoencoders*) (см. раздел 19.1). Целью такого обучения является принуждение сети к созданию некоторого внутреннего представления данных, полезного для решения других задач, таких как, например, классификация изображений.

Исторически сложилось так, что неконтролируемое обучение сыграло важную роль в успешном обучении первых глубоких сетей (не говоря уж о сверточных сетях). Каждый слой сети был сначала предварительно обучен с помощью неконтролируемого обучения, а затем вся сеть была обучена с помощью градиентного контролируемого обучения. Позже выяснилось, что фазу предварительного обучения можно пропустить, и глубокая сеть может быть обучена с нуля исключительно методом контролируемого обучения при наличии соответствующих условий.

Однако предварительное обучение и обучение представлений остаются центральными факторами глубокого обучения в других обстоятельствах. Наиболее ярким примером предварительного обучения является обработка естественного языка (см. главу 12), где преобразующие модели обучаются на больших объемах текста и способны изучать очень сложные внутренние представления языка, что обеспечивает впечатляющий диапазон возможностей на уровне человеческого языка и даже выше.

6.3.4. Трансферное обучение

Внутреннее представление, полученное в ходе обучения для какой-либо определенной задачи, может оказаться полезным и для похожих задач. Например, в результате обучения сети на большом наборе маркированных данных о повседневных объектах можно выработать способ преобразования представления изображения в представление для классификации объектов. Затем последний классификационный слой сети может быть переобучен на меньшем наборе маркированных данных с изображениями поражений кожи для создания классификатора этих поражений (см. раздел 1.1.1). Это пример *трансферного обучения* (*transfer learning*, также иногда называемого *переносом обучения*) (Hospedales et al., 2021), при котором достигается более высокая точность, чем если бы при обучении использовались только данные изображений поражения, поскольку сеть может использовать общие черты естественных изображений в целом. Схема трансферного обучения показана на рис. 6.13. На этой схеме: (a) сначала сеть обучается на задаче с большим количеством данных, например на классификации объектов на естественных изображениях; (b) ранние слои сети (показаны красным цветом) копируются

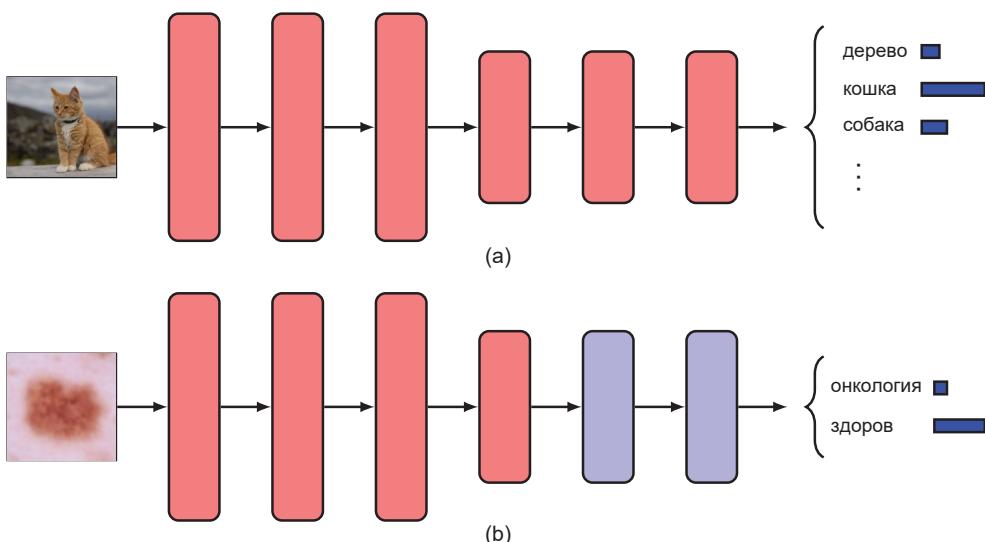


РИС. 6.13 Схематическое изображение трансферного обучения

из первой задачи, а последние несколько слоев сети (показаны синим цветом) переучиваются на новую задачу, например классификацию поражений кожи, для которой данных для обучения недостаточно, чтобы получить более точные результаты.

В общем случае трансферное обучение может быть использовано для повышения эффективности выполнения определенной задачи А, для которой не хватает обучающих данных, за счет использования данных из похожей задачи В, для которой данных больше. Две задачи должны иметь одинаковые входные данные, и между ними должна быть некоторая общность, чтобы низкоуровневые характеристики, или внутренние представления, полученные в задаче В, были полезны для задачи А. При изучении сверточных сетей (см. главу 10) будет показано, что многие задачи обработки изображений требуют аналогичных низкоуровневых характеристик, соответствующих начальным слоям глубокой нейронной сети, тогда как последующие слои более приспособлены к конкретной задаче, что делает такие сети пригодными для использования трансферного обучения.

При недостатке данных для решения задачи А можно просто переобучить последний слой сети. Напротив, если данных много, разумнее переобучить несколько слоев. Процесс обучения параметров одной задачи, которые затем применяются к одной или нескольким другим задачам, называется *предварительным обучением* (*pre-training*). Обратите внимание, что для новой задачи вместо применения стохастического градиентного спуска ко всей сети гораздо эффективнее один раз пропустить новые обучающие данные через фиксированную предварительно обученную сеть, чтобы затем оценить обучающие входы в новом представлении. В дальнейшем итеративная градиентная оптимизация может быть применена только к меньшей сети, состоящей из последних слоев. Помимо использования предварительно обученной сети в качестве фиксированного предварительного обработчика для другой задачи, можно также провести *точную настройку* (*fine-tuning*), при которой вся сеть адаптируется к данным для задачи А. Обычно это делается при очень малой скорости обучения за ограниченное число итераций, чтобы сеть не слишком подстраивалась под относительно небольшой набор данных новой задачи.

Близким по смыслу подходом является *многозадачное обучение* (*multitask learning*) (Caruana, 1997), в рамках которого сеть одновременно обучается более чем одной близкой по содержанию задаче. Например, требуется создать фильтр спама, позволяющий разным пользователям использовать различные классификаторы в соответствии с их индивидуальными предпочтениями. В качестве обучающих данных могут выступать примеры спама и не спама для множества различных пользователей, но количество примеров для одного пользователя может быть весьма ограниченным, и поэтому обучение отдельного классификатора для каждого пользователя даст недостаточно эффективные результаты. Вместо этого можно объединить наборы данных и обучить одну большую сеть, которая, например, может иметь общие начальные слои и отдельные обучаемые параметры для разных пользователей

в последующих слоях. Совместное использование данных для разных задач позволяет сети учитывать общие свойства задач, тем самым повышая точность для всех пользователей. При большом количестве обучающих примеров можно использовать более глубокую сеть с большим количеством параметров, что также способствует повышению производительности.

Обучение для выполнения нескольких задач может быть расширено до *метаобучения* (*meta-learning*), которое также называют «*обучением учиться*» (*learning to learn*). В то время как многозадачное обучение направлено на составление прогнозов для фиксированного набора задач, целью метаобучения является составление прогнозов для будущих задач, которые не рассматривались во время обучения. Этого можно добиться не только путем изучения общего внутреннего представления для всех задач, но также с помощью изучения самого алгоритма обучения (Hospedales et al., 2021). Метаобучение можно использовать для обобщения, например, классификационной модели для новых классов, когда имеется очень мало маркированных примеров новых классов. Это называется *обучением на ограниченных примерах* (*few-shot learning*). Когда используется только один маркированный пример, это называется *обучением по одному примеру* (*one-shot learning*).

6.3.5. Контрастивное обучение

Одним из наиболее распространенных и мощных методов обучения представлений является *контрастивное обучение* (*contrastive learning*) (Gutmann and Hyvärinen, 2010; Oord, Li and Vinyals, 2018; Chen, Kornblith, et al., 2020). Суть этой концепции заключается в обучении представлению таким образом, чтобы определенные пары входов, называемые положительными парами, были близки во вложенном пространстве, а другие пары входов, называемые отрицательными парами, находились далеко друг от друга. Суть заключается в том, что если подобрать семантически схожие положительные пары и семантически несхожие отрицательные пары, то в результате можно обучить пространство представления, в котором схожие входы будут близки, что значительно упростит последующие задачи, в том числе задачи классификации. Как и в других формах обучения представлениям, выходы обученной сети обычно не используются напрямую, а вместо этого для формирования пространства встраивания используются активации на каком-либо раннем слое. Контрастивное обучение отличается от большинства других методов машинного обучения тем, что функция ошибки для заданного входа определяется только по отношению к другим входам, а не по метке на входе или целевому выходу.

Пусть имеется заданная точка данных \mathbf{x} , называемая *якорем* (*anchor*), для которой задана другая точка данных \mathbf{x}^+ , составляющая вместе с \mathbf{x} положительную пару. Кроме того, необходимо задать набор точек данных $\{\mathbf{x}_1^-, \dots, \mathbf{x}_N^-\}$, каждая из которых составляет отрицательную пару с \mathbf{x} . Теперь необходимо найти функцию потерь, которая будет поддерживать близость между представлениями \mathbf{x} и \mathbf{x}^+ и в то же время увеличивать расстояние между каждой

парой $\{\mathbf{x}, \mathbf{x}_n^-\}$. Одним из примеров такой функции и наиболее часто применяемой функцией потерь для контрастивного обучения является *функция потерь InfoNCE* (Gutmann and Hyvarinen, 2010; Oord, Li and Vinyals, 2018), где NCE обозначает «сравнительная оценка шума» (*noise contrastive estimation*). Пусть существует нейросетевая функция $\mathbf{f}_w(\mathbf{x})$, которая отображает точки из входного пространства \mathbf{x} в пространство представления, управляемое обучаемыми параметрами w . Это представление нормализовано таким образом, что $\|\mathbf{f}_w(\mathbf{x})\| = 1$. Тогда для точки данных \mathbf{x} потери InfoNCE определяются как

$$E(w) = -\ln \frac{\exp\{\mathbf{f}_w(\mathbf{x})^T \mathbf{f}_w(\mathbf{x}^+)\}}{\exp\{\mathbf{f}_w(\mathbf{x})^T \mathbf{f}_w(\mathbf{x}^+)\} + \sum_{n=1}^N \exp\{\mathbf{f}_w(\mathbf{x})^T \mathbf{f}_w(\mathbf{x}_n^-)\}}. \quad (6.20)$$

Как видно, в этой функции значение косинусного сходства $\mathbf{f}_w(\mathbf{x})^T \mathbf{f}_w(\mathbf{x}^+)$ между представлением $\mathbf{f}_w(\mathbf{x})$ якоря и представлением $\mathbf{f}_w(\mathbf{x}^+)$ положительного образца служит мерой приближения положительных пар образцов в изученном пространстве, и эта же мера используется для оценки приближения якоря к отрицательным образцам. Обратите внимание, что эта функция напоминает функцию ошибки перекрестной энтропии классификации, в которой косинусное сходство положительной пары дает логит для класса метки, а косинусное сходство отрицательных пар – логиты для некорректных классов. Также следует отметить, что отрицательные пары имеют большое значение, поскольку без них встраивание просто обучится вырожденному решению, которое заключается в отображении каждой точки на одно и то же представление.

Конкретный алгоритм контрастивного обучения определяется преимущественно методом выбора положительных и отрицательных пар, т. е. посредством использования имеющихся знаний для определения параметров правильного представления. Например, рассмотрим проблему обучения представлений изображений. Здесь распространенным выбором является создание положительных пар путем изменения (искажения) входных изображений так, чтобы сохранить семантическую информацию образа, но при этом сильно изменить его в пиксельном пространстве (Wu et al., 2018; He et al., 2019; Chen, Kornblith, et al., 2020). Искажения тесно связаны с методикой *аугментации (расширения) данных (data augmentation)* (см. раздел 9.1.3), а в число примеров таких искажений входят повороты, преобразования и изменения цвета. Для создания отрицательных пар можно использовать другие изображения набора данных. Такой подход к контрастивному обучению известен как *дискриминация по образцу (instance discrimination)*.

Если же есть доступ к меткам классов, то можно использовать изображения одного класса в качестве положительных пар, а изображения разных классов – в качестве отрицательных пар. Это позволяет ослабить зависимость от указания дополнений, к которым представление должно быть инвариантным, а также избежать необходимости рассматривать два семантически схожих изображения как отрицательную пару. Это называется контролируемым контрастивным обучением (Khosla et al., 2020) ввиду зависимости от меток

классов, и зачастую оно может дать лучшие результаты, чем простое обучение представления с помощью классификации по перекрестной энтропии.

Элементы положительных и отрицательных пар не обязательно должны быть из одной и той же модальности данных. В модели *предварительного обучения контрастивно-языковым изображениям* (*contrastive-language image pretraining*, или *CLIP*) (Radford et al., 2021) положительная пара состоит из изображения и соответствующей ему текстовой подписи, и две отдельные функции, по одной для каждой модальности, используются для отображения входных данных в одно и то же пространство представления. Отрицательные пары – это несовпадающие изображения и подписи. Этот метод часто называют *слабо контролируемым* (*weakly supervised*), поскольку в его основе лежат изображения с подписями, которые зачастую проще получить путем сбора данных из интернета, чем размечать изображения по классам вручную. Функция потерь в этом случае имеет вид:

$$\begin{aligned} E(\mathbf{w}) &= -\frac{1}{2} \ln \frac{\exp\{\mathbf{f}_w(\mathbf{x}^+)^T \mathbf{g}_\theta(\mathbf{y}^+)\}}{\exp\{\mathbf{f}_w(\mathbf{x}^+)^T \mathbf{g}_\theta(\mathbf{y}^+)\} + \sum_{n=1}^N \exp\{\mathbf{f}_w(\mathbf{x}_n^-)^T \mathbf{g}_\theta(\mathbf{y}^+)\}} \\ &= -\frac{1}{2} \ln \frac{\exp\{\mathbf{f}_w(\mathbf{x}^+)^T \mathbf{g}_\theta(\mathbf{y}^+)\}}{\exp\{\mathbf{f}_w(\mathbf{x}^+)^T \mathbf{g}_\theta(\mathbf{y}^+)\} + \sum_{m=1}^M \exp\{\mathbf{f}_w(\mathbf{x}^+)^T \mathbf{g}_\theta(\mathbf{y}_m^-)\}}, \end{aligned} \quad (6.21)$$

где \mathbf{x}^+ и \mathbf{y}^+ представляют положительную пару, в которой \mathbf{x} – это изображение, а \mathbf{y} – это соответствующая ему текстовая подпись, \mathbf{f}_w представляет собой отображение из изображений в пространство представления, а \mathbf{g}_θ – это отображение из текстового ввода в пространство представления. Кроме того, также необходимо множество $\{\mathbf{x}_1^-, \dots, \mathbf{x}_N^-\}$ других изображений из набора данных, для которых можно предположить, что текстовая подпись \mathbf{y}^+ неуместна, и набор $\{\mathbf{y}_1^-, \dots, \mathbf{y}_M^-\}$ текстовых подписей, которые аналогичным образом не соответствуют входному изображению \mathbf{x} . Два члена в функции потерь гарантируют, что (a) представление изображения близко к представлению его текстовой подписи относительно других представлений изображения и (b) представление текстовой подписи близко к представлению изображения, которое она описывает, относительно других представлений текстовых подписей. Хотя в CLIP используются пары текст–изображение, для обучения представлений можно использовать любой набор данных с парными модальностями. Сравнение рассмотренных ранее методов контрастивного обучения показано на рис. 6.14. ((a) Метод дискриминации по образцу, при котором положительная пара состоит из якоря и дополненной версии одного и того же изображения. Они отображаются на точки в нормализованном пространстве, которое можно в виде единичной гиперсферы. Цветные стрелки показывают, что потеря помогает сблизить представления положительной пары, но при этом отделяет отрицательные пары друг от друга. (b) Контролируемое контрастивное обучение, при котором положительная пара состоит из двух разных изображений из одного класса. (c) Модель CLIP, в которой положительная пара состоит из изображения и связанного с ним фрагмента текста.)

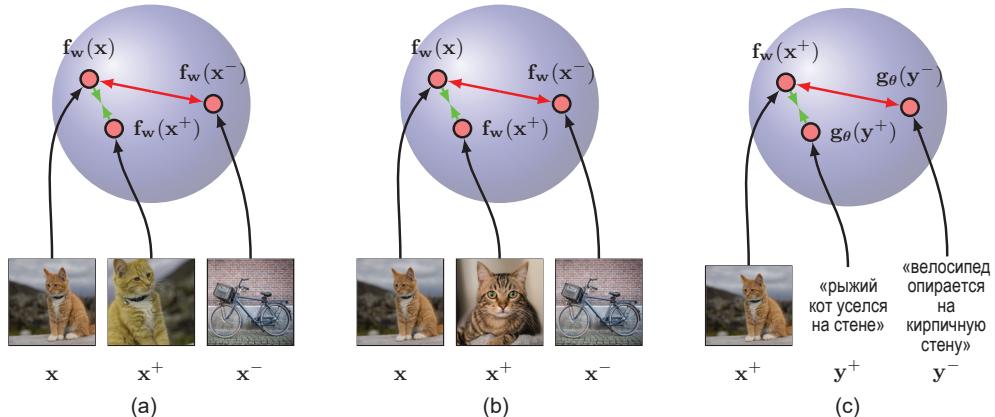


РИС. 6.14 Иллюстрация трех различных парадигм контрастивного обучения

6.3.6. Основные сетевые архитектуры

До сих пор речь шла об архитектуре нейронных сетей, состоящих из последовательности полностью связанных между собой слоев. Поскольку существует прямое соответствие между конфигурацией сети и ее математической функцией, можно разработать более общий вариант отображения сети при помощи более сложных сетевых структур. При этом необходимо ограничиться *архитектурой прямой передачи (feed-forward architecture)*, не имеющей замкнутых направленных циклов, чтобы гарантированно определить соответствие выходов детерминированным функциям входов. Это можно проиллюстрировать на простом примере на рис. 6.15. Обратите внимание, что каждый скрытый и выходной блок на рисунке имеет соответствующий параметр смещения (опущен для наглядности). Каждый (скрытый или выходной) элемент такой сети вычисляет функцию, заданную как

$$z_k = h\left(\sum_{j \in \mathcal{A}(k)} w_{kj} z_j + b_k\right), \quad (6.22)$$

где $\mathcal{A}(k)$ обозначает множество *предшественников (ancestors)* узла k , т. е. множество узлов, которые передают связи к узлу k , а b_k обозначает соответствую-

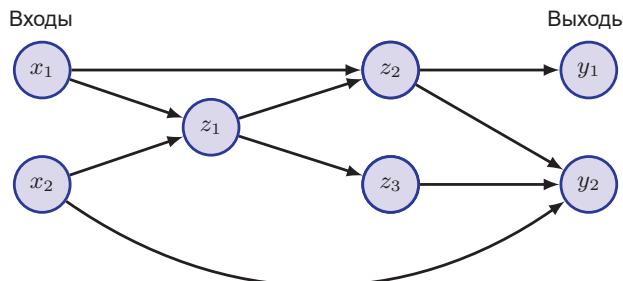


РИС. 6.15 Пример нейронной сети с общей топологией прямой передачи данных

щий параметр смещения. Для заданного набора значений, подаваемых на входы сети, последовательное применение (6.22) позволяет оценить активации всех узлов сети, включая активации выходных узлов.

6.3.7. Тензоры

Линейная алгебра является центральным элементом нейронных сетей, где такие величины, как массивы данных, активации и параметры сети, представлены в виде скаляров, векторов и матриц. Однако зачастую встречаются переменные более высокой размерности. Рассмотрим, например, набор данных из N цветных изображений, каждое из которых имеет высоту I пикселей и ширину J пикселей. Каждый пиксель индексируется по строке и столбцу изображения, и ему соответствуют значения красного, зеленого и синего цветов. Для каждого изображения в наборе данных имеется одно такое значение, поэтому конкретное значение интенсивности можно представить четырехмерным массивом \mathbf{X} с элементами x_{ijkn} , где $i \in \{1, \dots, I\}$ и $j \in \{1, \dots, J\}$ обозначают строку и столбец изображения, $k \in \{1, 2, 3\}$ означает интенсивность красного, зеленого и синего цветов, а $n \in \{1, \dots, N\}$ индексирует конкретное изображение в наборе данных. Такие массивы более высокой размерности называются тензорами (tensors). Они включают в себя в качестве частных случаев скаляры, векторы и матрицы. В дальнейшем в книге будет приведено множество примеров таких тензоров при изучении более сложных архитектур нейронных сетей. Для обработки тензоров особенно хорошо подходят мощные параллельные процессоры, такие как GPU.

6.4. Функции ошибок

В предыдущих главах рассматривались линейные модели для регрессии и классификации (см. главу 4), и в процессе изучения были выведены подходящие формы для функций ошибки, а также соответствующие варианты функций активации выходного элемента (см. главу 5). Для многослойных нейронных сетей применимы те же принципы выбора функции ошибки, поэтому для удобства здесь будут кратко изложены основные моменты.

6.4.1. Регрессия

Для начала рассмотрим вопросы регрессии, для чего определим одиночную целевую переменную t , которая может принимать любое реальное значение. Следуя описанию регрессии в однослойных сетях (см. раздел 2.3.4), будем считать, что t имеет гауссово распределение со средним значением, зависящим от \mathbf{x} , которое задается выходом нейронной сети, так что

$$p(t | \mathbf{x}, \mathbf{w}) = \mathcal{N}(t | y(\mathbf{x}, \mathbf{w}), \sigma^2), \quad (6.23)$$

где σ^2 – это дисперсия гауссова шума. Конечно, это предположение носит несколько ограничительный характер, и в некоторых задачах понадобится расширить этот метод с учетом более общих распределений (см. раздел 6.5). Для условного распределения, заданного в (6.23), достаточно принять функцию активации выходного блока за тождество, поскольку такая сеть может аппроксимировать любую непрерывную функцию от x до y . С учетом набора данных из *N* i.i.d.-наблюдений $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, а также соответствующих целевых значений $\mathbf{t} = \{t_1, \dots, t_N\}$ можно вывести соответствующую функцию правдоподобия:

$$p(\mathbf{t} | \mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N p(t_n | y(\mathbf{x}_n, \mathbf{w}), \sigma^2). \quad (6.24)$$

Обратите внимание, что в литературе по машинному обучению принято рассматривать минимизацию функции ошибки, а не максимизацию правдоподобия, поэтому здесь тоже будем следовать этому соглашению. Взяв отрицательный логарифм функции правдоподобия (6.24), можно получить функцию ошибки:

$$\frac{1}{2\sigma^2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2} \ln \sigma^2 + \frac{N}{2} \ln(2\pi), \quad (6.25)$$

которая позволяет узнать параметры \mathbf{w} и σ^2 . Сначала рассмотрим определение \mathbf{w} . Максимизация функции правдоподобия эквивалентна минимизации функции ошибки по сумме квадратов, которая задается как

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2, \quad (6.26)$$

где отброшены аддитивные и мультипликативные константы. Значение \mathbf{w} , найденное путем минимизации $E(\mathbf{w})$, обозначим как \mathbf{w}^* . Обратите внимание, что обычно оно не соответствует глобальному максимуму функции правдоподобия, поскольку нелинейность сетевой функции $y(\mathbf{x}_n, \mathbf{w})$ приводит к невыпуклой ошибке $E(\mathbf{w})$, и поэтому поиск глобального оптимума в общем случае невозможен. Более того, в функцию ошибки могут быть добавлены члены регуляризации (см. главу 9), а в процесс обучения могут быть внесены другие изменения, так что результирующее решение для параметров сети может значительно отличаться от решения максимального правдоподобия.

Определив \mathbf{w}^* , можно найти значение σ^2 путем минимизации функции ошибки (6.25) (см. упражнение 6.8), что дает

$$\sigma^{2*} = \frac{1}{N} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}^*) - t_n\}^2. \quad (6.27)$$

Обратите внимание, что это значение можно оценить после завершения итерационной оптимизации, необходимой для нахождения \mathbf{w}^* .

При наличии нескольких целевых переменных и условии, что они независимы и обусловлены \mathbf{x} и \mathbf{w} с общей дисперсией шума σ^2 , условное распределение целевых значений определяется как

$$p(\mathbf{t} | \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{t} | \mathbf{y}(\mathbf{x}, \mathbf{w}), \sigma^2 \mathbf{I}). \quad (6.28)$$

Следуя той же аргументации, что и для одной целевой переменной, можно увидеть, что максимизация функции правдоподобия относительно весов эквивалентна минимизации функции ошибки по сумме квадратов (см. упражнение 6.9):

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2. \quad (6.29)$$

В результате дисперсия шума определяется как

$$\sigma^{2\star} = \frac{1}{NK} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}^\star) - \mathbf{t}_n\|^2, \quad (6.30)$$

где K – это размерность целевой переменной. Предположение об условной независимости целевых переменных можно отбросить при несколько более сложной оптимизационной задаче (см. упражнение 6.10).

Помните, что существует естественное сопряжение функции ошибки, задаваемой отрицательным логарифмом правдоподобия (см. раздел 5.4.6), и функции активации выходного элемента. В регрессии можно рассматривать сеть с функцией активации на выходе, которая является тождеством, так что $y_k = a_k$. Тогда соответствующая функция ошибки суммы квадратов имеет свойство:

$$\frac{\partial E}{\partial a_k} = y_k - t_k. \quad (6.31)$$

6.4.2. Бинарная классификация

Теперь рассмотрим бинарную классификацию с единственной целевой переменной t , где $t = 1$ обозначает класс \mathcal{C}_1 , а $t = 0$ означает класс \mathcal{C}_2 . В продолжение обсуждения канонических функций связи (см. раздел 5.4.6) рассмотрим сеть с одним выходом, чьей функцией активации является логистическая сигмоидальная функция в (6.13), так что $0 \leq y(\mathbf{x}, \mathbf{w}) \leq 1$. Тогда можно интерпретировать $y(\mathbf{x}, \mathbf{w})$ как условную вероятность $p(\mathcal{C}_1 | \mathbf{x})$, а $p(\mathcal{C}_2 | \mathbf{x})$ в этом случае будет равна $1 - y(\mathbf{x}, \mathbf{w})$. В этом случае условное распределение целей с учетом входов будет иметь вид распределения Бернулли:

$$p(t | \mathbf{x}, \mathbf{w}) = y(\mathbf{x}, \mathbf{w})^t \{1 - y(\mathbf{x}, \mathbf{w})\}^{1-t}. \quad (6.32)$$

Если взять обучающий набор из независимых наблюдений, то функция ошибки, которая задается отрицательным логарифмическим правдоподобием, является ошибкой перекрестной энтропии в виде

$$E(\mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1-t_n) \ln(1-y_n)\}, \quad (6.33)$$

где y_n обозначает $y(\mathbf{x}_n, \mathbf{w})$. В работе (Simard, Steinkraus and Platt, 2003) отмечается, что использование функции ошибки перекрестной энтропии в задачах классификации вместо суммы квадратов приводит к ускорению обучения, а также к повышению качества обобщения.

Обратите внимание, что в (6.32) нет эквивалента дисперсии шума σ^2 , поскольку предполагается заведомо правильная маркировка целевых значений. Однако такая модель может быть легко расширена (см. упражнение 6.11) для учета ошибок маркировки путем введения вероятности ϵ того, что целевое значение t было заменено на неправильное (Opper and Winther, 2000). Вероятность ϵ может быть задана заранее или может рассматриваться в качестве гиперпараметра, значение которого определяется на основе данных.

Если есть необходимость в проведении K отдельных бинарных классификаций, то можно использовать сеть с K выходами, каждый из которых обладает логистической сигмоидной функцией активации. Каждый выход имеет бинарную метку класса $t_k \in \{0, 1\}$, где $k = 1, \dots, K$. Если предположить, что метки классов независимы, то условное распределение целей при заданном входном векторе имеет вид:

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}) = \prod_{k=1}^K y_k(\mathbf{x}, \mathbf{w})^{t_k} [1 - y_k(\mathbf{x}, \mathbf{w})]^{1-t_k}. \quad (6.34)$$

Взятие отрицательного логарифма соответствующей функции правдоподобия дает следующую функцию ошибки (см. упражнение 6.13):

$$E(\mathbf{w}) = -\sum_{n=1}^N \sum_{k=1}^K \{t_{nk} \ln y_{nk} + (1-t_{nk}) \ln(1-y_{nk})\}, \quad (6.35)$$

где y_{nk} обозначает $y_k(\mathbf{x}_n, \mathbf{w})$. И вновь производная функции ошибки касательно предварительной активации для конкретного выхода (см. упражнение 6.14) принимает вид (6.31), как и в случае регрессии.

6.4.3. Многоклассовая классификация

Наконец, рассмотрим стандартную задачу многоклассовой классификации, где каждый вход отнесен к одному из K взаимоисключающих классов. Двоичные целевые переменные $t_k \in \{0, 1\}$ имеют схему кодирования «1 из K », указывающую на принадлежность к определенному классу (см. раздел 5.1.3), а выходы сети интерпретируются как $y_k(\mathbf{x}, \mathbf{w}) = p(t_k = 1 | \mathbf{x})$, что приводит к использованию функции ошибки (5.80) вида

$$E(\mathbf{w}) = -\sum_{n=1}^N \sum_{k=1}^K t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w}). \quad (6.36)$$

Функция активации выходного элемента, которая соответствует канонической связи, задается функцией softmax (см. раздел 5.4.4):

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{\exp(a_k(\mathbf{x}, \mathbf{w}))}{\sum_j \exp(a_j(\mathbf{x}, \mathbf{w}))}, \quad (6.37)$$

что удовлетворяет условию $0 \leq y_k \leq 1$ и $\sum_k y_k = 1$. Обратите внимание, что $y_k(\mathbf{x}, \mathbf{w})$ остается неизменным при добавлении константы ко всем $a_k(\mathbf{x}, \mathbf{w})$. Это приводит к тому, что для некоторых направлений в весовом пространстве функция ошибки становится постоянной. Это вырождение можно устранить, если к функции ошибки добавить соответствующий член регуляризации (см. главу 9). И вновь производная функции ошибки по отношению к предварительной активации для конкретного выходного элемента принимает знакомый вид (6.31) (см. упражнение 6.15).

В итоге можно отметить, что выбор функции активации выходного элемента и функции ошибки согласования в зависимости от типа решаемой задачи является вполне логичным. Для регрессии можно использовать линейные выходы и ошибку суммы квадратов, для множественной независимой бинарной классификации подходят логистические сигмоидные выходы и функция ошибки перекрестной энтропии, а для многоклассовой классификации нужны выходы softmax с соответствующей функцией ошибки перекрестной энтропии для нескольких классов. Для задач классификации по двум классам можно использовать одну логистическую сигмоидную выходную величину или сеть с двумя выходами и функцией активации softmax на выходе.

Эта процедура достаточно универсальна, и при выборе других форм условия распределения можно вывести соответствующие функции ошибки, такие как соответствующее отрицательное логарифмическое правдоподобие. Подобный пример будет рассмотрен в следующем разделе, где речь пойдет о мультимодальных выходах сети.

6.5. Сети смешанной плотности

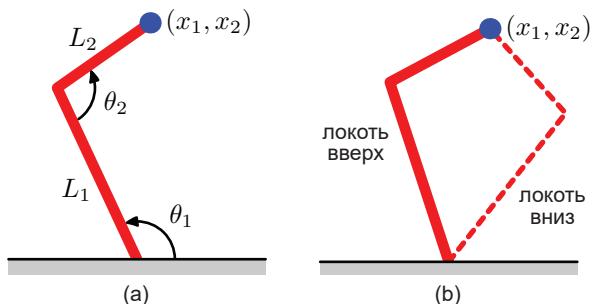
До сих пор в этой главе рассматривались нейронные сети, выходы которых представляют собой простые распределения вероятностей, начиная от гауссова распределения для непрерывных переменных и заканчивая бинарным распределением для дискретных переменных. В заключение главы рассмотрим возможности нейронной сети для представления более общих условных вероятностей, для чего необходимо рассматривать выходы сети как параметры более сложного распределения, в данном случае модели гауссовой смеси. Такая структура называется сетью смешанной плотности (mixture density network), и далее речь пойдет о методах определения соответствующей функции ошибки и соответствующих функций активации выходных элементов.

6.5.1. Пример кинематики робота

Целью контролируемого обучения является моделирование условного распределения $p(\mathbf{t} | \mathbf{x})$, при этом для решения многих простых задач регрессии оно выбирается как гауссово распределение. Однако практические задачи машинного обучения зачастую характеризуются распределением, существенно отличающимся от гауссова. Такие ситуации могут возникнуть, в частности, при решении *обратных задач* (*inverse problems*) с мультимодальным распределением, и в этом случае использование гауссова распределения может привести к весьма неудачным прогнозам.

В качестве простой иллюстрации обратной задачи рассмотрим кинематику руки робота (см. упражнение 6.16), как показано на рис. 6.16. *Прямая задача* (*forward problem*) состоит в определении позиции концевого механизма с учетом углов поворота суставов и имеет единственное решение. Однако на практике возникает необходимость переместить концевой механизм робота в конкретное положение, а для этого нужно задать соответствующие углы в суставах. Поэтому необходимо решить обратную задачу, которая имеет два решения, как показано на рис. 6.16. ((a) Роботизированная рука с двумя звенями, где декартовы координаты (x_1, x_2) концевого механизма определяются исключительно двумя суставными углами θ_1 и θ_2 и (фиксированными) длинами звеньев L_1 и L_2 . Такая конструкция известна как *прямая кинематика* (*forward kinematics*) руки. (b) На практике требуется найти такие углы сочленений, которые приведут к желаемому положению концевого механизма. Эта задача *обратной кинематики* (*inverse kinematics*) имеет два решения, которые соответствуют положениям «локоть вверх» и «локоть вниз».)

РИС. 6.16 Роботизированная рука с двумя суставными углами



Прямые задачи часто соответствуют причинно-следственным связям в физической системе и, как правило, имеют уникальное решение. Например, специфический набор симптомов в человеческом организме может быть вызван наличием конкретного заболевания. Однако в машинном обучении обычно приходится решать обратную задачу, например прогнозировать появление болезни по имеющемуся набору симптомов. Если прямая задача включает в себя отображение «многие к одному», то обратная задача будет иметь множество решений. Например, несколько различных заболеваний могут сопровождаться одними и теми же симптомами.

В примере с манипулятором робота кинематика определяется геометрическими уравнениями, и здесь явно прослеживается мультимодальность. Однако во многих задачах машинного обучения проявление мультимодальности, особенно в задачах с пространствами высокой размерности, может быть менее очевидным. Поэтому в качестве учебного примера рассмотрим простую искусственную задачу с наглядной визуализацией мультимодальности. Данные для этой задачи генерируются путем равномерной выборки переменной x на интервале $(0, 1)$ для получения набора значений $\{x_n\}$, а соответствующие целевые значения t_n получаются путем вычисления функции $x_n + 0,3\sin(2\pi x_n)$ и последующего добавления равномерного шума на интервале $(-0,1, 0,1)$. Для решения обратной задачи используются те же точки данных, но роли x и t меняются местами. На рис. 6.17 показаны наборы данных для прямой и обратной задач, а также результаты подгонки двухслойных нейронных сетей с шестью скрытыми элементами и одним линейным выходным элементом путем минимизации функции ошибки по методу суммы квадратов. Слева показан набор данных для простой прямой задачи, в которой красная кривая отображает результат подгонки двухслойной нейронной сети путем минимизации функции ошибки по сумме квадратов. Соответствующая обратная задача, показанная справа, получена путем замены ролей x и t . Здесь та же самая сеть, вновь обученная путем минимизации функции ошибки по сумме квадратов, дает плохой результат подгонки данных из-за мультимодальности набора данных. Сумма наименьших квадратов соответствует максимальному правдоподобию при условии гауссова распределения. Как видно, это позволяет получить хорошую модель для прямой задачи, но очень неудачную модель для обратного случая с большим несоответствием гауссовым характеристикам.

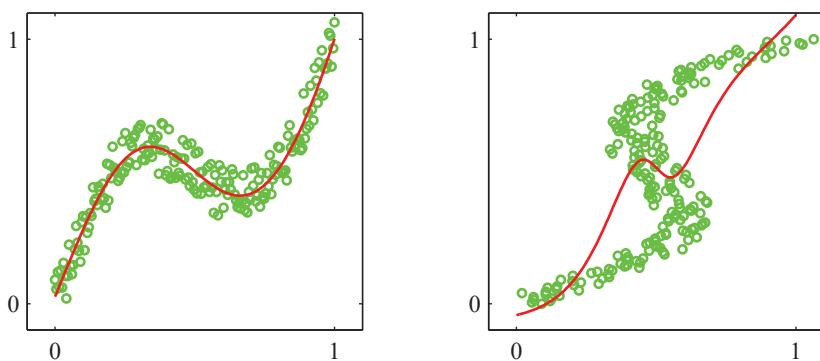


РИС. 6.17 Пример прямой и обратной задачи

6.5.2. Распределение условного смешивания

В этой связи возникает задача создания единой структуры для моделирования условных распределений вероятностей. Этого можно достичь с помощью модели смешивания для $p(t|x)$, в которой как коэффициенты смеси-

вания, так и плотности компонентов являются гибкими функциями входного вектора \mathbf{x} , что ведет к формированию сети смешанной плотности (mixtute density network). Для любого заданного значения \mathbf{x} модель смещивания обеспечивает общий формализм для моделирования произвольной функции условной плотности $p(\mathbf{t}|\mathbf{x})$. При условии, что речь идет о достаточно гибкой сети, можно получить механизм для аппроксимации произвольных условных распределений.

В данном конкретном случае рассмотрим модель для гауссовых компонентов, так что

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^K \pi_k(\mathbf{x}) \mathcal{N}(\mathbf{t}|\boldsymbol{\mu}_k(\mathbf{x}), \sigma_k^2(\mathbf{x})). \quad (6.38)$$

Здесь приводится пример *гетероскедастической модели* (*heteroscedastic model*), где дисперсия шума в данных является функцией входного вектора \mathbf{x} . Вместо гауссовых распределений для компонентов можно использовать другие распределения, например распределения Бернулли, если целевые переменные бинарны, а не непрерывны. Кроме того, в данном случае речь идет об изотропных ковариациях для компонентов, хотя сеть смешанной плотности может быть легко расширена для использования общих ковариационных матриц путем представления ковариаций с помощью факторизации Холески (Williams, 1996). Даже при изотропных компонентах условное распределение $p(\mathbf{t}|\mathbf{x})$ не предполагает факторизации относительно компонентов \mathbf{t} (в отличие от стандартной регрессионной модели с суммой квадратов), что является следствием распределения смещивания.

Теперь возьмем различные параметры модели смещивания, а именно коэффициенты смещивания $\pi_k(\mathbf{x})$, средние значения $\boldsymbol{\mu}_k(\mathbf{x})$ и дисперсии $\sigma_k^2(\mathbf{x})$, которые будут определяться выходами нейронной сети, принимающей \mathbf{x} в качестве входа. Структура этой сети смешанной плотности показана на рис. 6.18. Сеть смешанной плотности тесно связана с *моделью смеси мнений экспертов* (*mixture-of-experts model*) (Jacobs et al., 1991). Принципиальное различие заключается в том, что модель смеси мнений экспертов имеет независимые параметры для каждой компонентной модели в смеси, в то время как в сети смешанной плотности одна и та же функция используется для прогнозирования параметров всех компонентных плотностей, а также коэффициентов смещивания, и поэтому нелинейные скрытые элементы являются общими для всех функций, зависящих от входа.

Нейронная сеть на рис. 6.18 может быть, например, двухслойной с сигмоидными (\tanh) скрытыми элементами. Если в модели смещивания (6.38) имеется K компонентов и если \mathbf{t} имеет L компонентов, то сеть будет иметь K предварительных активаций выходных блоков, обозначенных как a_k^π , которые будут определять коэффициенты смещивания $\pi_k(\mathbf{x})$, а также K выходов, обозначенных как a_k^σ , которые будут определять гауссовые стандартные отклонения $\sigma_k(\mathbf{x})$, и $K \times L$ выходов, обозначенных как a_{kj}^μ , которые будут определять компоненты $\boldsymbol{\mu}_k(\mathbf{x})$ гауссовых средних значений $\boldsymbol{\mu}_k(\mathbf{x})$. Общее число вы-

ходов сети равно $(L + 2)K$, в отличие от обычных L выходов для сети, которая просто прогнозирует условные средние значения целевых переменных.

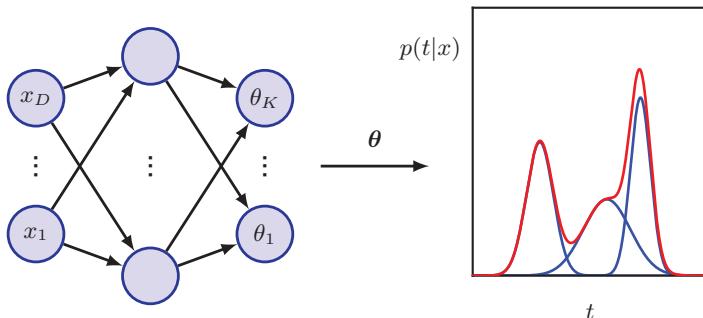


РИС. 6.18 Сеть смешанной плотности может представлять общие условные плотности вероятности $p(\mathbf{t}|\mathbf{x})$ в виде параметрической модели смешивания для распределения \mathbf{t} , при этом параметры этой модели определяются выходами нейронной сети, принимающей \mathbf{x} в качестве входного вектора

Коэффициенты смешивания должны удовлетворять ограничениям:

$$\sum_{k=1}^K \pi_k(\mathbf{x}) = 1, \quad 0 \leq \pi_k(\mathbf{x}) \leq 1, \quad (6.39)$$

что достигается с помощью набора выходов softmax:

$$\pi_k(\mathbf{x}) = \frac{\exp(a_k^\pi)}{\sum_{l=1}^K \exp(a_l^\pi)}. \quad (6.40)$$

Точно так же дисперсии должны удовлетворять условию $\sigma_k^2(\mathbf{x}) > 0$, и поэтому их можно представить в форме экспоненты соответствующих предварительных активаций сети, используя

$$\sigma_k(\mathbf{x}) = \exp(a_k^\sigma). \quad (6.41)$$

Наконец, поскольку средние значения $\mu_k(\mathbf{x})$ характеризуются вещественными компонентами, они могут быть представлены непосредственно выходами сети:

$$\mu_{kj}(\mathbf{x}) = a_{kj}^\mu, \quad (6.42)$$

при этом функции активации выходных элементов задаются тождеством $f(a) = a$.

Обучаемые параметры сети смешанной плотности состоят из вектора w весов и смещений нейронной сети, которые могут быть заданы методом максимального правдоподобия или – аналогично – путем минимизации функ-

ции ошибки, определяемой как отрицательный логарифм правдоподобия. Для независимых данных эта функция ошибки имеет вид:

$$E(\mathbf{w}) = -\sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k(\mathbf{x}_n, \mathbf{w}) \mathcal{N}(\mathbf{t}_n | \boldsymbol{\mu}_k(\mathbf{x}_n, \mathbf{w}), \sigma_k^2(\mathbf{x}_n, \mathbf{w})) \right\}, \quad (6.43)$$

где зависимость от \mathbf{w} выражена явным образом.

6.5.3. Градиентная оптимизация

Для минимизации функции ошибки потребуется вычисление производных ошибки $E(\mathbf{w})$ по компонентам \mathbf{w} . Позже будет описан способ автоматического вычисления этих производных (см. главу 8). Однако будет полезным определить подходящие выражения для производных ошибки по предварительным активациям выходных устройств в явном виде, поскольку это позволяет понять вероятностную интерпретацию этих величин. Поскольку функция ошибки (6.43) состоит из суммы членов, по одному для каждой точки обучения, производные для конкретных входных векторов \mathbf{x}_n и соответствующего целевого вектора \mathbf{t}_n можно рассматривать в явном виде. Производные от общей ошибки E получаются суммированием по всем точкам данных, либо для каждой точки данных могут быть использованы отдельные градиенты непосредственно в алгоритмах градиентной оптимизации (см. главу 7).

Здесь целесообразно ввести следующие переменные:

$$\gamma_{nk} = \gamma_k(\mathbf{t}_n | \mathbf{x}_n) = \frac{\pi_k \mathcal{N}_{nk}}{\sum_{l=1}^K \pi_l \mathcal{N}_{nl}}, \quad (6.44)$$

где \mathcal{N}_{nk} обозначает $\mathcal{N}((\mathbf{t}_n | \boldsymbol{\mu}_k(\mathbf{x}_n), \sigma_k^2(\mathbf{x}_n)))$. Эти величины имеют естественную интерпретацию как апостериорные вероятности для компонентов смешивания, в которых коэффициенты смешивания $\pi_k(\mathbf{x})$ рассматриваются как зависимые от \mathbf{x} априорные вероятности (см. упражнение 6.17).

Производные функции ошибки по предварительным активациям на выходе сети (упражнение 6.18), управляемым коэффициентами смешивания, имеют вид:

$$\frac{\partial E_n}{\partial a_k^\pi} = \pi_k - \gamma_{nk}. \quad (6.45)$$

Аналогичным образом производные по выходным предварительным активациям, определяющим компоненты средних значений (см. упражнение 6.19), имеют вид:

$$\frac{\partial E_n}{\partial a_{kl}^\mu} = \gamma_{nk} \left\{ \frac{\mu_{kl} - t_{nl}}{\sigma_k^2} \right\}. \quad (6.46)$$

Наконец, производные по выходным предварительным активациям, определяющим дисперсии компонентов (см. упражнение 6.20), имеют вид:

$$\frac{\partial E_n}{\partial a_{kl}^o} = \gamma_{nk} \left\{ L - \frac{\|\mathbf{t}_n - \boldsymbol{\mu}_k\|^2}{\sigma_k^2} \right\}. \quad (6.47)$$

6.5.4. Прогнозируемое распределение

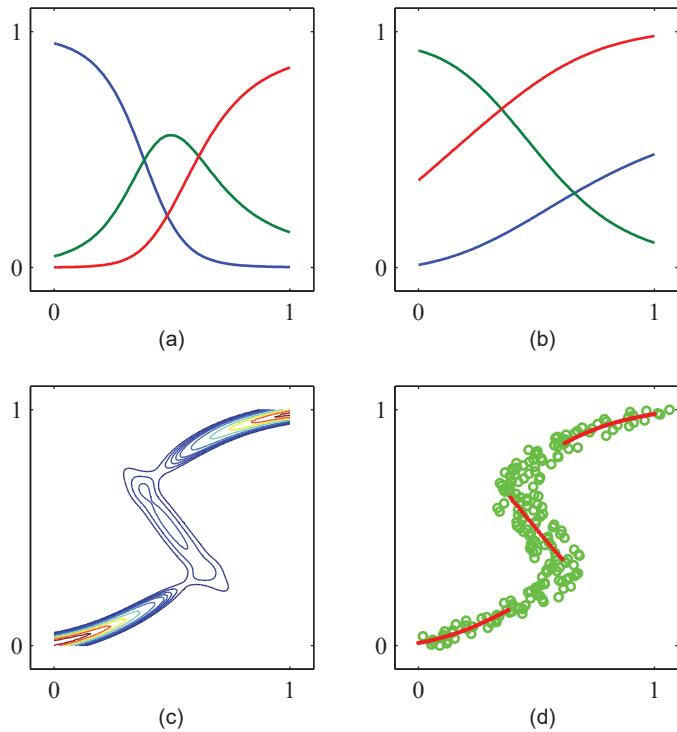
Для наглядного примера использования сети смешанной плотности вернемся к искусственному примеру обратной задачи на рис. 6.17. Графики коэффициентов смещивания $\pi_k(x)$, средних значений $\mu_k(x)$ и контуров условной плотности, соответствующих $p(t|x)$, показаны на рис. 6.19. Выходы нейронной сети и, следовательно, параметры модели смещивания обязательно являются непрерывными однозначными функциями входных переменных. Однако из рис. 6.19c следует, что модель способна выдавать условную плотность, которая является унимодальной для одних значений x и тримодальной для других значений, за счет модуляции амплитуд компонентов смещивания $\pi_k(x)$. ((a) График коэффициентов смещивания $\pi_k(x)$ как функции от x для трех компонентов смещивания в сети смешанной плотности, обученной на данных рис. 6.17. Модель состоит из трех гауссовых компонентов и использует двухслойную нейронную сеть с пятью сигмоидными элементами \tanh в скрытом слое и девятью выходами (соответствующими трем средним значениям, трем вариациям гауссовых компонентов и трем коэффициентам смещивания). При малых и больших значениях x , когда плотность условной вероятности целевых данных унимодальна, только одна из гауссовых компонент имеет высокое значение для своей априорной вероятности, тогда как при промежуточных значениях x , когда условная плотность тримодальна, все три коэффициента смещивания имеют сопоставимые значения. (b) Графики средних значений $\mu_k(x)$ с использованием той же цветовой кодировки, что и для коэффициентов смещивания. (c) График контуров соответствующей условной плотности вероятности целевых данных для той же сети плотности смещивания. (d) График приближенной условной моды (показан красными точками) условной плотности.)

После завершения обучения сети смешанной плотности она способна предсказать функцию условной плотности целевых данных для любого заданного значения входного вектора. Эта условная плотность представляет собой полное описание генератора данных для решения задачи предсказания значения выходного вектора. На основе этой функции плотности можно вычислить более конкретные величины, которые могут представлять интерес при решении различных задач. Одной из самых простых является среднее значение, соответствующее условному среднему значению целевых данных, которое задается как

$$\mathbb{E}[\mathbf{t}|\mathbf{x}] = \int \mathbf{t} p(\mathbf{t}|\mathbf{x}) d\mathbf{t} = \sum_{k=1}^K \pi_k(\mathbf{x}) \boldsymbol{\mu}_k(\mathbf{x}), \quad (6.48)$$

где используется (6.38). Поскольку стандартная сеть, обученная по методу наименьших квадратов, аппроксимирует условное среднее значение, получается, что сеть смешанной плотности может получить стандартный результат по методу наименьших квадратов в качестве частного случая. Разумеется, как уже было отмечено, в случае мультимодального распределения условное среднее значение имеет ограниченное значение.

РИС. 6.19 Пример использования сети смешанной плотности



Аналогичным образом можно оценить дисперсию функции плотности относительно условного среднего значения (см. упражнение 6.21), что дает

$$s^2(\mathbf{x}) = \mathbb{E}[||\mathbf{t} - \mathbb{E}[\mathbf{t} | \mathbf{x}]||^2 | \mathbf{x}] \quad (6.49)$$

$$= \sum_{k=1}^K \pi_k(\mathbf{x}) \left\{ \sigma_k^2(\mathbf{x}) + \left\| \boldsymbol{\mu}_k(\mathbf{x}) - \sum_{l=1}^K \pi_l(\mathbf{x}) \boldsymbol{\mu}_l(\mathbf{x}) \right\|^2 \right\}, \quad (6.50)$$

где используются (6.38) и (6.48). Это более общий результат, чем соответствующий результат наименьших квадратов, поскольку дисперсия является функцией от \mathbf{x} .

Ранее уже было показано, что для мультимодальных распределений условное среднее значение может давать плохое представление о данных. Например, при управлении простейшим манипулятором робота, показанным на рис. 6.16, для достижения желаемого положения конечного механизма

необходимо выбрать одно из двух возможных значений угла поворота сустава, но среднее значение двух решений само по себе не является решением. В таких случаях большее значение может иметь значение условной моды.

Поскольку условная мода для сети смешанной плотности не имеет простого аналитического решения, необходима численная итерация. Простой альтернативой этому является взятие среднего значения наиболее вероятного компонента (т. е. компонента с наибольшим коэффициентом смещивания) при каждом значении x . Это показано для искусственного набора данных на рис. 6.19d.

Упражнения

- 6.1** (★★★) Используйте результат (2.126), чтобы вывести выражение для площади поверхности S_D и объема V_D гиперсферы единичного радиуса в D измерениях. Для этого нужно учесть следующий результат, который получается при преобразовании из декартовых координат в полярные:

$$\prod_{i=1}^D \int_{-\infty}^{\infty} e^{-x_i^2} dx_i = S_D \int_0^{\infty} e^{-r^2} r^{D-1} dr. \quad (6.51)$$

Используя гамма-функцию, определяемую как

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt \quad (6.52)$$

вместе с (2.126), оцените обе стороны этого уравнения и, следовательно, докажите, что

$$S_D = \frac{2\pi^{D/2}}{\Gamma(D/2)}. \quad (6.53)$$

Далее, интегрируя по радиусу от 0 до 1, докажите, что объем единичной гиперсферы в D измерениях задается как

$$V_D = \frac{S_D}{D}. \quad (6.54)$$

Наконец, используя результаты $\Gamma(1) = 1$ и $\Gamma(3/2) = \sqrt{\pi}/2$, докажите, что (6.53) и (6.54) сводятся к обычным выражениям для $D = 2$ и $D = 3$.

- 6.2** (★★★) Рассмотрим гиперсферу радиуса a в D измерениях вместе с концентрическим гиперкубом со стороной $2a$ такого размера, чтобы гиперсфера касалась гиперкуба в центрах каждой из его сторон. Используя результаты упражнения 6.1, докажите, что отношение объема гиперсферы к объему куба равно

$$\frac{\text{объем гиперсферы}}{\text{объем куба}} = \frac{\pi^{D/2}}{D 2^{D-1} \Gamma(D/2)}. \quad (6.55)$$

Теперь воспользуйтесь формулой Стирлинга в виде

$$\Gamma(x+1) \simeq (2\pi)^{1/2} e^{-x} x^{x+1/2}, \quad (6.56)$$

которая справедлива для $x \gg 1$, чтобы доказать, что при $D \rightarrow \infty$ соотношение (6.55) обращается в ноль.

Также необходимо доказать, что расстояние от центра гиперкуба до одного из углов, деленное на расстояние перпендикуляра до одной из сторон, равно \sqrt{D} , и, следовательно, при $D \rightarrow \infty$ оно становится равным ∞ . Из этих результатов становится понятно, что в пространстве высокой размерности большая часть объема куба сосредоточена в большом количестве углов, которые сами превращаются в очень длинные «пики»!

- 6.3** (★★★) В этом упражнении исследуется поведение гауссова распределения в пространствах высокой размерности. Рассмотрим гауссово распределение в D измерениях, заданное как

$$p(\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right). \quad (6.57)$$

Требуется найти плотность как функцию радиуса в полярных координатах, в которых переменные направления проинтегрированы. Для этого нужно показать, что интеграл плотности вероятности по тонкой оболочке радиуса r и толщины ϵ , где $\epsilon \ll 1$, задается $p(r)\epsilon$, где

$$p(r) = \frac{S_D r^{D-1}}{(2\pi\sigma^2)^{D/2}} \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad (6.58)$$

и S_D – это площадь поверхности единичной гиперсферы в D измерениях. Докажите, что функция $p(r)$ имеет единственную стационарную точку, расположенную при больших D в точке $\hat{r} \simeq \sqrt{D}\sigma$. Рассматривая $p(\hat{r} + \epsilon)$, где $\epsilon \gg \hat{r}$, докажите, что для больших D

$$p(\hat{r} + \epsilon) = p(\hat{r}) \exp\left(-\frac{3\epsilon^2}{2\sigma^2}\right), \quad (6.59)$$

что свидетельствует о том, что \hat{r} является максимумом радиальной плотности вероятности, а также о том, что $p(r)$ экспоненциально убывает от своего максимума в \hat{r} с масштабом длины σ . Ранее было отмечено, что $\sigma\hat{r}$ для больших D , и поэтому можно видеть, что большая часть массы вероятности сосредоточена в тонкой оболочке большого радиуса. Наконец, докажите, что плотность вероятности $p(\mathbf{x})$ в начале координат больше в $\exp(D/2)$, чем на радиусе \hat{r} . Таким образом, можно видеть, что большая часть массы вероятности в гауссовом распределении высокой разрядности находится на ином радиусе от области высокой плотности вероятности.

- 6.4** (**) Рассмотрим двухслойную сетевую функцию вида (6.11), в которой нелинейные функции активации $h(\cdot)$ скрытого элемента задаются логистическими сигмоидными функциями вида

$$\sigma(a) = \{1 + \exp(-a)\}^{-1}. \quad (6.60)$$

Докажите, что существует эквивалентная сеть, которая вычисляет точно такую же функцию, но с функциями активации скрытых элементов, заданными функцией $\tanh(a)$, где функция \tanh определена в (6.14). Подсказка: сначала найдите связь между $\sigma(a)$ и $\tanh(a)$, а затем докажите, что параметры двух сетей различаются линейными преобразованиями.

- 6.5** (**) Функция активации *swish* (Ramachandran, Zoph and Le, 2017) определяется как

$$h(x) = x\sigma(\beta x), \quad (6.61)$$

где $\sigma(x)$ – это логистическая сигмоидная функция активации, определенная в (6.13). При использовании в нейронной сети β можно рассматривать в качестве обучаемого параметра. Начертите или постройте с помощью программного обеспечения графики функции активации *swish*, а также ее первой производной для $\beta = 0,1$, $\beta = 1,0$ и $\beta = 10$. Докажите, что при $\beta \rightarrow \infty$ функция *swish* превращается в функцию ReLU.

- 6.6** (*) в (5.72) было показано, что производная логистической сигмоидной функции активации может быть выражена в виде значения самой функции. Выведите соответствующий результат для функции активации \tanh , определенной в (6.14).
- 6.7** (**) Докажите, что функция активации *softplus* $\zeta(a)$, заданная в (6.16), удовлетворяет свойствам:

$$\zeta(a) - \zeta(-a) = a, \quad (6.62)$$

$$\ln \sigma(a) = -\zeta(-a), \quad (6.63)$$

$$\frac{d\zeta(a)}{da} = \sigma(a), \quad (6.64)$$

$$\zeta^{-1}(a) = \ln(\exp(a) - 1), \quad (6.65)$$

где $\sigma(a)$ – это логистическая сигмоидная функция активации, заданная в (6.13).

- 6.8** (*) Докажите, что минимизация функции ошибки (6.25) относительно дисперсии σ^2 дает результат (6.27).
- 6.9** (*) Докажите, что максимизация функции правдоподобия при условном распределении (6.28) для нейронной сети с несколькими выходами эквивалентна минимизации функции ошибки по сумме квадратов (6.29).

Также докажите, что дисперсия шума, минимизирующая эту функцию ошибки, задается в (6.30).

- 6.10** (**) Рассмотрим задачу регрессии с несколькими целевыми переменными, в которой предполагается, что распределение целевых переменных, обусловленное входным вектором \mathbf{x} , является гауссовым в виде

$$p(\mathbf{t} | \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{t} | \mathbf{y}(\mathbf{x}, \mathbf{w}), \Sigma), \quad (6.66)$$

где $\mathbf{y}(\mathbf{x}, \mathbf{w})$ – это выход нейронной сети с входным вектором \mathbf{x} и весовым вектором \mathbf{w} , а Σ – это ковариация предполагаемого гауссова шума для целей. Учитывая набор независимых наблюдений \mathbf{x} и \mathbf{t} , запишите функцию ошибки, которую необходимо минимизировать для нахождения решения максимального правдоподобия для \mathbf{w} , если считать, что Σ фиксирована и известна. Затем предположим, что Σ также должна быть определена из данных. Запишите выражение для решения максимального правдоподобия для Σ . Обратите внимание, что оптимизация \mathbf{w} и Σ теперь взаимосвязана, в отличие от случая независимых целевых переменных, рассмотренного в разделе 6.4.1.

- 6.11** (**) Рассмотрим задачу бинарной классификации, в которой целевыми значениями являются $t \in \{0, 1\}$, а выход сети $y(\mathbf{x}, \mathbf{w})$ соответствует $p(t = 1 | \mathbf{x})$, и предположим, что существует вероятность ошибочной установки метки класса в обучающей точке данных. При условии i.i.d. данных запишите функцию ошибки, соответствующую отрицательному логарифму правдоподобия. Убедитесь, что функция ошибки (6.33) получена при $\epsilon = 0$. Обратите внимание, что эта функция ошибки делает модель устойчивой к неверно заданным меткам, в отличие от обычной функции ошибки перекрестной энтропии.

- 6.12** (**) Функция ошибки (6.33) для задач бинарной классификации была получена для сети с логистической сигмоидной функцией активации на выходе, так что $0 \leq y(\mathbf{x}, \mathbf{w}) \leq 1$, и данными, имеющими целевые значения $t \in \{0, 1\}$. Выведите соответствующую функцию ошибки, если речь идет о сети с выходом $-1 \leq y(\mathbf{x}, \mathbf{w}) \leq 1$ и целевыми значениями $t = 1$ для класса C_1 и $t = -1$ для класса C_2 . Каким будет подходящий выбор функции активации выходного элемента?

- 6.13** (*) Докажите, что максимизация правдоподобия для модели нейронной сети с несколькими классами, в которой выходы сети интерпретируются как $y_k(\mathbf{x}, \mathbf{w}) = p(t_k = 1 | \mathbf{x})$, эквивалентна минимизации функции ошибки перекрестной энтропии (6.36).

- 6.14** (*) Докажите, что производная функции ошибки (6.33) по отношению к предварительной активации a_k для выходного элемента, имеющего логистическую сигмоидную функцию активации $y_k = \sigma(a_k)$, в которой $\sigma(a)$ определяется в (6.13), удовлетворяет условию (6.31).

- 6.15** (*) Докажите, что производная функции ошибки (6.36) по отношению к предварительной активации a_k для выходных элементов с функцией активации softmax (6.37) удовлетворяет (6.31).
- 6.16** (**) Запишите пару уравнений, выражающих декартовы координаты (x_1, x_2) манипулятора робота, показанного на рис. 6.16, в виде углов сочленений θ_1 и θ_2 и длин звеньев L_1 и L_2 . Начало системы координат задано точкой крепления нижней части манипулятора. Эти уравнения определяют кинематику движения манипулятора робота вперед.
- 6.17** (**) Докажите, что переменная γ_{nk} , определяемая в (6.44), может рассматриваться как апостериорная вероятность $p(k|\mathbf{t})$ для компонентов распределения смешивания (6.38), где коэффициенты смешивания $\pi_k(\mathbf{x})$ рассматриваются в качестве зависящих от \mathbf{x} априорных вероятностей $p(k)$.
- 6.18** (**) Выведите результат (6.45) для производной функции ошибки по отношению к выходным предварительным активациям сети, управляемым коэффициентами смешивания в сети смешанной плотности.
- 6.19** (**) Выведите результат (6.46) для производной функции ошибки относительно предварительных активаций на выходе сети, определяющих средние значения компонентов в сети смешанной плотности.
- 6.20** (**) Выведите результат (6.47) для производной функции ошибки относительно предварительных активаций на выходе сети, которые определяют дисперсию компонентов в сети смешанной плотности.
- 6.21** (***) Убедитесь в справедливости результатов (6.48) и (6.50) для условного среднего и дисперсии модели сети смешанной плотности.

Глава 7

Градиентный спуск

В предыдущей главе было показано, что нейронные сети являются очень широким и гибким классом функций и в принципе способны аппроксимировать любую желаемую функцию с произвольно высокой точностью при достаточно большом количестве скрытых элементов. Более того, выяснилось, что глубокие нейронные сети могут кодировать индуктивные смещения, соответствующие иерархическим представлениям, которые представляют большую ценность для широкого круга практических задач. Теперь рассмотрим задачу поиска подходящего значения параметров сети (весов и смещений) на основе набора обучающих данных.

Как и в случае с моделями регрессии и классификации, рассмотренными в предыдущих главах, выбор параметров модели осуществляется путем оптимизации функции ошибки. Ранее было показано, как определить подходящую функцию ошибки для конкретного приложения с помощью метода максимального правдоподобия (см. раздел 6.4). Несмотря на то что функцию ошибки, в принципе, можно минимизировать численно с помощью серии прямых оценок функции ошибки, такой способ оказался малоэффективным. Вместо этого обратимся к другой базовой концепции глубокого обучения, которая заключается в том, что оптимизация функции ошибки может быть выполнена гораздо эффективнее за счет использования градиентной информации, т. е. за счет оценки производных функции ошибки по параметрам сети. Именно поэтому было принято решение позаботиться о том, чтобы представляемая нейронной сетью функция была дифференцируемой. Точно так же и функция ошибки должна быть дифференцируемой.

Для эффективного вычисления требуемых производных функции ошибки по каждому из параметров сети используется метод *обратного распространения* (*backpropagation*) (см. главу 8), который заключается в последовательных вычислениях в обратном направлении через сеть, что подобно прямому потоку вычислений функций при оценке выходов сети.

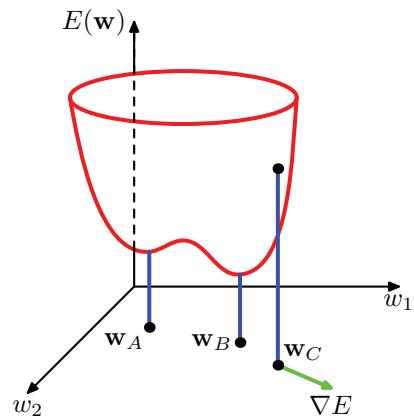
Несмотря на то что для определения функции ошибки используется показатель правдоподобия, целью оптимизации функции ошибки в нейронной сети является достижение эффективного обобщения на основе тестовых данных. В классической статистике максимальное правдоподобие используется для подгонки параметрической модели к конечному набору данных, где количество точек данных обычно намного превышает количество параметров модели. Оптимальным решением является максимальное значе-

ние функции правдоподобия, а непосредственный интерес представляют найденные значения подогнанных параметров. В отличие от этих методов современное глубокое обучение работает с очень сложными моделями, содержащими огромное количество обучаемых параметров (см. раздел 9.3.2), и целью никогда не является лишь простая точная оптимизация. Напротив, свойства и особенности работы самого алгоритма обучения, а также различные методы регуляризации (см. главу 9) играют важную роль в определении того, насколько хорошо это решение будет обобщаться на новых данных.

7.1. Поверхности ошибок

В процессе обучения нашей целью является поиск таких значений весов и смещений нейронной сети, которые бы обеспечивали ей способность делать эффективные прогнозы. Для удобства эти параметры будут сгруппированы в единый вектор \mathbf{w} , и для оптимизации \mathbf{w} будет использоваться определенная функция ошибки $E(\mathbf{w})$. На данном этапе полезно иметь геометрическое представление о функции ошибки, которую можно представить как поверхность над «пространством весов», как показано на рис. 7.1. Здесь точка \mathbf{w}_A – это локальный минимум, \mathbf{w}_B – это глобальный минимум, так что $E(\mathbf{w}_A) > E(\mathbf{w}_B)$. В любой точке \mathbf{w}_C локальный градиент поверхности ошибки задается вектором ∇E .

РИС. 7.1 Геометрическое представление функции ошибки $E(\mathbf{w})$ в виде поверхности над пространством весов



Прежде всего следует отметить, что если сделать небольшой шаг в весовом пространстве от \mathbf{w} до $\mathbf{w} + \delta\mathbf{w}$, то изменение функции ошибки будет выглядеть как

$$\delta E \simeq \delta\mathbf{w}^T \nabla E(\mathbf{w}), \quad (7.1)$$

где вектор $\nabla E(\mathbf{w})$ направлен в сторону наибольшего прироста функции ошибки. При условии, что ошибка $E(\mathbf{w})$ является плавной непрерывной функцией

от \mathbf{w} , ее наименьшее значение будет иметь место в точке весового пространства, где градиент функции ошибки исчезает, так что

$$\nabla E(\mathbf{w}) = 0, \quad (7.2)$$

поскольку в противном случае можно было бы сделать небольшой шаг в направлении $-\nabla E(\mathbf{w})$ и, таким образом, еще больше уменьшить ошибку. Точки, в которых градиент исчезает, называются стационарными точками (stationary points). Они также могут быть классифицированы на минимумы, максимумы и седловые точки (см. раздел 7.1.1).

Целью является поиск такого вектора \mathbf{w} , при котором $E(\mathbf{w})$ принимает наименьшее значение. Однако функция ошибки обычно имеет весьма нелинейную зависимость от весов и параметров смещения, и по этой причине в пространстве весов будет много точек, в которых градиент исчезает (или численно очень мал). Действительно, для любой точки \mathbf{w} , являющейся локальным минимумом, в пространстве весов, как правило, существуют другие точки, которые являются эквивалентными минимумами. Например, в двухслойной сети по типу показанной на рис. 6.9 с M скрытыми элементами каждая точка в пространстве весов является членом семейства из $M!2^M$ эквивалентных точек (см. раздел 6.2.4).

Кроме того, может существовать множество неэквивалентных стационарных точек и, в частности, множество неэквивалентных минимумов. Считается, что минимум, соответствующий наименьшему значению функции ошибки во всем \mathbf{w} -пространстве, является *глобальным минимумом* (*global minimum*). Любые другие минимумы, соответствующие большим значениям функции ошибки, считаются *локальными минимумами* (*local minima*). Поверхности ошибок для глубоких нейронных сетей могут быть очень сложными, и ранее существовало мнение, что градиентные методы могут угодить в ловушку плохих локальных минимумов. На практике этого не происходит, и большие сети могут достигать решений с одинаковой эффективностью при различных начальных условиях (см. раздел 9.3.2).

7.1.1. Локальная квадратичная аппроксимация

Представление о сути задачи оптимизации и о различных методах ее решения можно получить при изучении локальной квадратичной аппроксимации функции ошибки. Разложение Тейлора для $E(\mathbf{w})$ вблизи некоторой точки $\hat{\mathbf{w}}$ в весовом пространстве имеет вид:

$$E(\mathbf{w}) \approx E(\hat{\mathbf{w}}) + (\mathbf{w} - \hat{\mathbf{w}})^T \mathbf{b} + \frac{1}{2}(\mathbf{w} - \hat{\mathbf{w}})^T \mathbf{H}(\mathbf{w} - \hat{\mathbf{w}}), \quad (7.3)$$

где кубические и более высокие члены опущены. Здесь \mathbf{b} определяется как градиент E , рассчитанный в точке $\hat{\mathbf{w}}$:

$$\mathbf{b} \equiv \nabla E|_{\mathbf{w}=\hat{\mathbf{w}}}. \quad (7.4)$$

Матрица Гессе (Hessian) определяется как соответствующая матрица вторых производных:

$$\mathbf{H}(\hat{\mathbf{w}}) = \nabla \nabla E(\mathbf{w})|_{\mathbf{w}=\hat{\mathbf{w}}}. \quad (7.5)$$

Если в сети всего W весов и смещений, то \mathbf{w} и \mathbf{b} имеют длину W , а \mathbf{H} имеет размерность $W \times W$. Из (7.3) соответствующая локальная аппроксимация градиента задается как

$$\nabla E(\mathbf{w}) = \mathbf{b} + \mathbf{H}(\mathbf{w} - \hat{\mathbf{w}}). \quad (7.6)$$

Для точек \mathbf{w} , достаточно близких к $\hat{\mathbf{w}}$, эти выражения дадут разумные приближения для ошибки и ее градиента.

Рассмотрим частный случай локальной квадратичной аппроксимации в точке \mathbf{w}^* , которая является минимумом функции ошибки. В этом случае линейный член отсутствует, так как $\nabla E = 0$ при \mathbf{w}^* , и (7.3) приобретает вид:

$$E(\mathbf{w}) = E(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w} - \mathbf{w}^*), \quad (7.7)$$

где гессиан \mathbf{H} оценивается в точке \mathbf{w}^* . Для геометрической интерпретации рассмотрим уравнение с собственными значениями для матрицы Гессе:

$$\mathbf{H}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad (7.8)$$

где собственные векторы \mathbf{u}_i образуют полный ортонормальный набор (см. приложение А), так что

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}. \quad (7.9)$$

Теперь разложим $(\mathbf{w} - \mathbf{w}^*)$ как линейную комбинацию собственных векторов в виде

$$\mathbf{w} - \mathbf{w}^* = \sum_i \alpha_i \mathbf{u}_i. \quad (7.10)$$

Это можно рассматривать как преобразование системы координат, при котором начало координат переводится в точку \mathbf{w}^* , а оси поворачиваются так, чтобы совместить их с собственными векторами через ортогональную матрицу (см. приложение А) со столбцами $\{\mathbf{u}_1, \dots, \mathbf{u}_W\}$. Подставив (7.10) в (7.7) и используя (7.8) и (7.9), функцию ошибки можно записать в виде (см. упражнение 7.1)

$$E(\mathbf{w}) = E(\mathbf{w}^*) + \frac{1}{2} \sum_i \lambda_i \alpha_i^2. \quad (7.11)$$

Предположим, что нужно задать все $\alpha_i = 0$ для $i \neq j$, а затем варьировать α_j , что соответствует перемещению \mathbf{w} от \mathbf{w}^* в направлении \mathbf{u}_j . Из (7.11) видно, что функция ошибки будет увеличиваться, если соответствующее

собственное значение λ , положительно, и уменьшаться, если оно отрицательно. Если все собственные значения положительны, то w^* соответствует локальному минимуму функции ошибки, а если все они отрицательны, то w^* соответствует локальному максимуму. При наличии смеси положительных и отрицательных собственных значений w^* представляет собой седловую точку.

Матрица H считается положительно определенной (positive definite) тогда и только тогда, когда

$$v^T H v > 0 \text{ для всех } v. \quad (7.12)$$

Поскольку собственные векторы $\{u_i\}$ образуют полный набор, произвольный вектор v может быть записан в виде

$$v = \sum_i c_i u_i. \quad (7.13)$$

Из (7.8) и (7.9) следует, что

$$v^T H v = \sum_i c_i^2 \lambda_i, \quad (7.14)$$

и поэтому H будет положительно определенной тогда и только тогда, когда все ее собственные значения положительны (см. упражнение 7.2). Таким образом, необходимым и достаточным условием того, что w^* является локальным минимумом, является то, что градиент функции ошибки должен исчезать в w^* и матрица Гессе, оцененная в w^* , должна быть положительно определенной (см. упражнение 7.3). Итак, в новой системе координат, базисные векторы которой задаются собственными векторами $\{u_i\}$, контуры постоянной $E(w)$ представляют собой выровненные по оси эллипсы с центром в начале координат (см. упражнение 7.6), а длины обратно пропорциональны квадратным корням соответствующих собственных векторов λ_i , как показано на рис. 7.2.

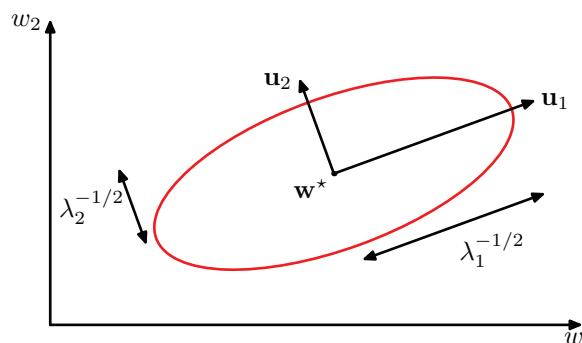


РИС. 7.2 В окрестности минимума w^* функция ошибки может быть аппроксимирована квадратично

7.2. Оптимизация методом градиентного спуска

Существует мало надежд найти аналитическое решение уравнения $\nabla E(\mathbf{w}) = 0$ для такой сложной функции ошибки, как функция, заданная нейронной сетью, поэтому в данном случае приходится прибегать к итерационным численным методам. Оптимизация непрерывных нелинейных функций широко изучена, существует большое количество литературы о способах ее эффективного решения. Большинство методов предполагает выбор некоторого начального значения $\mathbf{w}^{(0)}$ для весового вектора и последующее перемещение по весовому пространству в последовательности шагов вида

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} + \Delta\mathbf{w}^{(\tau-1)}, \quad (7.15)$$

где τ обозначает шаг итерации. Разные алгоритмы предполагают различные варианты обновления весового вектора $\Delta\mathbf{w}^{(\tau)}$.

Из-за сложной формы поверхности ошибки для всех нейронных сетей, кроме самых простых, полученное в результате решение будет зависеть, в частности, от конкретного выбора начальных значений параметров $\mathbf{w}^{(0)}$. Чтобы найти действительно хорошее решение, может потребоваться многократное выполнение градиентного алгоритма, каждый раз с использованием различных, случайно выбранных начальных точек, и последующее сравнение полученных результатов на независимом валидационном множестве.

7.2.1. Использование градиентной информации

Для эффективного определения градиента функции ошибки глубокой нейронной сети применяется техника обратного распространения ошибки (error backpropagation) (см. главу 8), и использование этой информации о градиенте может существенно повысить скорость обучения сети. И вот почему.

В квадратичной аппроксимации функции ошибки, заданной в (7.3), поверхность ошибки определяется величинами \mathbf{b} и \mathbf{H} , которые в сумме включают $W(W + 3)/2$ независимых элементов (поскольку матрица \mathbf{H} симметрична) (см. упражнение 7.7 3), где W – это размерность \mathbf{w} (т. е. общее число обучаемых параметров в сети). Таким образом, местоположение минимума этой квадратичной аппроксимации зависит от $\mathcal{O}(W^2)$ параметров, и не следует надеяться, что удастся найти минимум до тех пор, пока не будет собрано $\mathcal{O}(W^2)$ независимых фрагментов информации. Если не использовать информацию о градиенте, то придется выполнить $\mathcal{O}(W^2)$ оценок функций, каждая из которых потребует $\mathcal{O}(W)$ этапов. Получается, что объем необходимых вычислительных усилий для поиска минимума при таком подходе составит $\mathcal{O}(W^3)$.

Сравним это с алгоритмом, использующим информацию о градиенте. Поскольку ∇E – это вектор длины W , каждая оценка ∇E приносит W единиц

информации, и поэтому можно надеяться найти минимум функции за $\mathcal{O}(W)$ градиентных оценок (см. главу 8). Как будет показано далее, при использовании обратного распространения ошибки каждая такая оценка занимает всего $\mathcal{O}(W)$ этапов, поэтому минимум теперь может быть найден за $\mathcal{O}(W^2)$ этапов. Хотя квадратичная аппроксимация справедлива только в окрестностях минимума, выигрыш в эффективности является общим. По этой причине использование градиентной информации лежит в основе всех практических алгоритмов обучения нейронных сетей.

7.2.2. Пакетный градиентный спуск

Простейший подход к использованию градиентной информации заключается в выборе обновления веса в (7.15) таким образом, чтобы был небольшой шаг в направлении отрицательного градиента, так что

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \nabla E(\mathbf{w}^{(t-1)}), \quad (7.16)$$

где параметр $\eta > 0$ известен как *скорость обучения* (*learning rate*). После каждого такого обновления градиент заново оценивается для нового весового вектора $\mathbf{w}^{(t+1)}$, и затем процесс повторяется. На каждом шаге весовой вектор перемещается в направлении наибольшей скорости уменьшения функции ошибки, поэтому такой подход известен как *градиентный спуск* (*gradient descent*) или *наискорейший спуск* (*steepest descent*). Обратите внимание, что функция ошибки определяется относительно обучающего набора, поэтому для оценки ∇E на каждом шаге необходимо обрабатывать весь обучающий набор. Приемы, использующие весь набор данных разом, называются *пакетными методами* (*batch methods*).

7.2.3. Стохастический градиентный спуск

Эффективность методов глубокого обучения значительно возрастает при использовании очень больших наборов данных. Однако пакетные методы могут стать крайне неэффективными, если в обучающем наборе данных содержится слишком много точек, поскольку каждая функция ошибки или оценка градиента требуют обработки всего набора данных. В поисках более эффективного подхода следует обратить внимание на то, что функции ошибок, основанные на максимальном правдоподобии для набора независимых наблюдений, состоят из суммы членов, по одному для каждой точки данных:

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}). \quad (7.17)$$

Наиболее широко используемые алгоритмы обучения для больших наборов данных основаны на последовательной версии градиентного спуска, известной как *стохастический градиентный спуск* (*stochastic gradient descent*,

SGD) (Bottou, 2010), который обновляет весовой вектор на основе одной точки данных за один раз, так что

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \nabla E_n(\mathbf{w}^{(t-1)}). \quad (7.18)$$

Это обновление происходит в циклическом режиме. Полный проход по всему обучающему набору называется *эпохой обучения* (*training epoch*). Эта техника также известна как *онлайн-градиентный спуск* (*online gradient descent*), особенно если данные поступают в виде непрерывного потока новых точек данных. Стохастический градиентный спуск кратко описан ниже в алгоритме 7.1.

АЛГОРИТМ 7.1 Стохастический градиентный спуск

Input: обучающий набор точек данных в интервале $n \in \{1, \dots, N\}$

Функция ошибки для каждой точки данных $E_n(\mathbf{w})$

Параметр скорости обучения η

Начальный вектор весов \mathbf{w}

Output: итоговый вектор весов \mathbf{w}

```

 $n \leftarrow 1$ 
repeat
     $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_n(\mathbf{w})$  // обновление вектора весов
     $n \leftarrow n + 1 \text{ mod } N$  // итерация по данным
until convergence (до сходимости)
return  $\mathbf{w}$ 

```

Еще одно преимущество стохастического градиентного спуска, по сравнению с пакетным градиентным спуском, заключается в том, что он гораздо эффективнее справляется с избыточностью данных. Чтобы убедиться в этом, рассмотрим экстремальный пример, где набор данных удваивается за счет дублирования каждой точки данных. Обратите внимание, что это просто умножает функцию ошибки в 2 раза, что эквивалентно использованию исходной функции ошибки, если для компенсации скорректировать значение скорости обучения. Пакетные методы потребуют вдвое больше вычислительных усилий для оценки градиента функции ошибки, в то время как стохастический градиентный спуск не изменится. Еще одним свойством стохастического градиентного спуска является возможность выхода из локальных минимумов, поскольку стационарная точка относительно функции ошибки для всего набора данных, как правило, не будет стационарной точкой для каждой точки данных в отдельности.

7.2.4. Мини-батчи

Недостатком стохастического градиентного спуска является то, что вычисление градиента функции ошибки по одной точке данных дает очень зашумленную оценку градиента функции ошибки, вычисленной по всему набору

данных. Можно воспользоваться промежуточным подходом, при котором для оценки градиента на каждой итерации используется небольшое подмножество точек данных, которое называется *мини-батч* (*мини-пакет*, *mini-batch*). При определении оптимального размера мини-батча следует учитывать, что ошибка при вычислении среднего значения по N выборкам равна σ/\sqrt{N} , где σ – это стандартное отклонение распределения, генерирующего данные (см. упражнение 7.8). Это говорит о том, что увеличение размера партии обеспечивает все меньшую отдачу в оценке истинного градиента. Если увеличить размер мини-батча в 100 раз, то ошибка уменьшится только в 10 раз. Еще одним соображением при выборе размера мини-батча является желание эффективно использовать аппаратную архитектуру, на которой выполняется код. Например, на некоторых аппаратных платформах хорошо работают мини-батчи, размер которых представляет собой степень 2 (например, 64, 128, 256 и т. д.).

Важным моментом при использовании мини-батчей является необходимость случайного выбора точек данных из всего набора данных, поскольку в необработанных наборах данных могут существовать корреляции между последовательными точками данных, обусловленные способом сбора информации (например, если точки данных были упорядочены в алфавитном порядке или по дате). Зачастую такая проблема решается путем случайной перетасовки всего набора данных и последующего составления мини-батчей в виде последовательных блоков данных. Набор данных также можно перетасовывать между итерациями, чтобы каждый мини-батч с большой вероятностью не был использован ранее, что поможет избежать локальных минимумов. Вариант стохастического градиентного спуска с мини-батчами обобщен в алгоритме 7.2. Обратите внимание, что алгоритм обучения часто называют «стохастическим градиентным спуском», даже если используются мини-батчи.

АЛГОРИТМ 7.2 Мини-батч стохастического градиентного спуска.

Input: обучающий набор точек данных в интервале $n \in \{1, \dots, N\}$

Размер батча B

Функция ошибки для мини-батча $E_{n:n+B-1}(\mathbf{w})$

Параметр скорости обучения η

Начальный вектор весов \mathbf{w}

Output: итоговый вектор весов \mathbf{w}

```

 $n \leftarrow 1$ 
repeat
     $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_{n:n+B-1}(\mathbf{w})$  // обновление вектора весов
     $n \leftarrow n + B$ 
    if  $n > N$  then
        перетасовка данных
         $n \leftarrow 1$ 
    end if
until convergence (до сходимости)
return  $\mathbf{w}$ 

```

7.2.5. Инициализация параметров

При работе с итеративными алгоритмами, такими как градиентный спуск, требуется выбрать некоторое начальное значение для обучаемых параметров. Выбор конкретной инициализации может существенно повлиять на продолжительность процесса обучения и обобщающие характеристики обучаемой сети. К сожалению, теоретических рекомендаций по выбору стратегии инициализации существует сравнительно немного.

Однако одним из ключевых соображений является вопрос о *нарушении симметрии* (*symmetry breaking*). Рассмотрим набор скрытых элементов или выходных элементов, которые получают одинаковые входы. Если все параметры инициализировать одним и тем же значением, например установить их в ноль, то параметры этих блоков будут обновляться синхронно, и каждый из них будет вычислять одну и ту же функцию, а значит, будет избыточным. Эту проблему можно решить путем случайной инициализации параметров из некоторого распределения, что позволит нарушить симметрию. Если позволяют вычислительные ресурсы, сеть можно обучить несколько раз, начиная с разных случайных инициализаций, и затем сравнить результаты на отложенных данных.

Распределение, используемое для инициализации весов, обычно представляет собой либо равномерное распределение в диапазоне $[-\epsilon, \epsilon]$, либо гауссово распределение с нулевым средним значением вида $\mathcal{N}(0, \epsilon^2)$. Выбор значения ϵ очень важен, и для его определения уже разработаны различные эвристики. Один из широко используемых подходов называется *инициализацией Ге* (*He initialization*) (He et al., 2015b). Рассмотрим сеть, в которой слой l оценивает следующие преобразования:

$$a_i^{(l)} = \sum_{j=1}^M w_{ij} z_j^{(l-1)}, \quad (7.19)$$

$$z_i^{(l)} = \text{ReLU}(a_i^{(l)}), \quad (7.20)$$

где M – это количество элементов, передающих связи к элементу i , а функция активации ReLU задается в (6.17). Предположим, что весовые коэффициенты инициализируются гауссовым распределением $\mathcal{N}(0, \epsilon^2)$, и предположим, что выходы $z_j^{(l-1)}$ элементов в слое $l-1$ имеют дисперсию λ^2 (см. упражнение 7.9). Тогда можно без проблем показать, что

$$\mathbb{E}[a_i^{(l)}] = 0, \quad (7.21)$$

$$\text{var}[z_j^{(l)}] = \frac{M}{2} \epsilon^2 \lambda^2, \quad (7.22)$$

где коэффициент $1/2$ обусловлен функцией активации ReLU. В идеале хотелось бы, чтобы дисперсия предварительных активаций не уменьшалась до нуля и не росла значительно по мере перехода от одного слоя к другому. Поэтому необходимо, чтобы элементы слоя l также имели дисперсию λ^2 ,

и тогда получится следующий выбор стандартного отклонения гауссова распределения для инициализации весов, которые передаются элементу с M входами:

$$\epsilon = \sqrt{\frac{2}{M}}. \quad (7.23)$$

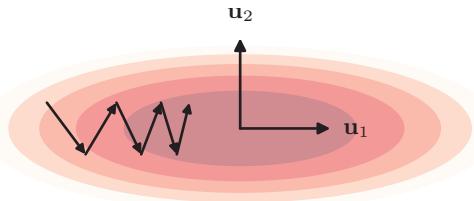
Также можно рассматривать масштаб ϵ распределения инициализации как гиперпараметр и исследовать различные его значения в процессе нескольких обучающих прогонов. Параметры смещения обычно устанавливаются на небольшие положительные значения, чтобы удостовериться, что большинство предварительных активаций изначально активны во время обучения. Это особенно полезно для элементов ReLU, поскольку необходимо, чтобы предварительные активации были положительными для получения ненулевого градиента в процессе обучения.

Другой важный класс методов инициализации параметров нейронной сети – это использование значений, полученных в результате обучения сети на другой задаче, или использование различных форм неконтролируемого обучения. Эти методы относятся к широкому классу методов трансферного обучения (см. раздел 6.3.4).

7.3. Сходимость

На практике при применении градиентного спуска необходимо выбрать значение параметра скорости обучения η . Рассмотрим простую поверхность ошибок, изображенную на рис. 7.3, для гипотетического двумерного весового пространства, где кривизна E в значительной степени изменяется в зависимости от направления, образуя своеобразную «долину» (показано эллипсами). В большинстве точек на поверхности ошибок локальный вектор градиента для пакетного градиентного спуска, перпендикулярный локальному контуру, не направлен прямо к минимуму. Интуитивно можно предположить, что увеличение значения η должно привести к большим шагам в пространстве весов и, следовательно, к более быстрой сходимости. Однако последовательные шаги будут колебаться взад-вперед по всей долине, и, если увеличить η слишком сильно, эти колебания станут расходящимися. Обратите внимание, что для большинства точек весового пространства локальный вектор отрицательного градиента $-\nabla E$ не направлен в сторону минимума функции ошибки. Векторы \mathbf{u}_1 и \mathbf{u}_2 являются собственными векторами матрицы Гессе. Поскольку величина η должна быть достаточно мала, чтобы избежать расходящихся колебаний по долине, продвижение по ней происходит очень медленно. В этом случае градиентный спуск требует множества небольших шагов для достижения минимума, и такая процедура является весьма неэффективной.

РИС. 7.3 Схематическое изображение градиентного спуска с фиксированным шагом для функции ошибки с существенно различной кривизной в разных направлениях



Более глубокое понимание природы этой проблемы можно получить при рассмотрении квадратичной аппроксимации функции ошибки в окрестности минимума (см. раздел 7.1.1). Исходя из (7.7), (7.8) и (7.10), градиент функции ошибки в этом приближении можно записать в виде

$$\nabla E = \sum_i \alpha_i \lambda_i \mathbf{u}_i. \quad (7.24)$$

И вновь используя (7.10), можно выразить изменение весового вектора в виде соответствующих изменений коэффициентов $\{\alpha_i\}$:

$$\Delta \mathbf{w} = \sum_i \Delta \alpha_i \mathbf{u}_i. \quad (7.25)$$

Сочетая (7.24) с (7.25) и формулой градиентного спуска (7.16) и воспользовавшись соотношением ортонормированности (7.9) для собственных векторов гессиана, можно получить следующее выражение для изменения α_i на каждом шаге алгоритма градиентного спуска:

$$\Delta \alpha_i = -\eta \lambda_i \alpha_i, \quad (7.26)$$

из которого следует (см. упражнение 7.10), что

$$\alpha_i^{\text{new}} = (1 - \eta \lambda_i) \alpha_i^{\text{old}}, \quad (7.27)$$

где old (старый) и new (новый) обозначают значения до и после обновления веса. Используя соотношение ортонормированности (7.9) для собственных векторов вместе с (7.10), получаем

$$\mathbf{u}_i^T (\mathbf{w} - \mathbf{w}^*) = \alpha_i, \quad (7.28)$$

и поэтому α_i можно интерпретировать как расстояние до минимума вдоль направления \mathbf{u}_i . Из (7.27) видно, что эти расстояния изменяются независимо, так что на каждом шаге расстояние вдоль направления \mathbf{u}_i умножается на коэффициент $(1 - \eta \lambda_i)$. После T шагов получается

$$\alpha_i^{(T)} = (1 - \eta \lambda_i)^T \alpha_i^{(0)}. \quad (7.29)$$

Отсюда следует, что при условии $|1 - \eta \lambda_i| < 1$ предел $T \rightarrow \infty$ приводит к $\alpha_i = 0$, а из (7.28) следует, что $\mathbf{w} = \mathbf{w}^*$ и, следовательно, весовой вектор достиг минимума ошибки.

Обратите внимание, что из (7.29) следует, что градиентный спуск приводит к линейной сходимости в окрестности минимума. Кроме того, сходи-

мость к стационарной точке требует, чтобы все λ_i были положительными, что, в свою очередь, означает, что стационарная точка действительно является минимумом. Увеличив η , можно уменьшить коэффициент $(1 - \eta\lambda_i)$, а значит, повысить скорость сходимости. Однако существует ограничение на увеличение η . Можно допустить, чтобы $(1 - \eta\lambda_i)$ имел отрицательное значение (что дают колеблющиеся значения α_i), но при этом необходимо обеспечить, чтобы $|1 - \eta\lambda_i| < 1$, иначе значения α_i будут расходиться. Это ограничивает значение η до $\eta < 2/\lambda_{\max}$, где λ_{\max} – это наибольшее из собственных значений. Однако скорость сходимости определяется наименьшим собственным значением, поэтому при η , установленном на наибольшее допустимое значение, сходимость вдоль направления, соответствующего наименьшему собственному значению (длинная ось эллипса на рис. 7.3), будет определяться как

$$\left(1 - \frac{2\lambda_{\min}}{\lambda_{\max}}\right), \quad (7.30)$$

где λ_{\min} – это наименьшее собственное значение. Если отношение $\lambda_{\min}/\lambda_{\max}$, которое в обратном виде называется числом обусловленности гессиана (*condition number of the Hessian*), очень мало, что соответствует сильно вытянутым эллиптическим контурам ошибок на рис. 7.3, то продвижение к минимуму будет крайне медленным.

7.3.1. Импульс

Один из простых методов решения проблемы сильно различающихся собственных значений заключается в добавлении *импульсного параметра* (*momentum term*) в формулу градиентного спуска. Это позволяет эффективно внести инерцию в движение через весовое пространство и сгладить колебания, показанные на рис. 7.3. Модифицированная формула градиентного спуска имеет вид:

$$\Delta \mathbf{W}^{(\tau-1)} = -\eta \nabla E \mathbf{W}^{(\tau-1)} + \mu \Delta \mathbf{W}^{(\tau-2)}, \quad (7.31)$$

где μ называется параметром импульса. Затем весовой вектор обновляется с помощью (7.15).

Для лучшего понимания влияния параметра импульса рассмотрим для начала движение через область весового пространства, для которой поверхность ошибок имеет относительно малую кривизну, как показано на рис. 7.4. Если сделать допущение, что градиент остается неизменным, тогда можно итеративно применить (7.31) к длинной серии весовых обновлений, а затем просуммировать полученный арифметический ряд, чтобы получить

$$\Delta \mathbf{W} = -\eta \nabla E \{1 + \mu + \mu^2 + \dots\} \quad (7.32)$$

$$= -\frac{\eta}{1 - \mu} \nabla E, \quad (7.33)$$

и здесь становится видно, что в результате введения импульсного параметра эффективная скорость обучения увеличивается с η до $\eta/(1 - \mu)$.

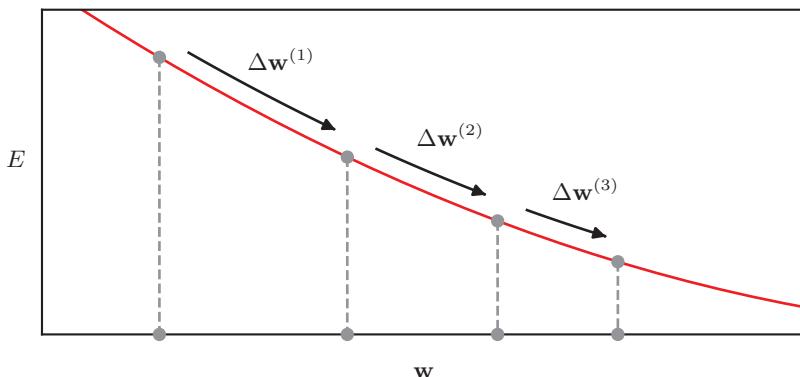


РИС. 7.4 При фиксированном параметре скорости обучения градиентный спуск по поверхности с малой кривизной приводит к последовательному уменьшению шагов, что соответствует линейной сходимости. Эффект от применения импульсного параметра в этой ситуации подобен увеличению эффективного параметра скорости обучения

Напротив, в области с высокой кривизной, где градиентный спуск имеет колебательный характер, как показано на рис. 7.5, последовательные вклады от импульсного параметра будут стремиться к нулю, и эффективная скорость обучения будет близка к η . Таким образом, импульсный параметр может привести к более быстрому сближению с минимумом без возникновения расходящихся колебаний. Схематическая иллюстрация влияния импульсного параметра приведена на рис. 7.6.

РИС. 7.5 Когда последовательные шаги градиентного спуска имеют колебательный характер, импульсный параметр оказывает незначительное влияние на эффективное значение параметра скорости обучения

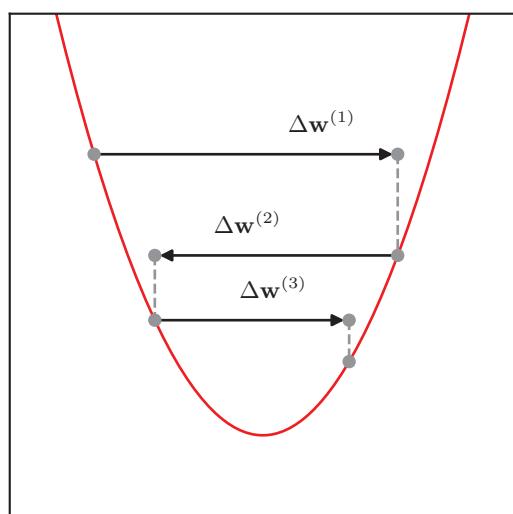
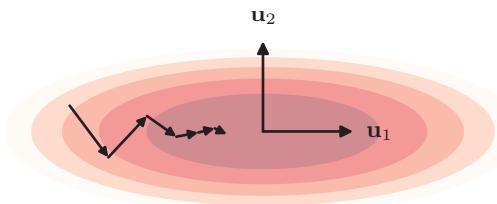


РИС. 7.6 Иллюстрация эффекта добавления импульсного параметра в алгоритм градиентного спуска, где показано более быстрое продвижение по долине функции ошибки по сравнению с немодифицированным градиентным спуском на рис. 7.3



Несмотря на то что добавление импульса может привести к улучшению эффективности градиентного спуска, оно также вводит второй параметр μ , значение которого необходимо выбирать в дополнение к значению параметра скорости обучения η . Из (7.33) видно, что μ должен находиться в диапазоне $0 \leq \mu \leq 1$. На практике обычно используется значение $\mu = 0,9$. Стохастический градиентный спуск с импульсом обобщен в алгоритме 7.3.

АЛГОРИТМ 7.3 Стохастический градиентный спуск с импульсом

Input: обучающий набор точек данных в интервале $n \in \{1, \dots, N\}$

Размер батча B

Функция ошибки для мини-батча $E_{n:n+B-1}(\mathbf{w})$

Параметр скорости обучения η

Параметр импульса μ

Начальный вектор весов \mathbf{w}

Output: итоговый вектор весов \mathbf{w}

```

 $n \leftarrow 1$ 
 $\Delta\mathbf{w} \leftarrow 0$ 
repeat
     $\Delta\mathbf{w} \leftarrow -\eta \nabla E_{n:n+B-1}(\mathbf{w}) + \mu \Delta\mathbf{w}$  // вычисление значения обновления
     $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$  // обновление весового вектора
     $n \leftarrow n + B$ 
    if  $n > N$  then
        перетасовка данных
         $n \leftarrow 1$ 
    end if
until convergence (до сходимости)
return  $\mathbf{w}$ 

```

Процесс сходимости можно дополнительно ускорить за счет использования модифицированной версии импульса, называемой *импульсом Нестерова* (*Nesterov momentum*) (Nesterov, 2004; Sutskever et al., 2013). В обычном стохастическом градиентном спуске с импульсом вначале вычисляется градиент в текущей позиции, а затем выполняется этап ускорения за счет добавления импульса от предыдущего этапа. В методе Нестерова порядок этих действий меняется, и сначала вычисляется этап на основе предыдущего импульса,

а затем вычисляется градиент в этой новой точке для определения обновления, так что

$$\Delta \mathbf{w}^{(\tau-1)} = -\eta \nabla E \mathbf{w}^{(\tau-1)} + \mu \Delta \mathbf{w}^{(\tau-2)} + \mu \Delta \mathbf{w}^{(\tau-2)}. \quad (7.34)$$

При пакетном градиентном спуске импульс Нестерова также может улучшить скорость сходимости, хотя для стохастического градиентного спуска он может оказаться менее эффективным.

7.3.2. График скорости обучения

В алгоритме стохастического градиентного спуска (7.18) необходимо задать значение параметра скорости обучения η . Если значение η очень мало, то обучение будет происходить медленно (см. раздел 7.3.1). Однако если увеличить η слишком сильно, это может привести к нарушению стабильности. Несмотря на допустимость некоторого уровня колебаний, они не должны быть расходящимися. На практике наилучшие результаты достигаются при использовании большего значения η в начале обучения с последующим уменьшением скорости обучения в ходе процесса, так что значение η становится функцией пошагового индекса τ :

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \eta^{(\tau-1)} \nabla E_n(\mathbf{w}^{(\tau-1)}). \quad (7.35)$$

К числу примеров графиков скорости обучения можно отнести режимы линейного, степенного и экспоненциального затухания:

$$\eta^{(\tau)} = (1 - \tau/K)\eta^{(0)} + (\tau/K)\eta^{(K)}, \quad (7.36)$$

$$\eta^{(\tau)} = \eta^{(0)}(1 + \tau/s)^c, \quad (7.37)$$

$$\eta^{(\tau)} = \eta^{(0)} e^{\tau/s}, \quad (7.38)$$

где в (7.36) значение η линейно уменьшается за K шагов, после чего его величина сохраняется постоянной на уровне $\eta(K)$. Оптимальные значения гиперпараметров $\eta^{(0)}$, $\eta^{(K)}$, K , S и c необходимо определять эмпирическим путем. На практике очень полезно наблюдать за *кривой обучения* (*learning curve*), которая отображает изменение функции ошибки в процессе итерации градиентного спуска, и убедиться, что она уменьшается с подходящей скоростью.

7.3.3. RMSProp и Adam

Ранее было показано, что оптимальная скорость обучения зависит от локальной кривизны поверхности ошибок (см. раздел 7.3), и, более того, эта кривизна может меняться в зависимости от направления в пространстве параметров. Это послужило основой для нескольких алгоритмов с различными скоростями обучения для каждого параметра сети. Значения этих скоростей обучения корректируются автоматически в процессе обучения. Здесь представлены некоторые из наиболее широко используемых образцов. Обратите

внимание, что эта концепция действительно применима только в том случае, если основные направления кривизны выровнены с осями в весовом пространстве, что соответствует локально диагональной матрице Гессе, что маловероятно на практике. Тем не менее алгоритмы такого типа вполне эффективны и находят широкое применение.

Ключевая идея алгоритма *AdaGrad* (сокращение от *adaptive gradient, адаптивный градиент*) заключается в том, что каждый параметр скорости обучения со временем уменьшается за счет использования накопленной суммы квадратов всех производных, вычисленных для этого параметра (Duchi, Hazan and Singer, 2011). Таким образом, параметры, связанные с большой кривизной, уменьшаются быстрее всего. А именно:

$$r_i^{(\tau)} = r_i^{(\tau-1)} + \left(\frac{\partial E(\mathbf{w})}{\partial w_i} \right)^2, \quad (7.39)$$

$$w_i^{(\tau)} = w_i^{(\tau-1)} - \frac{\eta}{\sqrt{r_i^\tau} + \delta} \left(\frac{\partial E(\mathbf{w})}{\partial w_i} \right), \quad (7.40)$$

где η – это параметр скорости обучения, а δ – это небольшая константа, порядка 10^{-8} , которая обеспечивает численную устойчивость в случае, если r_i стремится к нулю. Алгоритм инициализируется при $r_i^{(0)} = 0$. Здесь $E(\mathbf{w})$ – это функция ошибки для конкретного мини-батча, а обновление (7.40) – это стандартный стохастический градиентный спуск, но с модифицированной скоростью обучения, которая специфична для каждого параметра.

Одна из проблем AdaGrad заключается в накоплении квадратичных градиентов с самого начала обучения. Поэтому соответствующие обновления весов могут стать очень маленькими, и это может слишком сильно замедлить обучение на последующих этапах. Идея алгоритма *RMSProp* (сокращение от *root mean square propagation, распространение среднеквадратичного значения*) заключается в замещении суммы квадратичных градиентов AdaGrad экспоненциально взвешенным средним значением (Hinton, 2012), что дает

$$r_i^{(\tau)} = \beta r_i^{(\tau-1)} + (1 - \beta) \left(\frac{\partial E(\mathbf{w})}{\partial w_i} \right)^2, \quad (7.41)$$

$$w_i^{(\tau)} = w_i^{(\tau-1)} - \frac{\eta}{\sqrt{r_i^\tau} + \delta} \left(\frac{\partial E(\mathbf{w})}{\partial w_i} \right), \quad (7.42)$$

где $0 < \beta < 1$, а типичное значение $\beta = 0,9$.

Если сочетать RMSProp с импульсом, то получится метод оптимизации *Adam* (сокращение от *adaptive moments, адаптивные импульсы*) (Kingma and Ba, 2014). Алгоритм Adam хранит импульс для каждого параметра отдельно и использует уравнения обновления, которые состоят из экспоненциально взвешенных скользящих средних значений для градиентов и квадратов градиентов в виде

$$s_i^{(\tau)} = \beta_1 s_i^{(\tau-1)} + (1 - \beta_1) \left(\frac{\partial E(\mathbf{w})}{\partial w_i} \right), \quad (7.43)$$

$$r_i^{(\tau)} = \beta_2 r_i^{(\tau-1)} + (1 - \beta_2) \left(\frac{\partial E(\mathbf{w})}{\partial w_i} \right)^2, \quad (7.44)$$

$$\hat{s}_i^{(\tau)} = \frac{s_i^{(\tau)}}{1 - \beta_1^\tau}, \quad (7.45)$$

$$\hat{r}_i^{(\tau)} = \frac{r_i^{(\tau)}}{1 - \beta_2^\tau}, \quad (7.46)$$

$$w_i^{(\tau)} = w_i^{(\tau-1)} - \eta \frac{\hat{s}_i^{(\tau)}}{\sqrt{\hat{r}_i^{(\tau)}} + \delta}. \quad (7.47)$$

Здесь коэффициенты $1/(1 - \beta_1^\tau)$ и $1/(1 - \beta_2^\tau)$ корректируют смещение, вносимое инициализацией $s_i^{(0)}$ и $r_i^{(0)}$ до нуля (см. упражнение 7.12). Обратите внимание, что смещение обращается в ноль по мере увеличения τ , так как $\beta_i < 1$, и поэтому на практике эту поправку на смещение иногда опускают. Типичными значениями весовых параметров являются $\beta_1 = 0,9$ и $\beta_2 = 0,99$. Алгоритм Adam является самым распространенным алгоритмом обучения в глубоком обучении. Он обобщенно представлен в алгоритме 7.4.

АЛГОРИТМ 7.4 Оптимизация Adam

Input: обучающий набор точек данных в интервале $n \in \{1, \dots, N\}$

Размер батча B

Функция ошибки для мини-батча $E_{n:n+B-1}(\mathbf{w})$

Параметр скорости обучения η

Параметры замедления β_1 и β_2

Параметр стабилизации δ

Output: итоговый вектор весов \mathbf{w}

$n \leftarrow 1$

$\mathbf{s} \leftarrow \mathbf{0}$

$\mathbf{r} \leftarrow \mathbf{0}$

repeat

 Выбор произвольного мини-батча из \mathcal{D}

$\mathbf{g} = -\nabla E_{n:n+B-1}(\mathbf{w})$ // оценка вектора градиента

$\mathbf{s} \leftarrow \beta_1 \mathbf{s} + (1 - \beta_1) \mathbf{g}$

$\mathbf{r} \leftarrow \beta_2 \mathbf{r} + (1 - \beta_2) \mathbf{g} \odot \mathbf{g}$ // поэлементное умножение

$\hat{\mathbf{s}} \leftarrow \mathbf{s}/(1 - \beta_1^\tau)$ // коррекция смещения

$\hat{\mathbf{r}} \leftarrow \mathbf{r}/(1 - \beta_2^\tau)$ // коррекция смещения

$\Delta \mathbf{w} \leftarrow -\eta \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta}$ // поэлементные операции

```

w ← w + Δw // обновление весового вектора
n ← n + B
if n + B > N then
    перетасовка данных
    n ← 1
end if
until convergence (до сходимости)
return w

```

7.4. Нормализация

Нормализация переменных, вычисляемых при прямом проходе нейронной сети, избавляет ее от необходимости иметь дело с очень большими или очень маленькими значениями. В принципе, веса и смещения в нейронной сети могут адаптироваться к любым значениям входных и скрытых переменных, однако на практике нормализация может оказаться решающим фактором для эффективного обучения. Здесь будут рассмотрены три вида нормализации в соответствии со спецификой нормализации входных данных, мини-батчей или слоев.

7.4.1. Нормализация данных

Иногда приходится сталкиваться с наборами данных, где различные входные переменные находятся в совершенно разных диапазонах значений. Например, в медицинских данных рост пациента может измеряться в метрах (например, 1,8 м), а количество тромбоцитов в крови может измеряться в тромбоцитах на микролитр (например, 300 000 тромбоцитов на микролитр). Такие вариации могут значительно усложнить обучение методом градиентного спуска. Рассмотрим однослойную регрессионную сеть с двумя весами, в которой две соответствующие входные переменные имеют очень разные диапазоны. Изменение значения одного из весов приводит к гораздо большим изменениям в выходном сигнале и, следовательно, в функции ошибки, чем аналогичные изменения другого веса. Это соответствует поверхности ошибки с очень разной кривизной вдоль разных осей, как показано на рис. 7.3.

Поэтому для непрерывных входных переменных может оказаться очень полезным изменение масштаба входных значений, чтобы они находились в одинаковых диапазонах. Сделать это можно путем оценки среднего значения и дисперсии каждой входной величины:

$$\mu_i = \frac{1}{N} \sum_{n=1}^N x_{ni}, \quad (7.48)$$

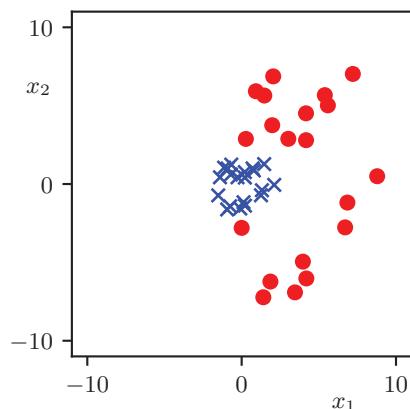
$$\sigma_i^2 = \frac{1}{N} \sum_{n=1}^N (x_{ni} - \mu_i)^2. \quad (7.49)$$

Это вычисление выполняется один раз, до начала обучения. Затем входные значения масштабируются заново посредством

$$\tilde{x}_{ni} = \frac{x_{ni} - \mu_i}{\sigma_i} \quad (7.50)$$

так, чтобы заново масштабированные значения $\{\tilde{x}_{ni}\}$ имели нулевое среднее значение и единичную дисперсию (см. упражнение 7.14). Обратите внимание, что одни и те же значения μ_i и σ_i должны использоваться для предварительной обработки любых данных разработки, валидации или тестирования, чтобы обеспечить одинаковое масштабирование всех входных данных. Нормализация входных данных показана на рис. 7.7. Красные кружки показывают исходные точки данных для набора данных с двумя переменными. Синие крестики показывают набор данных после нормализации, в результате которой каждая переменная теперь имеет нулевое среднее значение и единичную дисперсию по всему набору данных.

РИС. 7.7 Иллюстрация эффекта нормализации входных данных



7.4.2. Пакетная нормализация

Ранее уже рассматривалась значимость нормализации входных данных, и теперь аналогичные принципы можно применить к переменным в каждом скрытом слое глубокой сети. Если диапазон значений активации в определенном скрытом слое сильно различается, то нормализация этих значений до нулевого среднего значения и единичной дисперсии должна облегчить задачу обучения для следующего слоя. Тем не менее в отличие от нормализации входных значений, которую можно выполнить один раз перед началом обучения, нормализацию значений скрытых блоков придется повторять в процессе обучения при каждом обновлении значений весов. Такая операция называется *пакетной нормализацией* (*batch normalization*) (Ioffe and Szegedy, 2015).

Еще одной причиной необходимости пакетной нормализации служат такие явления, как *исчезающие градиенты* (*vanishing gradients*) и *взрывающие-*

ся градиенты (*exploding gradients*), которые характерны для задач обучения очень глубоких нейронных сетей. Из правила цепных вычислений следует, что градиент функции ошибки E относительно параметра в первом слое сети (см. раздел 8.1.5) определяется как

$$\frac{\partial E}{\partial w_i} = \sum_m \cdots \sum_l \sum_j \frac{\partial z_m^{(1)}}{\partial w_i} \cdots \frac{\partial z_j^{(K)}}{\partial z_l^{(K-1)}} \frac{\partial E}{\partial z_j^{(K)}}, \quad (7.51)$$

где $z_j^{(k)}$ обозначает активацию узла j в слое k , а каждая из частных производных (см. раздел 8.1.5) в правой части (7.51) представляет собой элементы матрицы Якоби для этого слоя. Произведение большого числа таких членов будет стремиться к 0, если большинство из них имеют величину < 1 , и будет стремиться к ∞ , если большинство из них имеют величину > 1 . Следовательно, с увеличением глубины сети градиенты функции ошибки могут стремиться к очень большим или очень малым значениям. Пакетная нормализация в значительной степени решает эту проблему.

В качестве примера пакетной нормализации рассмотрим отдельный слой в многослойной сети. Каждый скрытый элемент в этом слое вычисляет нелинейную функцию своей входной предварительной активации $z_i = h(a_i)$, и поэтому возникает выбор, что нормализовать – значения предварительной активации a_i или значения активации z_i . На практике можно использовать любой из этих подходов, но здесь проиллюстрируем процедуру на примере нормализации предварительных активаций. Поскольку значения весов обновляются после каждого мини-батча примеров, то нормализация применяется к каждому мини-батчу. В частности, для мини-батча размера K задается как

$$\mu_i = \frac{1}{K} \sum_{n=1}^K a_{ni}, \quad (7.52)$$

$$\sigma_i^2 = \frac{1}{K} \sum_{n=1}^K (a_{ni} - \mu_i)^2, \quad (7.53)$$

$$\hat{a}_{ni} = \frac{a_{ni} - \mu_i}{\sqrt{\sigma_i^2 + \delta}}, \quad (7.54)$$

где слагаемые по $n = 1, \dots, K$ взяты по элементам мини-батча. Здесь δ – это небольшая константа, введенная для предотвращения численных проблем в ситуациях, когда величина σ_i^2 слишком мала.

За счет нормализации предварительных активаций в определенном слое сети происходит уменьшение числа степеней свободы в параметрах этого слоя и, следовательно, уменьшение его репрезентативной способности. Компенсировать это можно путем изменения масштаба предварительных активаций пакета для получения среднего значения β_i и стандартного отклонения γ_i , используя

$$\tilde{a}_{ni} = \gamma_i \hat{a}_{ni} + \beta_i, \quad (7.55)$$

где β_i и γ_i – это адаптивные параметры, которые обучаются методом градиентного спуска совместно с весами и смещениями сети. Эти обучаемые параметры представляют собой ключевое отличие от процесса нормализации входных данных (см. раздел 7.4.1).

Может показаться, что преобразование (7.55) просто отменяет эффект пакетной нормализации, поскольку среднее значение и дисперсия теперь могут снова адаптироваться к произвольным значениям. Однако принципиальная разница заключается в характере изменения параметров в процессе обучения. Для исходной сети среднее значение и дисперсия в мини-батче определяются сложной функцией для всех весов и смещений этого слоя, в то время как в представлении (7.55) они определяются непосредственно независимыми параметрами β_i и γ_i , которые гораздо проще определить в процессе обучения методом градиентного спуска.

Уравнения (7.52) – (7.55) описывают преобразование переменных, которое является дифференцируемым относительно обучаемых параметров β_i и γ_i . Это преобразование можно рассматривать как дополнительный слой в нейронной сети, и, таким образом, после каждого стандартного скрытого слоя может следовать слой пакетной нормализации. Структура процесса пакетной нормализации показана на рис. 7.8. При пакетной нормализации, показанной в (a), среднее значение и дисперсия вычисляются по всему мини-батчу отдельно для каждого скрытого элемента. При послойной нормализации, показанной в (b), среднее значение и дисперсия вычисляются по скрытым блокам отдельно для каждой точки данных.

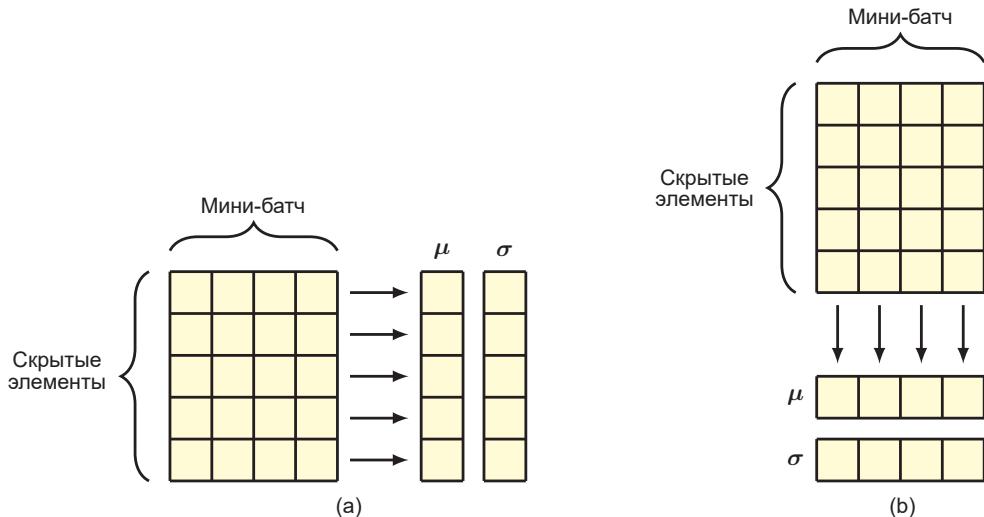


РИС. 7.8 Схема пакетной нормализации и нормализации слоев в нейронной сети

После обучения сети, когда возникает необходимость сделать прогноз на новых данных, обучающие мини-батчи отсутствуют, и определить среднее

значение и дисперсию только по одному примеру данных невозможно. Для решения этой проблемы, в принципе, можно было бы оценить μ_i и σ_i^2 для каждого слоя на всем обучающем множестве после окончательного обновления весов и смещений. Однако это потребовало бы обработки всего набора данных исключительно для оценки этих величин, что обычно слишком затратно. Вместо этого в течение всего этапа обучения производится расчет скользящих средних значений:

$$\bar{\mu}_i^{(\tau)} = \alpha \bar{\mu}_i^{(\tau)} + (1 - \alpha) \mu_i, \quad (7.56)$$

$$\bar{\sigma}_i^{(\tau)} = \alpha \bar{\sigma}_i^{(\tau)} + (1 - \alpha) \sigma_i, \quad (7.57)$$

где $0 \leq \alpha \leq 1$. Эти скользящие средние значения не играют никакой роли во время обучения, однако используются для обработки новых точек данных на этапе вывода.

Несмотря на высокую эффективность пакетной нормализации на практике, до сих пор неясно, почему она так хорошо работает. Изначально пакетная нормализация была продиктована тем, что обновление весов в более ранних слоях сети изменяет распределение значений в более поздних слоях – это явление называется *внутренним сдвигом ковариаций* (*internal covariate shift*). Однако более поздние исследования (Santurkar et al., 2018) позволяют предположить, что сдвиг ковариаций не является существенным фактором, а улучшение обучения происходит за счет улучшения слаженности ландшафта функции ошибки.

7.4.3. Нормализация слоев

При пакетной нормализации в случае, если размер пакета слишком мал, оценки среднего значения и дисперсии получаются слишком зашумленными. Кроме того, для очень больших обучающих наборов вычисление мини-батча может быть распределено по разным графическим процессорам, что приводит к снижению эффективности общей нормализации по всему мини-батчу. Альтернативой нормализации по образцам в мини-батче для каждого скрытого элемента по отдельности является нормализация по значениям скрытых элементов для каждой точки данных по отдельности. Этот способ известен как *нормализация слоя* (*layer normalization*) (Ba, Kiros and Hinton, 2016) (см. раздел 12.2.5). Она была предложена в контексте рекуррентных нейронных сетей, где распределения меняются после каждого этапа времени, что делает пакетную нормализацию неосуществимой.

Однако она полезна и в рамках других архитектур, таких как *сети-трансформеры* (*transformer networks*) (см. главу 12). По аналогии с пакетной нормализацией выполним следующее преобразование:

$$\mu_n = \frac{1}{M} \sum_{i=1}^m a_{ni}, \quad (7.58)$$

$$\sigma_n^2 = \frac{1}{M} \sum_{i=1}^M (a_{ni} - \mu_i)^2, \quad (7.59)$$

$$\hat{a}_{ni} = \frac{a_{ni} - \mu_n}{\sqrt{\sigma_n^2 + \delta}}, \quad (7.60)$$

где суммы $i = 1, \dots, M$ берутся по всем скрытым элементам в слое. Как и в случае пакетной нормализации, дополнительные обучаемые параметры среднего значения и стандартного отклонения вводятся для каждого скрытого элемента по отдельности в виде (7.55). Обратите внимание, что одна и та же функция нормализации может использоваться как при обучении, так и при выводе. Поэтому хранить скользящие средние значения нет никакой необходимости. Сравнение нормализации слоя с пакетной нормализацией показано на рис. 7.8.

Упражнения

- 7.1** (*) Подставив (7.10) в (7.7) и используя (7.8) и (7.9), докажите, что функция ошибки (7.7) может быть записана в виде (7.11).
- 7.2** (*) Рассмотрим матрицу Гессе \mathbf{H} с собственными векторами в уравнении (7.8). Определяя вектор \mathbf{v} в (7.14) равным каждому из собственных векторов \mathbf{u}_i по очереди, докажите, что \mathbf{H} положительно определена тогда и только тогда, когда все ее собственные значения положительны.
- 7.3** (**) Рассматривая локальное разложение Тейлора (7.7) функции ошибки относительно стационарной точки \mathbf{w}^* , докажите, что необходимым и достаточным условием того, что стационарная точка является локальным минимумом функции ошибки, является то, что матрица Гессе \mathbf{H} , определяемая (7.5) при $\hat{\mathbf{w}} = \mathbf{w}^*$, является положительно определенной.
- 7.4** (**) Рассмотрим линейную регрессионную модель с одной входной переменной x и одной выходной переменной y вида

$$y(x, w, b) = wx + b \quad (7.61)$$

вместе с функцией ошибки в виде суммы квадратов, заданной

$$E(w, b) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, w, b) - t_n\}^2. \quad (7.62)$$

Выполните выражения для элементов матрицы Гессе 2×2 , заданной вторыми производными функции ошибки по весовому параметру w и параметру смещения b . Докажите, что след и определитель этой матрицы Гессе положительны. Поскольку след представляет собой сумму собственных значений, а детерминант соответствует произведению соб-

ственных значений (см. приложение А), то оба собственных значения положительны и, следовательно, стационарная точка функции ошибки является минимумом.

- 7.5** (**) Рассмотрим однослойную классификационную модель с одной входной переменной x и одной выходной переменной y вида

$$y(x, w, b) = \sigma(wx + b), \quad (7.63)$$

где $\sigma(\cdot)$ – это логистическая сигмоидная функция, определенная в (5.42), вместе с функцией ошибки перекрестной энтропии, представленной в виде

$$E(w, b) = \sum_{n=1}^N \{t_n \ln y(x_n, w, b) + (1 - t_n) \ln(1 - y(x_n, w, b))\}. \quad (7.64)$$

Выполните выражения для элементов матрицы Гессе 2×2 , заданной вторыми производными функции ошибки по весовому параметру w и параметру смещения b . Докажите, что след и определитель этого Гессе положительны. Поскольку след представляет собой сумму собственных значений, а определитель – это произведение собственных значений (см. приложение А), то оба собственных значения положительны, а значит, стационарная точка функции ошибки является минимумом.

- 7.6** (**) Рассмотрим заданную в (7.7) квадратичную функцию ошибки, где матрица Гессе \mathbf{H} имеет уравнение с собственными значениями, заданное в (7.8). Докажите, что контуры постоянной ошибки представляют собой эллипсы, оси которых совпадают с собственными векторами \mathbf{u}_i , а длины обратно пропорциональны квадратным корням из соответствующих собственных значений λ_i .
- 7.7** (*) Докажите, что в силу симметрии матрицы Гессе \mathbf{H} число независимых элементов в квадратичной функции ошибки (7.3) равно $W(W + 3)/2$.
- 7.8** (*) Рассмотрим набор значений x_1, \dots, x_N , взятый из распределения со средним значением μ и дисперсией σ^2 , и определим выборочное среднее значение как

$$\bar{x} = \frac{1}{N} \sum_{n=1}^N x_n. \quad (7.65)$$

Докажите, что математическое ожидание квадрата ошибки $(\bar{x} - \mu)^2$ относительно распределения, из которого взяты данные, равно σ^2/N . Это говорит о том, что среднеквадратичная ошибка среднего выборочного значения равна σ/\sqrt{N} , и она уменьшается сравнительно медленно по мере увеличения объема выборки N .

- 7.9** (**) Рассмотрим многослойную сеть, которая вычисляет функции (7.19) и (7.20) в слое l . Предположим, что для инициализации весов использу-

ется гауссово распределение $\mathcal{N}(0, \epsilon^2)$, и предположим, что выходы $z_i^{(l-1)}$ элементов в слое $l - 1$ имеют дисперсию λ^2 . Используя форму функции активации ReLU, докажите, что среднее значение и дисперсия выходов в слое l задаются в (7.21) и (7.22) соответственно. Таким образом, можно доказать, что в случае, если элементы в слое l также должны иметь предварительные активации с дисперсией λ^2 , то значение параметра должно быть задано в (7.23).

- 7.10** (**) Используя (7.7), (7.8) и (7.10), выведите результаты (7.24) и (7.25), которые выражают вектор градиента и общее обновление веса как выражения по собственным векторам матрицы Гессе. Используйте эти результаты вместе с соотношением ортонормированности собственных векторов (7.9) и формулой пакетного градиентного спуска (7.16), чтобы получить результат (7.26) для обновления пакетного градиентного спуска, выраженного в коэффициентах $\{\alpha_i\}$.
- 7.11** (*) Рассмотрим плавно изменяющуюся поверхность ошибки с малой кривизной, на которой градиент медленно изменяется с изменением положения. Докажите, что при малых значениях параметров скорости обучения и импульса обновление градиента с импульсом Нестерова, определенное в (7.34), эквивалентно стандартному градиентному спуску с импульсом, определенному в (7.31).
- 7.12** (**) Рассмотрим последовательность значений $\{x_1, \dots, x_N\}$ некоторой переменной x , и пусть необходимо вычислить экспоненциально взвешенное скользящее среднее значение по формуле

$$\mu_n = \beta \mu_{n-1} + (1 - \beta)x_n, \quad (7.66)$$

где $0 \leq \beta \leq 1$. Используя следующий результат для суммы конечного геометрического ряда

$$\sum_{k=1}^n \beta^{k-1} = \frac{1 - \beta^n}{1 - \beta}, \quad (7.67)$$

докажите, что если последовательность средних значений инициализируется с помощью $\mu_0 = 0$, то оценки являются смещенными и что это смещение может быть скорректировано с помощью

$$\hat{\mu}_n = \frac{\mu_n}{1 - \beta^n}. \quad (7.68)$$

- 7.13** (*) При градиентном спуске весовой вектор w обновляется пошагово в пространстве весов в направлении отрицательного градиента, управляемого параметром скорости обучения η . Предположим, что вместо этого в пространстве весов будет выбрано направление d , вдоль которого минимизируется функция ошибки при нынешнем весовом векторе $w^{(t)}$. Это предполагает минимизацию величины

$$E(\mathbf{w}^{(t)} + \lambda \mathbf{d}) \quad (7.69)$$

как функции от λ для получения значения λ^* , соответствующего новому весовому вектору $\mathbf{w}^{(t+1)}$. Докажите, что градиент $E(\mathbf{w})$ в точке $\mathbf{w}^{(t+1)}$ ортогонален вектору \mathbf{d} . Этот подход известен как метод «линейного поиска» (*line search*). Он является основой множества различных алгоритмов численной оптимизации (Bishop, 1995b).

- 7.14** (*) Докажите, что перенормированные входные переменные, определяемые в (7.50), где μ_i определяется в (7.48), а σ_i^2 определяется в (7.49), имеют нулевое среднее значение и единичную дисперсию.

Глава 8

Обратное распространение

Целью этой главы является определение эффективной методики оценки градиента функции ошибки $E(w)$ для нейронной сети прямого распространения. В этой главе будет рассмотрено использование локальной схемы передачи сообщений, в которой информация передается назад по сети, что называется *обратным распространением ошибки* (*error backpropagation*, иногда просто *backprop*).

Исторически уравнения обратного распространения выводились вручную, а затем внедрялись в программное обеспечение вместе с уравнениями прямого распространения, при этом оба этапа занимали много времени и были сопряжены с возможными ошибками. Однако современные программные среды для нейронных сетей позволяют эффективно вычислять практически любые интересующие производные, затрачивая лишь минимум усилий, не считая кодирования исходной сетевой функции (см. раздел 8.2). Эта концепция, называемая *автоматическим дифференцированием* (*automatic differentiation*), играет ключевую роль в современном глубоком обучении. Однако важно разобраться в механизме вычислений, чтобы не полагаться на программные решения в виде «черного ящика». Поэтому в этой главе представлены ключевые концепции обратного распространения и подробно рассмотрена схема автоматического дифференцирования.

Обратите внимание, что термин «обратное распространение» используется в литературе по нейронным вычислениям в различных значениях. Например, сеть с архитектурой прямой передачи могут называть сетью обратного распространения. Также термин «обратное распространение» иногда используется для описания сквозной процедуры обучения нейронной сети, включая обновление параметров градиентного спуска. В этой книге термин «обратное распространение» будет использоваться именно для обозначения вычислительной процедуры численной оценки производных, таких как градиент функции ошибки по отношению к весам и смещениям сети. Эта процедура также может быть применена для оценки других важных производных, таких как матрица Якоби и матрица Гессе.

8.1. Оценка градиентов

Теперь рассмотрим алгоритм обратного распространения для сети произвольного вида с произвольной топологией прямой передачи, произвольными дифференцируемыми нелинейными функциями активации и широким классом функций ошибки. Полученные формулы будут проиллюстрированы на примере структуры простой многослойной сети с одним слоем сигмоидных скрытых элементов и суммой квадратов ошибок.

Многие функции ошибок, представляющие практический интерес, например определяемые методом максимального правдоподобия для набора i.i.d.-данных, состоят из суммы членов, по одному для каждой точки данных в обучающем наборе, так что

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}). \quad (8.1)$$

Далее будет рассмотрена задача оценки $\nabla E_n(\mathbf{w})$ для одного такого члена в функции ошибки. Это может быть использовано непосредственно для метода стохастического градиентного спуска, или результаты могут быть накоплены на множестве обучающих точек данных для методов пакетной или мини-пакетной обработки.

8.1.1. Однослойные сети

Рассмотрим для начала простую линейную модель, в которой выходы y_k представляют собой линейные комбинации входных переменных x_i вида

$$y_k = \sum_i w_{ki} x_i \quad (8.2)$$

вместе с функцией ошибки в виде суммы квадратов, которая для конкретной точки входных данных n имеет вид:

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2, \quad (8.3)$$

где $y_{nk} = y_k(\mathbf{x}_n, \mathbf{w})$, а t_{nk} – это соответствующее целевое значение. Градиент этой функции ошибки относительно веса w_{ji} определяется как

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni}. \quad (8.4)$$

Это можно интерпретировать как «локальное» вычисление с использованием произведения «сигнала ошибки» $y_{nj} - t_{nj}$, связанного с выходным концом звена w_{ji} , и переменной x_{ni} , связанной с входным концом звена. В разделе 5.4.3 было описано, как аналогичная формула выводится для логистической сигмоидной функции активации вместе с функцией ошибки перекрестной эн-

тропии и аналогично для функции активации softmax вместе с соответствующей ей многомерной функцией ошибки перекрестной энтропии. Теперь разберемся, как этот простой результат распространяется на более сложный случай многослойных сетей с прямой передачей данных.

8.1.2. Общие сети с прямой передачей

В общем случае сеть с прямой передачей данных состоит из набора элементов, каждый из которых вычисляет взвешенную сумму своих входов:

$$a_j = \sum_i w_{ji} z_i, \quad (8.5)$$

где z_i – это либо активация другого элемента, либо входной элемент, который передает связь на элемент j , а w_{ji} – это вес, относящийся к этой связи. В эту сумму можно включить смещения с помощью добавления дополнительного элемента или входа с активацией, фиксированной на +1 (см. раздел 6.2), и поэтому в данном случае не обязательно рассматривать смещения в явном виде. Сумма в (8.5), так называемая предварительная активация, преобразуется с помощью нелинейной функции активации $h(\cdot)$ для активации z_j элемента j в виде

$$z_j = h(a_j). \quad (8.6)$$

Обратите внимание, что одна или несколько переменных z_i в сумме по формуле (8.5) может быть входом, и точно так же элемент j в (8.6) может быть выходом.

Для каждой точки данных в обучающем множестве предполагается подача соответствующего входного вектора в сеть и вычисление активаций всех скрытых и выходных элементов в сети путем последовательного применения (8.5) и (8.6). Этот процесс называется *прямым распространением* (*forward propagation*), поскольку его можно рассматривать как прямой поток информации по сети.

Теперь рассмотрим оценку производной E_n по весу w_{ji} . Выходы различных элементов будут зависеть от конкретной точки входных данных n . Чтобы не усложнять обозначения, подстрочный индекс n в переменных сети можно опустить. Прежде всего следует отметить, что E_n зависит от веса w_{ji} только через суммарный вход a_j элемента j . Поэтому для получения результата можно применить правило цепочки для частных производных:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \quad (8.7)$$

Теперь введем полезные обозначения:

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j}, \quad (8.8)$$

где δ зачастую называют *ошибками* (*errors*) по причинам, которые будут описаны далее. Используя (8.5), можно записать:

$$\frac{\partial a_j}{\partial w_{ji}} = z_i. \quad (8.9)$$

Подставляя (8.8) и (8.9) в (8.7), получаем:

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i. \quad (8.10)$$

Уравнение (8.10) указывает на то, что требуемая производная получается простым умножением значения δ для элемента на выходе весов на значение z для элемента на входе весов (где $z = 1$ для смещения). Обратите внимание, что это выражение имеет ту же форму, что и в случае простой линейной модели в (8.4). Таким образом, для оценки производных необходимо вычислить только значение δ_j для каждого скрытого и выходного элемента сети, а затем применить (8.10).

Как уже было замечено, для выходных элементов есть

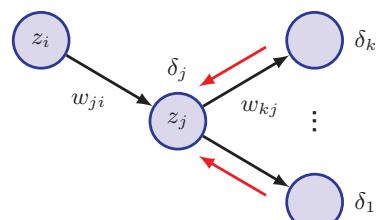
$$\delta_k = y_k - t_k \quad (8.11)$$

при условии, что в качестве функции активации выходного элемента используется каноническая связь (см. раздел 5.4.6). Для оценки δ для скрытых элементов снова воспользуемся цепным правилом для частных производных:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j}, \quad (8.12)$$

где сумма относится ко всем элементам k , которым элемент j посылает соединения. Расположение элементов и весов показано на рис. 8.1. Черные стрелки обозначают направление потока информации при прямом распространении, а красные – при обратном распространении информации об ошибке. Обратите внимание, что элементы, обозначенные k , включают другие скрытые элементы и/или выходные элементы. При записи (8.12) учитывается тот факт, что вариации a_j приводят к изменениям функции ошибки только за счет вариаций переменных a_k .

РИС. 8.1 Иллюстрация вычисления δ_j для скрытого элемента j путем обратного распространения δ от тех элементов k , которым элемент j посылает связи



Если теперь подставить определение δ_j из (8.8) в формулу (8.12) и воспользоваться (8.5) и (8.6), то получится следующая формула обратного распространения (см. упражнение 8.1):

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k, \quad (8.13)$$

которая говорит о том, что значение δ для конкретного скрытого элемента может быть получено путем обратного распространения δ от элементов, расположенных выше в сети, как показано на рис. 8.1. Обратите внимание, что суммирование в (8.13) происходит по первому индексу w_{kj} , что соответствует обратному распространению информации по сети (см. упражнение 8.2), тогда как в уравнении прямого распространения (8.5) оно происходит по второму индексу. Поскольку значения δ для выходных элементов уже известны, из этого следует, что рекурсивное применение (8.13) даст возможность оценить δ для всех скрытых элементов сети с прямой передачей, независимо от ее топологии. Процедура обратного распространения кратко описана в алгоритме 8.1.

АЛГОРИТМ 8.1 Обратное распространение

Input: вектор входных данных x_n

Параметры сети w

Функция ошибки $E_n(w)$ для входа x_n

Функция активации $h(a)$

Output: Производные функции ошибки $\{\partial E_n / \partial w_{ji}\}$

// Прямое распространение

for $j \in$ всех скрытых и выходных элементов **do**
| $a_j \leftarrow \sum_i w_{ji} z_i$ // $\{z_i\}$ включает входы $\{x_i\}$
| $z_j \leftarrow h(a_j)$ // функция активации
end for

// Оценка ошибки

for $k \in$ всех выходных элементов **do**
| $\delta_k \leftarrow \frac{\partial E_n}{\partial a_k}$ // вычисление ошибки
end for.

// Обратное распространение, в обратном порядке

for $j \in$ всех выходных элементов **do**
| $\delta_j \leftarrow h'(a_j) \sum_k w_{kj} \delta_k$ // рекурсивная обратная оценка
| $\frac{\partial E_n}{\partial w_{ji}} \leftarrow \delta_j z_i$ // оценка производных
end for

return $\left\{ \frac{\partial E_n}{\partial w_{ji}} \right\}$

Для пакетных методов производная общей ошибки E может быть получена путем повторения описанных выше действий для каждой точки данных в обучающем наборе с последующим суммированием по всем точкам данных в пакете или мини-батче:

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}}. \quad (8.14)$$

В приведенном примере вывода косвенно предполагается, что каждый скрытый или выходной элемент сети имеет одну и ту же функцию активации $h(\cdot)$. Это предположение несложно обобщить таким образом, чтобы различные элементы могли иметь индивидуальные функции активации, для чего достаточно просто отслеживать соответствие формы $h(\cdot)$ тому или иному элементу.

8.1.3. Простой пример

Приведенный выше метод обратного распространения позволяет использовать общие формы для функции ошибки, функций активации и сетевой топологии. В качестве иллюстрации применения этого алгоритма рассмотрим двухслойную сеть, изображенную на рис. 6.9, с суммой квадратов ошибок. Выходные элементы имеют линейные функции активации, так что $y_k = a_k$, а скрытые элементы имеют сигмоидные функции активации, заданные как

$$h(a) = \tanh(a), \quad (8.15)$$

где $\tanh(a)$ определяется в (6.14). Полезной особенностью этой функции является то, что ее производная может быть выражена в очень простой форме:

$$h'(a) = 1 - h(a)^2. \quad (8.16)$$

Также рассмотрим функцию ошибки в виде суммы квадратов, чтобы для точки данных n ошибка задавалась как

$$E_n = \frac{1}{2} \sum_{k=1}^K (y_k - t_k)^2, \quad (8.17)$$

где y_k – это активация выходного элемента k , а t_k – это соответствующее целевое значение для конкретного входного вектора \mathbf{x}_n .

Сначала для каждой точки данных в обучающем множестве поочередно выполняется метод прямого распространения с использованием

$$a_j = \sum_{i=0}^D w_{ji}^{(1)} x_i, \quad (8.18)$$

$$z_j = \tanh(a_j), \quad (8.19)$$

$$y_k = \sum_{j=0}^M w_{kj}^{(2)} z_j, \quad (8.20)$$

где D – это размерность входного вектора \mathbf{x} , а M – это общее количество скрытых элементов. Также здесь используется $x_0 = z_0 = 1$ для включения параметров смещения в веса. Далее необходимо вычислить δ для каждого выходного элемента, используя

$$\delta_k = y_k - t_k. \quad (8.21)$$

Затем методом обратного распространения этих ошибок получаем δ для скрытых элементов, используя

$$\delta_j = (1 - z_j^2) \sum_{k=1}^K w_{kj}^{(2)} \delta_k, \quad (8.22)$$

что следует из (8.13) и (8.16). Наконец, производные по весам первого и второго слоев имеют вид:

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i, \quad \frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j. \quad (8.23)$$

8.1.4. Численное дифференцирование

Одним из важнейших преимуществ обратного распространения является его вычислительная эффективность. Для ее понимания рассмотрим зависимость числа вычислительных операций, необходимых для оценки производных функции ошибки, от общего числа W весов и смещений в сети.

Для одной оценки функции ошибки (для заданной точки входных данных) потребуется $\mathcal{O}(W)$ операций при достаточно большом W . Так происходит потому, что, за исключением сети с очень редкими связями, количество весов обычно намного больше количества элементов, и поэтому основная часть вычислительных усилий при прямом распространении приходится на оценку сумм в (8.5), а оценка функций активации составляет небольшую часть накладных расходов. Каждый член в сумме в (8.5) требует одного умножения и одного сложения, что приводит к общим вычислительным затратам в раз- мере $\mathcal{O}(W)$.

Альтернативным подходом к обратному распространению для вычисления производных функции ошибки является использование конечных разностей. Для этого необходимо поочередно воздействовать на каждый вес и аппроксимировать производные с помощью выражения

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji})}{\epsilon} + \mathcal{O}(\epsilon), \quad (8.24)$$

где $\epsilon \ll 1$. При программном моделировании точность аппроксимации производных можно повысить путем их уменьшения до тех пор, пока не возникнут проблемы с численным округлением. Точность метода конечных разностей можно значительно повысить за счет использования симметричных центральных разностей (*central differences*) вида

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji} - \epsilon)}{2\epsilon} + \mathcal{O}(\epsilon^2). \quad (8.25)$$

В этом случае (см. упражнение 8.3) коррекции $\mathcal{O}(\epsilon)$ отменяются, в чем можно убедиться с помощью разложения в ряд Тейлора правой части (8.25), и поэтому остаточные корректины равны $\mathcal{O}(\epsilon^2)$. Тем не менее обратите внимание, что количество вычислительных шагов увеличивается примерно вдвое по сравнению с (8.24). На рис. 8.2 показан график зависимости погрешности численной оценки градиента с помощью конечных разностей (8.24) и центральных разностей (8.25) от аналитического результата в зависимости от величины шага ϵ . Здесь красная кривая показывает график ошибки между численным вычислением градиента с помощью конечных разностей (8.24) и аналитическим результатом как функцию от ϵ . По мере уменьшения ϵ график сначала показывает линейное уменьшение ошибки, и это соответствует степенному закону, поскольку здесь логарифмические оси. Наклон этой линии равен 1, и это показывает, что ошибка ведет себя как $\mathcal{O}(\epsilon)$. В какой-то момент оцененный градиент достигает предела численного округления, и дальнейшее уменьшение приводит к зашумленной линии, которая снова следует степенному закону, но теперь ошибка увеличивается с уменьшением ϵ . Синяя кривая показывает соответствующий результат для центральных разностей (8.25). Ошибка гораздо меньше, чем в случае конечных разностей, а наклон линии равен 2, что свидетельствует о том, что ошибка равна $\mathcal{O}(\epsilon^2)$.

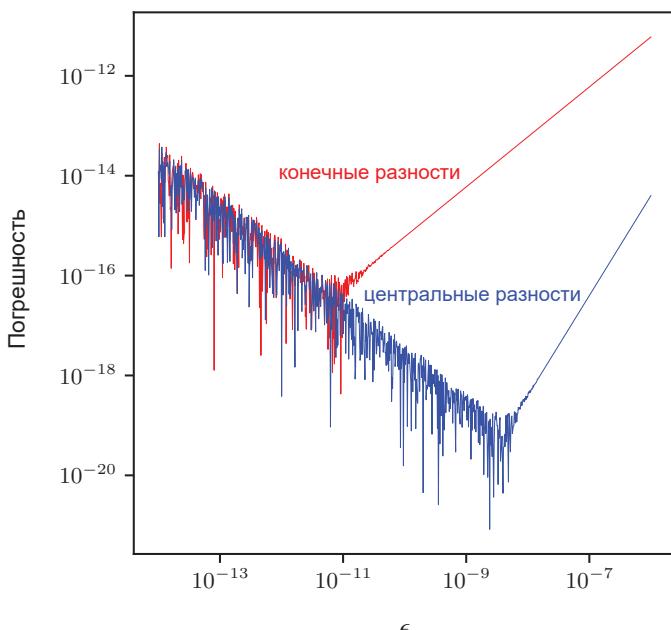


РИС. 8.2 График зависимости погрешности численной оценки градиента с помощью конечных и центральных разностей от аналитического результата в зависимости от шага ϵ

Основная проблема численного дифференцирования заключается в потере очень желательного масштабирования $\mathcal{O}(W)$. Каждое прямое распространение требует $\mathcal{O}(W)$ шагов, и в сети имеется W весов, каждый из которых необходимо корректировать по отдельности, так что общие вычислительные затраты составляют $\mathcal{O}(W^2)$.

Однако на практике численное дифференцирование может быть очень полезным, так как сравнение производных, вычисленных при прямой реализации обратного распространения или при автоматическом дифференцировании, с производными, полученными с помощью центральных разностей, дает возможность проверить правильность работы программного обеспечения.

8.1.5. Матрица Якоби

Ранее с помощью обратного распространения ошибок по сети были получены производные функции ошибки по отношению к весам. Обратное распространение можно использовать и для вычисления других производных. Далее рассмотрим оценку матрицы Якоби, элементы которой представляют собой производные выходов сети по отношению к входам:

$$J_{ki} \equiv \frac{\partial y_k}{\partial x_i}, \quad (8.26)$$

где каждая такая производная оценивается при фиксированных значениях всех остальных входов. Матрицы Якоби играют полезную роль в системах из нескольких отдельных элементов, как показано на рис. 8.3. Каждый из этих элементов может состоять из фиксированной или обучаемой функции, которая может быть линейной или нелинейной, при условии, что она дифференцируема.

Предположим, что необходимо минимизировать функцию ошибки E относительно параметра w на рис. 8.3. Производная функции ошибки определяется как

$$\frac{\partial E}{\partial w} = \sum_{k,j} \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial w}, \quad (8.27)$$

где матрица Якоби для красного модуля на рис. 8.3 представлена в виде среднего члена в правой части.

Поскольку матрица Якоби обеспечивает меру локальной чувствительности выходов к изменениям каждой из входных переменных, она также позволяет распространять любые известные ошибки Δx_i , связанные с входами, по обученной сети, что позволяет оценить их вклад Δy_k в ошибки на выходах, через соотношение:

$$\Delta y_k \approx \sum_i \frac{\partial y_k}{\partial x_i} \Delta x_i, \quad (8.28)$$

при этом предполагается, что $|\Delta x_i|$ невелики. В общем случае отображение сети, представленное обученной нейронной сетью, будет нелинейным, и поэтому элементы матрицы Якоби будут не постоянными, а зависимыми от конкретного используемого входного вектора. Получается, что (8.28) справедливо только для небольших изменений входных сигналов, а матрица Якоби должна быть заново оценена для каждого нового входного вектора.

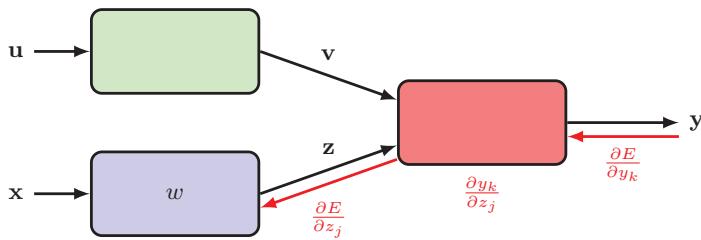


РИС. 8.3 Иллюстрация модульной архитектуры глубокого обучения, где матрица Якоби может быть использована для обратной передачи сигналов об ошибках с выходов на более ранние элементы системы

Матрицу Якоби можно оценить с помощью процедуры обратного распространения, подобной той, что была выведена ранее для оценки производных функции ошибки по весам. Начнем с того, что запишем элемент J_{ki} в виде

$$\begin{aligned} J_{ki} &= \frac{\partial y_k}{\partial x_i} = \sum_j \frac{\partial y_k}{\partial a_j} \frac{\partial a_j}{\partial x_i} \\ &= \sum_j w_{ji} \frac{\partial y_k}{\partial a_j}, \end{aligned} \quad (8.29)$$

где используется (8.5). Сумма в (8.29) выполняется по всем элементам j , к которым входной элемент i посылает связи (например, по всем элементам первого скрытого слоя в рассмотренной ранее слоистой топологии). Теперь запишем рекурсивную формулу обратного распространения для производных $\partial y_k / \partial a_j$:

$$\begin{aligned} \frac{\partial y_k}{\partial a_j} &= \sum_l \frac{\partial y_k}{\partial a_l} \frac{\partial a_l}{\partial a_j} \\ &= h'(a_j) \sum_l w_{lj} \frac{\partial y_k}{\partial a_j}, \end{aligned} \quad (8.30)$$

где сумма охватывает все элементы l , которым элемент j посылает соединения (соответствующие первому индексу w_{lj}). Опять же, здесь используется (8.5) и (8.6). Обратное распространение начинается с выходных элементов, для которых необходимые производные могут быть найдены непосредственно из функциональной формы функции активации выходного элемента. Для линейных выходных устройств получается:

$$\frac{\partial y_k}{\partial a_l} = \delta_{kl}, \quad (8.31)$$

где δ_{kl} – это элементы матрицы тождества, которые определяются как

$$\delta_{kl} = \begin{cases} 1, & \text{если } k = l, \\ 0 & \text{в противном случае.} \end{cases} \quad (8.32)$$

Если для каждого выходного элемента имеются отдельные логистические сигмоидные функции активации (раздел 3.4), то

$$\frac{\partial y_k}{\partial a_l} = \delta_{kl} \sigma'(a_l), \quad (8.33)$$

тогда как для выходов softmax (раздел 3.4) получается:

$$\frac{\partial y_k}{\partial a_l} = \delta_{kl} y_k - y_k y_l. \quad (8.34)$$

Процедуру вычисления матрицы Якоби можно описать следующим образом. Задаем входной вектор, соответствующий точке в пространстве входов, в которой должна быть оценена матрица Якоби, и проводим прямое распространение обычным способом для получения состояний всех скрытых и выходных элементов сети. Затем для каждой строки k матрицы Якоби, соответствующей выходному элементу k , выполняем обратное распространение с помощью рекурсивного соотношения (8.30), начиная с (8.31), (8.33) или (8.34), для всех скрытых элементов сети. Наконец, используем (8.29) для обратного распространения ко входам. Для оценки матрицы Якоби также можно использовать альтернативный формализм прямого распространения, который выводится по аналогии с приведенным здесь подходом обратного распространения (см. упражнение 8.5).

Реализацию таких алгоритмов можно проверить численным дифференцированием в форме

$$\frac{\partial y_k}{\partial x_i} = \frac{y_k(x_i + \epsilon) - y_k(x_i - \epsilon)}{2\epsilon} + \mathcal{O}(\epsilon^2), \quad (8.35)$$

что предполагает двумерные проходы прямого распространения для сети с D входами и, следовательно, требует $\mathcal{O}(DW)$ шагов в целом.

8.1.6. Матрица Гессе

Ранее уже приводились примеры использования обратного распространения для получения первых производных функции ошибки по отношению к весам сети. Обратное распространение также может быть использовано для оценки вторых производных ошибки, которые определяются как

$$\frac{\partial^2 E}{\partial w_j \partial w_k}. \quad (8.36)$$

Зачастую удобнее рассматривать все параметры веса и смещения как элементы w_i единичного вектора \mathbf{w} , и в этом случае вторые производные образуют элементы H_{ij} матрицы Гессе (гессиан) \mathbf{H} :

$$H_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}, \quad (8.37)$$

где $i, j \in \{1, \dots, W\}$, а W – это общее число весов и смещений. Матрица Гессе используется в некоторых алгоритмах нелинейной оптимизации, применяемых для обучения нейронных сетей, что обусловлено свойствами второго порядка поверхности ошибки (Bishop, 2006). Она также имеет значение в некоторых байесовских подходах к нейронным сетям (MacKay, 1992; Bishop, 2006) и используется для снижения точности весов в больших языковых моделях для сокращения объема занимаемой ими памяти (Shen et al., 2019).

Важным моментом для многих применений матрицы Гессе является эффективность ее оценки. Если в сети имеется W параметров (весов и смещений), то матрица Гессе имеет размерность $W \times W$, и поэтому вычислительные усилия для ее оценки будут иметь масштаб $\mathcal{O}(W^2)$ для каждой точки набора данных. Расширение процедуры обратного распространения (Bishop, 1992) позволяет эффективно оценивать матрицу Гессе с масштабом, который действительно равен $\mathcal{O}(W^2)$ (см. упражнение 8.6). Иногда матрица Гессе не нужна в явном виде, а требуется только произведение $\mathbf{v}^T \mathbf{H}$ этой матрицы на некоторый вектор \mathbf{v} , и это произведение может быть эффективно вычислено за $\mathcal{O}(W)$ шагов с помощью расширения обратного распространения (Møller, 1993; Pearlmutter, 1994).

Поскольку нейронные сети могут содержать миллионы или даже миллиарды параметров, вычисление или даже просто хранение полной матрицы Гессе для многих моделей является невыполнимой задачей. Оценка обратной матрицы Гессе еще более требовательна, так как она характеризуется вычислительным масштабом $\mathcal{O}(W^2)$. В связи с этим возникает интерес к поиску эффективных приближений к полной матрице Гессе.

Одна из аппроксимаций предполагает простую оценку только диагональных элементов матрицы Гессе и неявное обращение в ноль элементов вне диагонали. Это требует хранилища размером $\mathcal{O}(W)$ и позволяет оценить обратную величину за $\mathcal{O}(W)$ шагов, но все равно требует $\mathcal{O}(W^2)$ вычислений (Ricotti, Ragazzini and Martinelli, 1988), хотя при дальнейшей аппроксимации это может быть уменьшено до $\mathcal{O}(W)$ шагов (Becker and LeCun, 1989; LeCun, Denker and Solla, 1990). На практике тем не менее матрица Гессе обычно имеет значимые члены вне диагонали, поэтому к такому приближению следует относиться с осторожностью.

Более убедительный подход, известный как *приближение внешнего произведения* (*outer product approximation*), получается следующим образом. Рас-

смотрим задачу регрессии с использованием функции ошибки суммы квадратов вида

$$E = \frac{1}{2} \sum_{n=1}^N (y_n - t_n)^2, \quad (8.38)$$

где для простоты обозначений рассматривается один выход (расширение на несколько выходов не представляет сложности) (см. упражнение 8.8). Тогда матрицу Гессе можно записать в виде

$$\mathbf{H} = \nabla \nabla E = \sum_{n=1}^N \nabla y_n (\nabla y_n)^T + \sum_{n=1}^N (y_n - t_n) \nabla \nabla y_n, \quad (8.39)$$

где ∇ обозначает градиент относительно w . Если сеть была обучена на наборе данных и ее выходы y_n очень близки к целевым значениям t_n , то последний член в (8.39) будет мал, и им можно пренебречь. Однако в более общем случае может оказаться целесообразным пренебречь этим членом, исходя из следующего аргумента. Вспомним из раздела 4.2, что оптимальной функцией, минимизирующей потери от суммы квадратов, является условное среднее значение целевых данных. Тогда величина $(y_n - t_n)$ является случайной переменной с нулевым средним значением. Если предположить, что ее значение не коррелирует со значением второго производного члена в правой части (8.39), то при суммировании по n весь член будет в среднем равен нулю (см. упражнение 8.9).

Пренебрегая вторым членом в (8.39), получаем *приближение Левенберга–Марквардта* (*Levenberg–Marquardt approximation*), известное также как *приближение внешнего произведения* (*outer product approximation*), поскольку матрица Гессе строится из суммы внешних произведений векторов:

$$\mathbf{H} \approx \sum_{n=1}^N \nabla a_n \nabla a_n^T. \quad (8.40)$$

Вычисление аппроксимации внешнего произведения для матрицы Гессе является простым процессом, поскольку в нем участвуют только первые производные функции ошибки, и они могут быть эффективно вычислены за $\mathcal{O}(W)$ шагов с помощью стандартного обратного распространения. Элементы матрицы могут быть найдены за $\mathcal{O}(W^2)$ шагов путем простого умножения. Важно подчеркнуть, что это приближение, скорее всего, будет справедливо только для сети, которая была обучена надлежащим образом, и что для общего отображения сети вторые производные в правой части (8.39), как правило, не будут пренебрежимо малыми.

Для функции ошибки перекрестной энтропии для сети с логистическими сигмоидными функциями активации выходного элемента (см. упражнение 8.10) соответствующая аппроксимация имеет вид:

$$\mathbf{H} \approx \sum_{n=1}^N y_n (1 - y_n) \nabla a_n \nabla a_n^T. \quad (8.41)$$

Аналогичный результат можно получить для сетей с несколькими классами, имеющих функции активации выходного элемента softmax (упражнение 8.11). Аппроксимация внешним произведением также может быть использована для создания эффективной последовательной процедуры аппроксимации обратной матрицы Гессе (см. упражнение 8.12) (Hassibi and Stork, 1993).

8.2. Автоматическое дифференцирование

Ранее уже отмечалась важность использования информации о градиенте для эффективного обучения нейронных сетей (см. раздел 7.2.1). На самом деле существует четыре способа оценки градиента функции ошибки нейронной сети.

Первый подход, который на протяжении многих лет был основным в нейронных сетях, заключается в ручном составлении уравнений обратного распространения и последующей их программной реализации в явном виде. При тщательном исполнении это позволяет получить эффективный код, дающий точные результаты с высокой степенью точности. Однако процесс выведения уравнений и их кодирования занимает много времени, а также сопряжен с ошибками. Это также приводит к некоторой избыточности кода, поскольку уравнения прямого распространения кодируются отдельно от уравнений обратного распространения. Поскольку они зачастую содержат дублирующиеся вычисления, то при изменении модели требуется изменить и прямую, и обратную реализацию. Эти усилия могут быстро стать ограничением возможностей быстрого и эффективного эмпирического исследования различных архитектур.

Второй подход предполагает численную оценку градиентов с помощью конечных разностей (см. раздел 8.1.4). Для этого требуется только программная реализация уравнений прямого распространения. Одна из проблем численного дифференцирования заключается в ограниченной вычислительной точности, однако для обучения сети это вряд ли станет проблемой, поскольку в этом случае можно использовать стохастический градиентный спуск, при котором каждая оценка является лишь очень шумной оценкой локального градиента. Главный недостаток этого подхода заключается в его слабой масштабируемости по мере увеличения размера сети. Однако этот метод полезен для отладки других подходов, поскольку градиенты оцениваются только с помощью кода прямого распространения и могут быть использованы для подтверждения корректности обратного распространения или другого кода, используемого для оценки градиентов.

Третий подход называется *символическим дифференцированием* (*symbolic differentiation*). Он основан на использовании специализированного программного обеспечения для автоматизации аналитических манипуляций, которые в первом методе выполняются вручную. Такой процесс является воплощением принципов *компьютерной алгебры* или *символьных вычисле-*

ний. Он предполагает автоматическое применение правил исчисления, таких как правило цепочки, в полностью автоматизированном процессе. Затем полученные выражения реализуются в обычном программном обеспечении. Очевидным преимуществом такого подхода является предотвращение человеческих ошибок, связанных с ручным выводом уравнений обратного распространения. Более того, градиенты всегда рассчитываются с машинной точностью, что позволяет избежать плохого масштабирования при численном дифференцировании. Однако главный недостаток символьного дифференцирования заключается в том, что получаемые выражения для производных могут стать экспоненциально длиннее исходной функции, что, соответственно, требует длительного времени на вычисления. Рассмотрим функцию $f(x)$, заданную произведением $u(x)$ и $v(x)$. Функция и ее производная определяются как

$$f(x) = u(x)v(x), \quad (8.42)$$

$$f'(x) = u'(x)v(x) + u(x)v'(x). \quad (8.43)$$

Здесь наблюдаются избыточные вычисления, поскольку для вычисления $f(x)$ и $f'(x)$ нужно оценить $u(x)$ и $v(x)$. Если коэффициенты $u(x)$ и $v(x)$ включают в себя коэффициенты, в итоге получается вложенное дублирование выражений, которое стремительно усложняется. Эту проблему называют «разбуханием выражений» (*expression swell*).

В качестве следующей иллюстрации процесса рассмотрим функцию, которая по своей структуре напоминает два слоя нейронной сети (Grosse, 2018) с одним входом x , скрытым блоком с активацией z и выходом y , где

$$z = h(w_1x + b_1), \quad (8.44)$$

$$y = h(w_2z + b_2), \quad (8.45)$$

при этом $h(a)$ – это мягкий ReLU:

$$\zeta(a) = \ln(1 + \exp(a)). \quad (8.46)$$

Таким образом, общая функция задается в виде

$$y(x) = h(w_2h(w_1x + b_1) + b_2), \quad (8.47)$$

а производная выхода сети по w_1 , оцененная символическим способом (см. упражнение 8.13), имеет вид:

$$\frac{\partial y}{\partial w_1} = \frac{w_2x \exp(w_1x + b_1 + b_2 + w_2 \ln[1 + e^{w_1x + b_1}])}{(1 + e^{w_1x + b_1})(1 + \exp(b_2 + w_2 \ln[1 + e^{w_1x + b_1}])).} \quad (8.48)$$

Помимо того что эта функция значительно сложнее исходной, здесь также наблюдаются избыточные вычисления, когда такие выражения, как $w_1x + b_1$, встречаются в нескольких местах.

Еще один существенный недостаток символьного дифференцирования заключается в необходимости выразить дифференцируемое выражение в закрытой форме. Поэтому оно исключает такие важные операции потока управления, как циклы, рекурсии, условное выполнение и вызовы процедур, т. е. важные конструкции, которые могут понадобиться для определения сетевой функции.

Поэтому обратимся к четвертому методу оценки производных в нейронных сетях – *автоматическому дифференцированию* (*automatic differentiation*), также известному как «автодифференцирование» (*autodiff*) или «алгоритмическое дифференцирование» (*algorithmic differentiation*) (Baydin et al., 2018). В отличие от символьного дифференцирования задачей автоматического дифференцирования является не поиск математического выражения для производных, а автоматическая генерация программного обеспечения для вычисления градиента при наличии только кода уравнений прямого распространения. Как и в случае с символьным дифференцированием, такой код обеспечивает высокую точность вычислений, однако при этом он более эффективен, поскольку дает возможность использовать промежуточные переменные, которые применяются при определении уравнений прямого распространения, и тем самым позволяет избежать избыточных вычислений. Важно отметить, что автоматическое дифференцирование может работать не только с обычными математическими выражениями в закрытой форме, но и с элементами управления потоком, такими как ветвления, циклы, рекурсия и вызовы процедур, и тем самым является значительно более мощным, чем символьическое дифференцирование. Автоматическое дифференцирование – это устоявшаяся и широко используемая технология, которая была разработана в основном за пределами сообщества машинного обучения. Современное глубокое обучение – это в значительной степени эмпирический процесс, включающий оценку и сравнение различных архитектур, поэтому автоматическое дифференцирование играет ключевую роль в обеспечении точного и эффективного проведения таких исследований.

Основная идея автоматического дифференцирования заключается в использовании кода, оценивающего функцию, например, уравнений прямого распространения для оценки функции ошибки нейронной сети, и включении в этот код дополнительных переменных, значения которых накапливаются в процессе выполнения программы для получения требуемых производных. Существует две основные формы автоматического дифференцирования, которые называются прямым и обратным режимом. Для начала рассмотрим прямой режим, который с концептуальной точки зрения несколько проще.

8.2.1. Прямой режим автоматического дифференцирования

При прямом режиме автоматического дифференцирования каждая промежуточная переменная z_i , известная как «первичная» (primal) переменная процесса оценки функции, например функции ошибки нейронной сети, допол-

няется дополнительной переменной, представляющей значение некоторой производной от этой переменной, которую принято обозначать \dot{z}_i и которая известна как «касательная» (tangent) переменная. Касательные переменные и связанный с ними код генерируются автоматически программным окружением. Вместо простой прямой передачи для вычисления $\{z_i\}$ этот код распространяет кортежи данных (z_i, \dot{z}_i) таким образом, что переменные и производные оцениваются в параллельном режиме. Исходная функция обычно определяется в виде элементарных операторов, состоящих из арифметических операций и отрицания, а также трансцендентных функций, таких как экспоненциальная, логарифмическая и тригонометрическая, каждая из которых имеет простые формулы для своих производных. Использование этих производных в сочетании с цепным правилом вычислений дает возможность автоматически построить программный код для оценки градиентов.

В качестве примера рассмотрим следующую функцию с двумя входными переменными:

$$f(x_1, x_2) = x_1 x_2 + \exp(x_1 x_2) - \sin(x_2). \quad (8.49)$$

При реализации в программе код состоит из последовательности операций, которые можно выразить как *след оценки* (*evaluation trace*) базовых элементарных операций. Этот след можно представить в виде графа, как показано на рис. 8.4. Здесь определены следующие первичные переменные:

$$v_1 = x_1, \quad (8.50)$$

$$v_2 = x_2, \quad (8.51)$$

$$v_3 = v_1 v_2, \quad (8.52)$$

$$v_4 = \sin(v_2), \quad (8.53)$$

$$v_5 = \exp(v_3), \quad (8.54)$$

$$v_6 = v_5 - v_4, \quad (8.55)$$

$$v_7 = v_5 + v_6. \quad (8.56)$$

Теперь предположим, что необходимо оценить производную $\partial f / \partial x_1$. Определим касательные переменные через $\dot{v}_i = \partial v_i / \partial x_1$. Выражения для их оценки можно построить автоматически с помощью цепного правила исчисления:

$$\dot{v}_i = \frac{\partial v_i}{\partial x_1} = \sum_{j \in \text{pa}(i)} \frac{\partial v_j}{\partial x_1} \frac{\partial v_i}{\partial v_j} = \sum_{j \in \text{pa}(i)} \dot{v}_j \frac{\partial v_i}{\partial v_j}, \quad (8.57)$$

где $\text{pa}(i)$ обозначает множество *родителей* узла i на диаграмме трассировки оценок, т. е. множество переменных со стрелками, указывающими на узел i . Например, на рис. 8.4 родителями узла v_3 являются узлы v_1 и v_2 . Применяя (8.57) к уравнениям трассировки оценок (8.50)–(8.56), получим следующие уравнения трассировки оценок для касательных переменных:

$$\dot{v}_1 = 1, \quad (8.58)$$

$$\dot{v}_2 = 0, \quad (8.59)$$

$$\dot{v}_3 = v_1 \dot{v}_2 + \dot{v}_1 v_2, \quad (8.60)$$

$$\dot{v}_4 = \dot{v}_2 \cos(v_2), \quad (8.61)$$

$$\dot{v}_5 = \dot{v}_3 \exp(v_3), \quad (8.62)$$

$$\dot{v}_6 = \dot{v}_3 - \dot{v}_4, \quad (8.63)$$

$$\dot{v}_7 = \dot{v}_5 + \dot{v}_6. \quad (8.64)$$

Автоматическое дифференцирование для этого примера можно обобщить следующим образом. Сначала нужно написать код для оценки первичных переменных, заданных в (8.50)–(8.56). Связанные с ними уравнения и соответствующий код для оценки касательных переменных (8.58)–(8.64) генерируются автоматически. Для оценки производной $\partial f / \partial x_1$ необходимо ввести конкретные значения x_1 и x_2 , после чего код обрабатывает первичные и касательные уравнения, численно оценивая кортежи данных (v_i, \dot{v}_i) последовательно до получения \dot{v}_5 (см. упражнение 8.17), что и является требуемой производной.

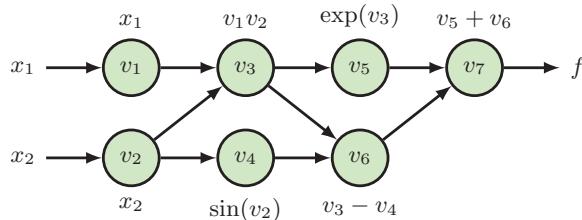


РИС. 8.4 Диаграмма оценки следов, показывающая этапы численной оценки функции (8.49) с помощью первичных уравнений (8.50)–(8.56)

Теперь рассмотрим пример с двумя выходами $f_1(x_1, x_2)$ и $f_2(x_1, x_2)$, где $f_1(x_1, x_2)$ определяется в (8.49) и

$$f_2(x_1, x_2) = (x_1 x_2 - \sin(x_2)) \exp(x_1 x_2), \quad (8.65)$$

как показано на диаграмме оценки на рис. 8.5. Здесь видно, что уравнения для первичной и касательной переменных требуют лишь небольшого расширения, поэтому и $\partial f_1 / \partial x_1$, и $\partial f_2 / \partial x_1$ могут быть оценены вместе за один проход. Однако недостатком этого способа является необходимость оценки производных по другой входной переменной x_2 , что требует отдельного прямого прохода.

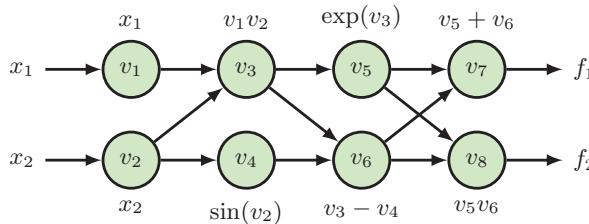


РИС. 8.5 Расширенный пример из рис. 8.4 для функции с двумя выходами f_1 и f_2

В целом если имеется функция с D входами и K выходами, то один проход автоматического дифференцирования в прямом режиме дает один столбец матрицы Якоби $K \times D$:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_K}{\partial x_1} & \dots & \frac{\partial f_K}{\partial x_D} \end{bmatrix}. \quad (8.66)$$

Для вычисления столбца j матрицы Якоби необходимо инициализировать прямой проход уравнений касательной, задав $\dot{x}_j = 1$ и $\dot{x}_i = 0$ при $i \neq j$. В векторной форме это можно записать как $\dot{\mathbf{x}} = \mathbf{e}_j$, где \mathbf{e}_j – это j -й единичный вектор. Для вычисления полной матрицы Якоби потребуется D проходов в прямом режиме. Однако если необходимо оценить произведение матрицы Якоби на вектор $\mathbf{r} = (r_1, \dots, r_D)^T$:

$$\mathbf{J} = \left[\begin{array}{ccc|c} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_D} & r_1 \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial f_K}{\partial x_1} & \dots & \frac{\partial f_K}{\partial x_D} & r_D \end{array} \right], \quad (8.67)$$

то это можно сделать за один прямой проход, задав $\dot{\mathbf{x}} = \mathbf{r}$ (см. упражнение 8.18).

Здесь видно, что автоматическое дифференцирование в прямом режиме позволяет оценить всю матрицу Якоби производных $K \times D$ с помощью D проходов в прямом режиме. Это очень эффективно для сетей с небольшим количеством входов и большим количеством выходов, таких как $K \gg D$. Однако зачастую приходится работать в режиме с единственной функцией, а именно функцией ошибки, используемой для обучения, и большим количеством переменных, которые требуется дифференцировать, включая веса и смещения в сети, которых могут быть миллионы или миллиарды. В таких ситуациях автоматическое дифференцирование в прямом режиме оказывается крайне неэффективным. Поэтому обратимся к альтернативному варианту автома-

тического дифференцирования, основанному на обратном потоке производных данных через граф оценки трассировки.

8.2.2. Обратный режим автоматического дифференцирования

Автоматическое дифференцирование в обратном режиме можно рассматривать как обобщение процедуры обратного распространения ошибки. Как и в прямом режиме, каждая промежуточная переменная v_i получит дополнительные переменные, которые в данном случае называются *смежными переменными* (*adjoint variables*) и обозначаются \bar{v}_i . Рассмотрим еще раз ситуацию с одиночной выходной функцией f , для которой смежные переменные определяются как

$$\bar{v}_i = \frac{\partial f}{\partial v_i}. \quad (8.68)$$

Их можно оценивать последовательно, начиная с выхода и двигаясь назад, применяя цепное правило вычислений:

$$\bar{v}_i = \frac{\partial f}{\partial v_i} = \sum_{j \in \text{ch}(i)} \frac{\partial f}{\partial v_j} \frac{\partial v_j}{\partial v_i} = \sum_{j \in \text{ch}(i)} \bar{v}_j \frac{\partial v_j}{\partial v_i}. \quad (8.69)$$

Здесь $\text{ch}(i)$ обозначает *дочерние* (*children*) элементы узла i в графе трассировки оценок, т. е. множество узлов, в которые ведут стрелки из узла i . Последовательная оценка смежных переменных представляет собой поток информации в обратном направлении по графу (рис. 8.1), как было показано ранее.

Если вновь обратиться к конкретному примеру функции, заданной в (8.50)–(8.56), то для оценки смежных переменных можно получить следующие уравнения (см. упражнение 8.16):

$$\bar{v}_7 = 1, \quad (8.70)$$

$$\bar{v}_6 = \bar{v}_7, \quad (8.71)$$

$$\bar{v}_5 = \bar{v}_7, \quad (8.72)$$

$$\bar{v}_4 = -\bar{v}_6, \quad (8.73)$$

$$\bar{v}_3 = \bar{v}_5 v_5 + \bar{v}_6, \quad (8.74)$$

$$\bar{v}_2 = \bar{v}_2 v_1 + \bar{v}_4 \cos(v_2), \quad (8.75)$$

$$\bar{v}_1 = \bar{v}_3 v_2. \quad (8.76)$$

Обратите внимание, что они начинаются на выходе, а затем идут в обратном направлении по графу к входам. Даже при наличии нескольких

входов для оценки производных требуется только один проход в обратном направлении. Для функции ошибки нейронной сети производные от E по весу и смещениям получаются как соответствующие смежные переменные. Однако если имеется более одного выхода, то для каждой выходной переменной необходимо выполнить отдельный проход в обратном направлении (см. рис. 8.5).

Обратный режим зачастую требует больше памяти, чем прямой, поскольку все промежуточные первичные переменные должны быть сохранены и доступны при оценке смежных переменных во время обратного прохода. На-против, в прямом режиме первичные и касательные переменные вычисляются вместе во время прямого прохода, и поэтому переменные могут быть отброшены после их использования. Поэтому реализация прямого режима в целом более проста по сравнению с обратным режимом.

Как для прямого, так и для обратного режима автоматического дифференцирования можно гарантировать, что один проход по сети потребует не более чем 6-кратного превышения вычислительных затрат на оценку одной функции. На практике эти затраты обычно ближе к двух- или трехкратному значению (Griewank and Walther, 2008). Интерес представляют также варианты гибридного использования прямого и обратного режимов. Например, при оценке произведения матрицы Гессе на вектор, что можно сделать без явной оценки полной матрицы Гессе (Pearlmutter, 1994). Здесь для вычисления градиента кода, который сам был сгенерирован прямой моделью, можно использовать обратный режим. Сначала задается вектор \mathbf{b} и точка \mathbf{x} , в которой должно быть вычислено векторное произведение матрицы Гессе. Установливая $\dot{\mathbf{x}} = \mathbf{v}$ и используя прямой режим, можно получить направленную производную $\mathbf{v}^T \nabla f$. Затем она дифференцируется в обратном режиме для получения $\nabla^2 f \mathbf{v} = \mathbf{Hv}$. Если W – это количество параметров в нейронной сети, то эта оценка имеет сложность $\mathcal{O}(W)$, несмотря на то что размер матрицы Гессе составляет $W \times W$. Сама матрица Гессе также может быть оценена явно с помощью автоматического дифференцирования, но в этом случае сложность составляет $\mathcal{O}(W^2)$.

Упражнения

- 8.1** (*) Используя (8.5), (8.6), (8.8) и (8.12), убедитесь, что формула обратного распространения (8.13) предназначена для оценки производных функции ошибки.
- 8.2** (**) Для сети с несколькими слоями перепишите формулу обратного распространения (8.13) в матричной форме, начав с уравнения прямого распространения (6.19). Обратите внимание, что результат включает в себя умножение на транспонирование матриц.
- 8.3** (*) Используя разложение Тейлора, докажите, что члены с коэффициентом $\mathcal{O}(\epsilon)$ в правой части (8.25) аннулируются.

- 8.4** (**) Рассмотрим двухслойную сеть в форме на рис. 6.9 с дополнительными параметрами, соответствующими связям с пропуском слоя, которые идут непосредственно от входов к выходам. Развивая идеи раздела 8.1.3, запишите уравнения для производных функции ошибки по этим дополнительным параметрам.
- 8.5** (****) В разделе 8.1.5 описана процедура оценки матрицы Якоби нейронной сети с использованием метода обратного распространения. Выполните альтернативный способ нахождения матрицы Якоби на основе уравнений прямого распространения.
- 8.6** (****) Рассмотрим двухслойную нейронную сеть. Определите величины:

$$\delta_k = \frac{\partial E_n}{\partial a_k}, \quad M_{kk'} \equiv \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}}. \quad (8.77)$$

Выполните выражения для элементов матрицы Гессе в выражениях δ_k и $M_{kk'}$ для элементов, в которых (i) оба веса находятся во втором слое, (ii) оба веса находятся в первом слое и (iii) один вес находится в каждом слое.

- 8.7** (****) Расширьте результаты упражнения 8.6 для точной матрицы Гессе двухслойной сети, чтобы включить в нее соединения пропущенных слоев, ведущие непосредственно от входов к выходам.
- 8.8** (**) Аппроксимация внешнего произведения матрицы Гессе для нейронной сети с помощью функции ошибки суммы квадратов приведена в (8.40). Расширьте этот результат для примера с несколькими выходами.
- 8.9** (**) Рассмотрим квадратичную функцию потерь вида

$$E(\mathbf{w}) = \frac{1}{2} \iint \{y(\mathbf{x}, \mathbf{w}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt, \quad (8.78)$$

где $y(\mathbf{x}, \mathbf{w})$ – это параметрическая функция, например нейронная сеть. Результат (4.37) показывает, что функция $y(\mathbf{x}, \mathbf{w})$, минимизирующая эту ошибку, задается условным ожиданием t по \mathbf{x} . Используйте этот результат для доказательства того, что вторая производная E по двум элементам w_r и w_s вектора \mathbf{w} задается как

$$\frac{\partial^2 E}{\partial w_r \partial w_s} = \int \frac{\partial y}{\partial w_r} \frac{\partial y}{\partial w_s} p(\mathbf{x}) d\mathbf{x}. \quad (8.79)$$

Обратите внимание, что для конечной выборки из $p(\mathbf{x})$ получится (8.40).

- 8.10** (**) Выполните выражение (8.41) для аппроксимации внешнего произведения матрицы Гессе для сети с одним выходом, с логистической сигмоидной функцией активации выходного элемента и функцией

ошибки перекрестной энтропии, соответствующее результату (8.40) для функции ошибки суммы квадратов.

- 8.11** (***) Выведите выражение внешнего произведения аппроксимации матрицы Гессе для сети с K выходами, функцией активации выходного звена softmax и функцией ошибки перекрестной энтропии, соответствующее результату (8.40) для функции ошибки суммы квадратов.

- 8.12** (****) Рассмотрим матричное тождество

$$(\mathbf{M} + \mathbf{v}\mathbf{v}^T)^{-1} = \mathbf{M}^{-1} - \frac{(\mathbf{M}^{-1}\mathbf{v})(\mathbf{v}^T\mathbf{M}^{-1})}{1 + \mathbf{v}^T\mathbf{M}^{-1}\mathbf{v}}, \quad (8.80)$$

которое является простым частным случаем тождества Вудбери (A.7). Применив (8.80) к приближению внешнего произведения (8.40) для матрицы Гессе, выведите формулу, которая позволяет вычислять обратную матрицу Гессе за один проход по обучающим данным и обновлять обратную матрицу Гессе с каждой точкой данных. Обратите внимание, что алгоритм можно инициализировать с помощью $\mathbf{H} = \alpha\mathbf{I}$, где α – это небольшая константа, и что результаты не особенно чувствительны к точному значению α .

- 8.13** (**) Убедитесь, что производная от (8.47) определяется в (8.48).
- 8.14** (**) Логистическая карта – это функция, определяемая итерационным соотношением $L_{n+1}(x) = 4L_n(x)(1 - \ln(x))$ при $L_1(x) = x$. Запишите уравнения следов оценки для $L_2(x)$, $L_3(x)$ и $L_4(x)$, а затем запишите выражения для соответствующих производных $L'_1(x)$, $L'_2(x)$, $L'_3(x)$ и $L'_4(x)$. Не упрощайте выражения, а просто обратите внимание на то, что сложность формул для производных растет гораздо быстрее, чем сложность выражений для самих функций.
- 8.15** (**) Исходя из уравнений трассировки оценок (8.50)–(8.56) для примера функции (8.49), используйте (8.57) для получения уравнений оценки касательной переменной в прямом режиме (8.58)–(8.64).
- 8.16** (**) Исходя из уравнений следа оценки (8.50)–(8.56) для примера функции (8.49) и ориентируясь на рис. 8.4, используйте (8.69) для получения уравнений оценки смежных переменных в обратном режиме (8.70)–(8.76).
- 8.17** (****) Рассмотрим пример функции (8.49). Запишите выражение для $\partial f / \partial x_1$ и оцените эту функцию для $x_1 = 1$ и $x_2 = 2$. Теперь используйте уравнения оценки следа (8.50)–(8.56) для оценки переменных v_1 – v_7 , а затем используйте уравнения оценки следа прямого режима автоматического дифференцирования для оценки касательных переменных \dot{v}_1 – \dot{v}_7 и убедитесь, что полученное значение $\partial f / \partial x_1$ совпадает с найденным

напрямую. Аналогично используйте уравнения оценки следа обратного автоматического дифференцирования (8.70)–(8.76) для оценки смежных переменных \bar{v}_7 – \bar{v}_1 и снова подтвердите, что полученное значение $\partial f / \partial x_1$ согласуется с найденным прямым путем.

- 8.18** (**) Выразив произвольный вектор $\mathbf{r} = (r_1, \dots, r_D)^T$ как линейную комбинацию единичных векторов \mathbf{e}_i , где $i = 1, \dots, D$, докажите, что произведение матрицы Якоби для функции от \mathbf{r} в форме (8.67) можно вычислить с помощью одного прохода автоматического дифференцирования в прямом режиме, задав $\dot{\mathbf{x}} = \mathbf{r}$.

Глава 9

Регуляризация

Концепция регуляризации была введена в этой книге при описании подгонки полиномиальных кривых (см. раздел 1.2) в качестве способа сокращения чрезмерной подгонки путем предотвращения принятия параметрами модели значений с большой величиной. Для этого к функции ошибки добавляется простое штрафное слагаемое, в результате чего получается регуляризованная функция ошибки вида

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}, \quad (9.1)$$

где \mathbf{w} – это вектор параметров модели, $E(\mathbf{w})$ – это нерегуляризованная функция ошибки, а гиперпараметр регуляризации λ управляет интенсивностью эффекта регуляризации. Улучшение точности прогнозирования с помощью такого регуляризатора можно рассматривать в контексте компромисса между смещением и дисперсией за счет уменьшения дисперсии полученного решения при некотором увеличении смещения (см. раздел 4.3). В этой главе регуляризация будет рассмотрена более подробно, при этом будут разобраны различные подходы к ее использованию. Также более подробно будет рассмотрена важная роль смещения в достижении хорошего обобщения по ограниченным наборам обучающих данных.

На практике крайне маловероятно, что процесс, генерирующий данные, будет точно соответствовать определенной архитектуре нейронной сети, поэтому любая нейронная сеть будет лишь приближенно отражать истинное состояние генератора данных. Более крупные сети могут обеспечить более точное приближение, но это сопряжено с риском чрезмерной подгонки. Поэтому на практике лучшие результаты обобщения почти всегда достигаются при использовании более крупной сети в сочетании с некоторой формой регуляризации. В этой главе рассматриваются различные альтернативные методы регуляризации, включая раннюю остановку, усреднение модели, прореживание, аугментацию данных и совместное использование параметров. При желании можно использовать несколько форм регуляризации в совокупности. Например, регуляризация функции ошибки в виде (9.1) часто используется вместе с прореживанием.

9.1. Индуктивное смещение

При сравнении ошибок предсказания полиномов различных порядков для задачи синтетических данных синусоидальной формы (см. раздел 1.2.4) было замечено, что наименьшая ошибка обобщения достигается при использовании полинома промежуточной сложности, не являющегося ни слишком простым, ни слишком гибким. Аналогичный результат был получен при использовании регуляризационного члена в форме (9.1) для управления сложностью модели (см. раздел 1.2.5), так как промежуточное значение коэффициента регуляризации λ давало наилучшие прогнозы для новых входных значений. Этот результат был получен при разложении смещения-вариации (см. раздел 4.3), где говорилось, что соответствующий уровень смещения в модели важен для обобщения по ограниченным наборам данных. Простые модели с высоким уровнем смещения не способны уловить вариации в основном процессе генерации данных, в то время как очень гибкие модели с низким уровнем смещения склонны к чрезмерной подгонке, что приводит к ухудшению обобщения. По мере увеличения размера набора данных появляется возможность использовать более гибкие модели с меньшей погрешностью без чрезмерной дисперсии, что приводит к улучшению обобщения. Обратите внимание, что в практических условиях на выбор модели также могут повлиять такие факторы, как количество памяти или скорость обработки данных. Далее в этой книге соображения о таких вспомогательных факторах игнорируются, и основное внимание будет уделено достижению хороших прогностических характеристик, т. е. качественному обобщению.

9.1.1. Обратные задачи

Выбор модели является ключевым моментом в машинном обучении. Это обусловлено тем, что большинство задач машинного обучения являются примерами обратных задач (*inverse problems*). Если задано условное распределение $p(t | \mathbf{x})$ и конечное множество входных точек $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, то, в принципе, несложно сделать выборку соответствующих значений $\{t_1, \dots, t_N\}$ из этого распределения. Однако в машинном обучении приходится решать обратную задачу, а именно определять все распределение по конечному числу выборок. По своей сути эта задача является поставленной некорректно (*ill-posed*), поскольку существует бесконечно много распределений, потенциально способных генерировать наблюдаемые обучающие данные. Фактически любое распределение, имеющее ненулевую плотность вероятности при наблюдаемых целевых значениях, является подходящим кандидатом.

Машинное обучение должно приносить пользу, но при этом необходимо делать предсказания для новых значений \mathbf{x} , и, следовательно, нужен способ выбора конкретного распределения из бесконечно большого числа вариантов. Предпочтение одного варианта перед другими называют *индуктивным смещением* (*inductive bias*), или *предварительным знанием* (*prior knowledge*), и оно играет важнейшую роль в машинном обучении. Предварительные

знания могут быть получены из фоновой информации, позволяющей ограничить пространство решений. Для многих задач небольшие изменения входных значений приводят к небольшим изменениям выходных значений, и потому стоит ориентироваться на решения с плавно меняющимися функциями. Условия регуляризации в форме (9.1) позволяют весам модели иметь меньшую величину и, следовательно, вносят смещение в сторону функций, которые меняются медленнее при изменении входных данных. Точно так же при распознавании объектов на изображениях можно ввести предварительное знание об идентичности объекта, как правило, не зависящее от его местоположения на изображении (см. главу 10). Это называется *инвариантность преобразования* (*translation invariance*), и включение этого фактора в состав решения может значительно упростить задачу построения системы с хорошим обобщением. Однако необходимо следить за отсутствием погрешностей и ограничений, которые не согласуются с основным процессом, генерирующим данные. Например, предположение о том, что связь между выходными и входными данными линейна, в то время как на самом деле существуют значительные нелинейности, может привести к созданию системы, дающей неправильные ответы.

Такие методы, как трансферное обучение и многозадачное обучение (см. раздел 6.3.4), также можно рассматривать в контексте регуляризации. Когда обучающие данные для конкретной задачи ограничены, можно использовать дополнительные данные из другой, но схожей задачи, чтобы помочь определить обучаемые параметры в нейронной сети. Предположение о сходстве между задачами представляет собой более сложную форму индуктивного смещения по сравнению с простой регуляризацией, и это объясняет улучшение производительности в результате использования дополнительных данных.

9.1.2. Теорема об отсутствии бесплатного обеда

Основное внимание в этой книге уделено важному классу моделей машинного обучения, называемых глубокими нейронными сетями. Это очень гибкие модели, которые произвели революцию во многих областях, включая компьютерное зрение, распознавание речи и обработку естественного языка. По сути, они стали основой для большинства приложений машинного обучения. Поэтому может показаться, что они представляют собой «универсальный» алгоритм обучения, способный решать любые задачи. Однако даже очень гибкие нейронные сети не лишены индуктивных смещений. Например, сверточные нейронные сети кодируют специфические формы индуктивных смещений (см. главу 10), такие как эквивариантность перевода, которые особенно полезны в задачах работы с изображениями.

Теорема об отсутствии бесплатного обеда (*no free lunch theorem*) (Wolpert, 1996), получившая свое название от крылатого выражения «бесплатных обедов не бывает» (*there's no such thing as a free lunch*), утверждает, что каждый

алгоритм обучения так же хорош, как и любой другой, если рассматривать его усредненно по всем возможным проблемам. Если конкретная модель или алгоритм лучше среднего при решении одних задач, то в других они наверняка будут хуже среднего. Впрочем, это, скорее, теоретическое рассуждение, поскольку пространство возможных задач включает взаимосвязи между входом и выходом, которые были бы совершенно нехарактерны для любого приемлемого практического применения. Например, как уже отмечалось, большинство примеров, представляющих практический интерес, демонстрируют некоторую степень слаженности, при которой небольшие изменения входных значений связаны по большей части с небольшими изменениями целевых значений. Такие модели, как нейронные сети и, собственно, большинство широко используемых методов машинного обучения, демонстрируют такую форму индуктивного смещения, и именно поэтому в определенной степени имеют широкое распространение.

Хотя теорема об отсутствии бесплатного обеда является в определенной степени теоретической, она подчеркивает центральное значение смещения в определении эффективности алгоритма машинного обучения. Невозможно обучаться «исключительно на данных» в отсутствие какого-либо смещения. На практике смещение может быть неявным. Например, каждая нейронная сеть имеет конечное число параметров, что ограничивает круг представляемых ею функций. Однако смещение может быть закодировано и в явном виде, отражая предшествующие знания о специфике конкретной решаемой задачи.

В реальности при попытке найти алгоритмы обучения общего назначения приходится искать индуктивные смещения, подходящие для широкого класса практических приложений. Однако для каждой конкретной задачи можно получить лучшие результаты, если включить в алгоритм более сильные индуктивные смещения, характерные именно для этого случая. Концепция *машинного обучения с использованием моделей* (*model-based machine learning*) (Winn et al., 2023) утверждает, что все предположения в моделях машинного обучения должны быть явными, чтобы была возможность выбрать подходящий вариант индуктивных смещений.

Ранее уже говорилось о том, что индуктивное смещение может быть учтено в форме распределения, например, если указать, что выход является линейной функцией фиксированного набора определенных базисных функций. Оно также может быть учтено за счет добавления члена регуляризации к функции ошибки, используемой во время обучения. Еще один способ контролировать сложность нейронной сети – это сам процесс обучения (см. главу 7). Далее будет рассказано о том, что глубокие нейронные сети могут давать хорошие обобщения, даже если количество настраиваемых параметров превышает количество точек обучающих данных, при условии правильной настройки процесса обучения. Отчасти навыки применения глубокого обучения для решения реальных задач связаны с тщательной разработкой индуктивных смещений и использованием полученных ранее знаний.

9.1.3. Симметрия и инвариантность

Во многих приложениях машинного обучения прогнозы не должны изменяться, т. е. быть *инвариантными* (*invariant*), даже в случае одного или нескольких преобразований входных переменных. Например, при классификации объектов двумерных изображений, таких как «кошка» или «собака», определенному объекту должна быть присвоена одна и та же классификация независимо от его положения на изображении. Это называется *инвариантностью перевода*, или *трансляционной инвариантностью* (*translation invariance*). Точно так же при изменении размера объекта на изображении его классификация должна оставаться неизменной. Это называется *масштабной инвариантностью* (*scale invariance*). Использование таких симметрий для создания индуктивных смещений может значительно улучшить производительность моделей машинного обучения и поэтому является предметом изучения *глубокого геометрического обучения* (*geometric deep learning*) (Bronstein et al., 2021).

Преобразования, такие как перевод или масштабирование, при которых определенные свойства остаются неизменными, представляют собой *симметрии* (*symmetries*). Совокупность всех преобразований, соответствующих определенной симметрии, образует математическую структуру, называемую *группой* (*group*). Группа состоит из множества элементов $\mathcal{A}, \mathcal{B}, \mathcal{C}, \dots$ и бинарной операции объединения пар элементов, которую принято обозначать $\mathcal{A} \circ \mathcal{B}$. Для группы справедливы следующие четыре аксиомы:

- 1) **замкнутость (closed)**: для любых двух элементов \mathcal{A}, \mathcal{B} в этом наборе также должна быть пара $\mathcal{A} \circ \mathcal{B}$;
- 2) **ассоциативность (associative)**: для любых трех элементов $\mathcal{A}, \mathcal{B}, \mathcal{C}$ в наборе справедливо $(\mathcal{A} \circ \mathcal{B}) \circ \mathcal{C} = \mathcal{A} \circ (\mathcal{B} \circ \mathcal{C})$;
- 3) **тождественность (identity)**: в наборе имеется элемент \mathcal{I} , называемый тождеством, со свойством: $\mathcal{A} \circ \mathcal{I} = \mathcal{I} \circ \mathcal{A} = \mathcal{A}$ для каждого элемента \mathcal{A} в этом наборе;
- 4) **инверсия (inverse)**: для каждого элемента \mathcal{A} набора существует другой элемент набора, обозначенный через \mathcal{A}^{-1} , который называется обратным (*inverse*) и который обладает свойством $\mathcal{A} \circ \mathcal{A}^{-1} = \mathcal{A}^{-1} \circ \mathcal{A} = \mathcal{I}$.

Простыми примерами групп являются набор поворотов квадрата на кратные 90° (см. упражнение 9.1) или набор непрерывных перемещений объекта в двумерной плоскости.

Инвариантность предсказаний нейронной сети относительно преобразований входного пространства, в принципе, можно узнать из данных, не внося никаких особых изменений в сеть или процедуру обучения. Однако на практике это может оказаться чрезвычайно сложной задачей, поскольку такие преобразования могут привести к существенным изменениям в исходных данных. Например, относительно небольшое перемещение объекта на изображении, даже на несколько пикселей, может привести к значительному изменению значений пикселей. Кроме того, зачастую одновременно должны выполняться несколько инвариантных преобразований, например

инвариантность к переводам в двух измерениях, а также масштабирование, поворот, изменение интенсивности, изменение цветового баланса и многие другие. Существует экспоненциально много возможных комбинаций таких преобразований, что делает размер обучающего набора, необходимого для изучения всех этих инвариантов, непомерно большим.

Поэтому приходится искать более эффективные подходы для стимулирования адаптивной модели к проявлению требуемых параметров инвариантности. В целом их можно разделить на четыре категории.

1. **Предварительная обработка (pre-processing).** Инвариантности за-кладываются на этапе предварительной обработки путем вычисления характеристик данных, которые инвариантны при требуемых преоб-разованиях. Любая последующая система регрессии или классификации, использующая такие признаки в качестве входных данных, в обязательном порядке также будет соблюдать эти условия инвари-антности.
2. **Регуляризованная функция ошибки (regularized error function).** В функцию ошибки добавляется член регуляризации, который призван повлиять на изменения в выходных данных модели, когда входные данные подвергаются одному из инвариантных преобразований.
3. **Наращивание данных (data augmentation).** Обучающий набор рас-ширяется с помощью копий точек обучающих данных, преобразо-ванных в соответствии с требуемыми значениями инвариантности и имеющих те же целевые значения на выходе, что и примеры без преобразований.
4. **Архитектура сети (network architecture).** Свойства инвариантности встраиваются в структуру нейронной сети с помощью соответствую-щего выбора архитектуры сети.

Одна из проблем подхода № 1 заключается в создании функций, облада-ющих требуемой инвариантностью, не отбрасывая при этом информацию, которая может быть полезна для определения выходов сети. Ранее уже от-мечалось, что фиксированные, созданные вручную признаки располагают ограниченными возможностями, и теперь их вытесняют обучаемые пред-ставления, полученные с помощью глубоких нейронных сетей (см. главу 6).

Примером подхода № 2 является техника *касательного распространения* (*tangent propagation*) (Simard et al., 1992), когда в процессе обучения к функции ошибки добавляется член регуляризации. Этот член напрямую штрафует за изменения в выходных данных, возникающие в результате изменений во входных переменных, которые соответствуют одному из инвариантных преобразований. Ограничением этого метода, помимо дополнительной сложности обучения, является способность работать только с небольшими преобразованиями (например, с перемещениями не более чем на пиксель).

Подход № 3 известен как метод *дополнения набора данных* (*data set augmentation*). Зачастую он относительно прост в реализации и может оказаться очень эффективным на практике. Его часто применяют в контексте анализа

изображений, поскольку он позволяет напрямую создавать преобразованные обучающие данные. На рис. 9.1 показаны примеры таких преобразований для изображения кота. Для медицинских изображений мягких тканей дополнение данных может также включать непрерывные деформации методом «резинового листа» (*rubber sheet*) (Ronneberger, Fischer and Brox, 2015).

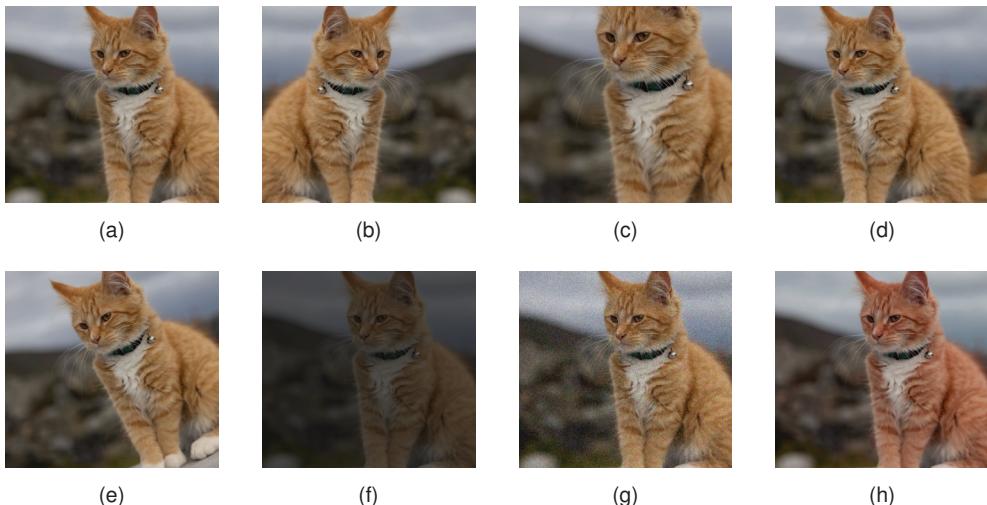


РИС. 9.1 Иллюстрация увеличения набора данных: (a) исходное изображение, (b) инверсия по горизонтали, (c) масштабирование, (d) преобразование, (e) поворот, (f) изменение яркости и контрастности, (g) аддитивный шум и (h) изменение цвета

Для алгоритмов последовательного обучения, таких как стохастический градиентный спуск, набор данных может быть расширен за счет преобразования каждой точки входных данных перед ее подачей на модель таким образом, чтобы при повторном использовании точек данных каждый раз применялось другое преобразование (из соответствующего распределения). Для пакетных методов аналогичного эффекта можно добиться с помощью многократной репликации каждой точки данных и независимого преобразования каждой копии.

Эффект от использования дополненных данных можно проанализировать на примере преобразований, которые представляют собой небольшие изменения исходных примеров, а затем выполнить разложение Тейлора для функции ошибки по величине преобразования (Bishop, 1995c; Leen, 1995; Bishop, 2006). Это приводит к регуляризованной функции ошибки, в которой регуляризатор штрафует градиент выхода сети по отношению к входным переменным, спроектированным на направление преобразования. Это связано с рассмотренной выше техникой распространения по касательной. Особый случай связан с тем, что преобразование входных переменных сводится к добавлению случайного шума, и в этом случае регуляризатор штрафует производные выходов сети по отношению к входам (см. упражнение 9.2). И в этом

случае это вполне оправдано с практической точки зрения, поскольку необходимо стремиться к тому, чтобы выходы сети оставались неизменными, несмотря на добавление шума к входным переменным.

Наконец, подход № 4, где встраивание инвариантности в структуру сети оказывается очень мощным и эффективным, а также обеспечивает ряд других важных преимуществ. Этот подход будет подробно рассмотрен в контексте сверточных нейронных сетей для компьютерного зрения (см. главу 10).

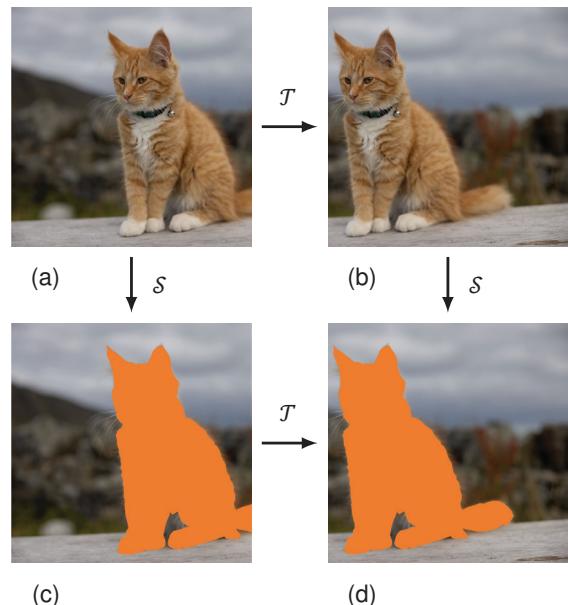
9.1.4. Эквивариантность

Обобщение инвариантности носит название *эквивариантности* (*equivariance*), когда выход сети вместо сохранения своей неизменности при преобразовании входа сам преобразуется каким-либо определенным образом. Например, рассмотрим сеть, которая принимает на вход изображение и выдает сегментацию этого изображения, где каждый пиксель классифицируется по принадлежности либо к объекту переднего плана, либо к фону. В этом случае, если местоположение объекта на изображении изменяется, необходимо, чтобы соответствующая сегментация объекта изменилась соответствующим образом. Пусть изображение обозначено через \mathbf{I} , а операция сети сегментации через \mathcal{S} . Тогда для операции перевода \mathcal{T} справедливо выражение

$$\mathcal{S}(\mathcal{T}(\mathbf{I})) = \mathcal{T}(\mathcal{S}(\mathbf{I})), \quad (9.2)$$

что указывает на тот факт, что сегментация преобразованного изображения задается преобразованием сегментации исходного изображения. Это проиллюстрировано на рис. 9.2.

РИС. 9.2 Иллюстрация эквивариантности для (9.2). Если изображение (a) сначала преобразуется в (b), а затем сегментируется в (d), то результат будет таким же, как и в случае сегментации изображения в (c), а затем преобразования в (d)



В более общем случае эквивариантность возможна в случае, если преобразование, применяемое к выходу, отличается от преобразования, применяемого к входу:

$$\mathcal{S}(\mathcal{T}(\mathbf{I})) = \tilde{\mathcal{T}}(\mathcal{S}(\mathbf{I})). \quad (9.3)$$

Например, если сегментированное изображение имеет меньшее разрешение, чем исходное, если \mathcal{T} – это преобразование в пространстве исходного изображения, то $\tilde{\mathcal{T}}$ представляет собой соответствующее преобразование в пространстве сегментации более низкой размерности. Аналогично если \mathcal{S} – это оператор, измеряющий ориентацию объекта на изображении, а \mathcal{T} – это поворот (который представляет собой сложное нелинейное преобразование всех значений пикселей на изображении), то $\tilde{\mathcal{T}}$ будет увеличивать или уменьшать скалярное значение ориентации, полученное с помощью \mathcal{S} .

Кроме того, можно заметить, что инвариантность – это частный случай эквивариантности, когда выходное преобразование просто тождественно. Например, если \mathcal{C} – это нейронная сеть, классифицирующая объекты на изображении, а \mathcal{T} – это оператор преобразования, то

$$\mathcal{C}(\mathcal{T}(\mathbf{I})) = \mathcal{C}(\mathbf{I}), \quad (9.4)$$

т. е. класс объекта не зависит от его положения на изображении.

9.2. Уменьшение весов

Регуляризация была введена в контексте линейной регрессии (см. раздел 1.2.5) для контроля сложности модели в качестве альтернативы ограничению числа параметров в модели. Простейший регуляризатор включает в себя сумму квадратов параметров модели для получения регуляризованной функции ошибки вида (9.1), которая штрафует значения параметров с крупными величинами. Эффективная сложность модели определяется выбором коэффициента регуляризации λ .

Также было замечено, что этот аддитивный член регуляризации можно интерпретировать как отрицательный логарифм гауссова априорного распределения с нулевым средним значением для весового вектора w (см. раздел 2.6.2). Это обеспечивает вероятностную перспективу включения предварительно полученных знаний в процесс обучения модели. К сожалению, эти предварительные знания выражаются через параметры модели, в то время как любые реальные знания, которые могут быть связаны с решаемой проблемой, скорее всего, будут выражены в форме функции сети от входов к выходам. Связь между параметрами и сетевой функцией является чрезвычайно сложной, и поэтому только очень ограниченные виды предварительных знаний могут быть напрямую заданы как априорные значения параметров модели.

Регуляризатор суммы квадратов в (9.1) в литературе по машинному обучению называют *уменьшением весов* (*weight decay*), поскольку в алгоритмах последовательного обучения он способствует снижению значений весов до нуля (см. упражнение 9.3), если только они не подтверждаются данными. Одно из преимуществ такого регуляризатора заключается в том, что для использования в обучении методом градиентного спуска достаточно просто оценить его производные. В частности, для (9.1) градиент определяется как

$$\nabla \tilde{E}(\mathbf{w}) = \nabla E(\mathbf{w}) + \lambda \mathbf{w}. \quad (9.5)$$

Здесь можно заметить, что коэффициент $1/2$ в (9.1), который часто включают по общему правилу, исчезает при взятии производной.

Общий эффект квадратичного регуляризатора можно оценить на примере двумерного пространства параметров с нерегуляризованной функцией ошибки $E(\mathbf{w})$, которая является квадратичной функцией от \mathbf{w} , что соответствует простой линейной регрессионной модели с функцией ошибки в виде суммы квадратов (см. раздел 4.1.2), как показано на рис. 9.3. Как показано на рисунке, оси в пространстве параметров повернуты таким образом, чтобы они совпадали с собственными векторами матрицы Гессе, соответствующими осям эллиптических контуров нерегуляризованной функции ошибки. Для $\lambda = 0$ минимальная ошибка обозначена \mathbf{w}^* . Когда $\lambda > 0$, минимум регуляризованной функции ошибки $E(\mathbf{w}) + \lambda(w_1^2 + w_2^2)$ смещается к началу координат. Этот сдвиг больше в направлении w_1 , поскольку нерегуляризованная ошибка отно-

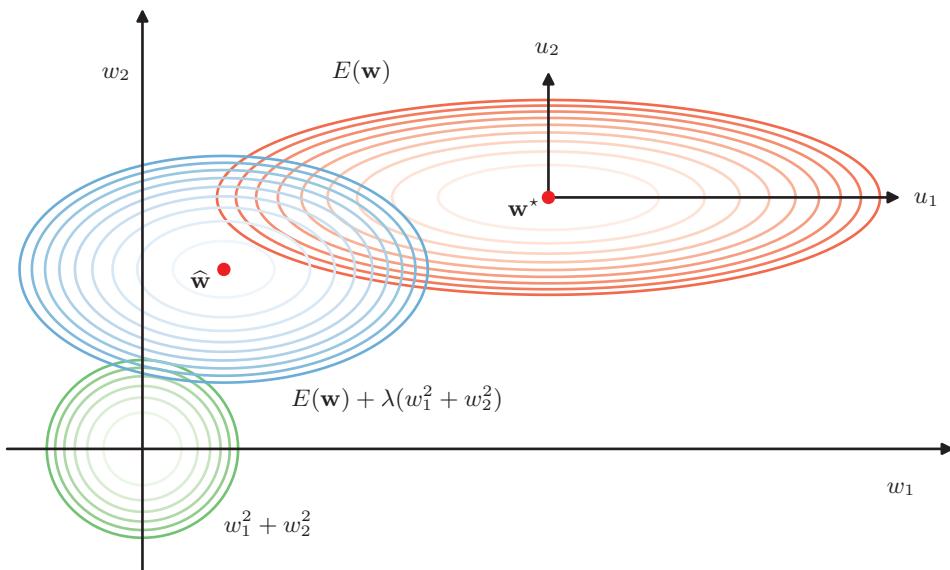


РИС. 9.3 Контуры функции ошибки (красный), члена регуляризации (зеленый) и их линейной комбинации (синий) для квадратичной функции ошибки и регуляризатора суммы квадратов $\lambda(w_1^2 + w_2^2)$

сительно нечувствительна к значению параметра, и меньше в направлении w_2 , где ошибка сильнее зависит от значения параметра. Член регуляризации эффективно подавляет параметры, которые оказывают лишь незначительное влияние на точность предсказаний сети. Фактически только параметр w_2 является «активным», поскольку выход относительно нечувствителен к w_1 , и, следовательно, регуляризатор приближает w_1 к нулю. Под *эффективным числом параметров* понимается то их количество, которое остается активным после регуляризации, и эта концепция может быть формализована как в байесовском, так и в частотном аспекте (Bishop, 2006; Hastie, Tibshirani and Friedman, 2009). При $\lambda \rightarrow \infty$ все параметры стремятся к нулю, и эффективное число параметров в этом случае равно нулю. По мере уменьшения λ число параметров увеличивается до тех пор, пока для $\lambda = 0$ оно не сравняется с общим числом фактических параметров в модели. Как видно, управление сложностью модели путем регуляризации имеет сходство с управлением сложностью модели путем ограничения числа параметров.

9.2.1. Последовательные регуляризаторы

Одним из ограничений простого уменьшения весов в форме (9.1) является то, что оно приводит к нарушению некоторых желательных свойств сетевого отображения. В качестве иллюстрации рассмотрим многослойную перцептронную сеть с двумя слоями весов и линейными выходными элементами, которая выполняет отображение набора входных переменных $\{x_i\}$ в набор выходных переменных $\{y_k\}$. Активации скрытых блоков в первом скрытом слое имеют вид:

$$z_j = h\left(\sum_i w_{ji}x_i + w_{j0}\right), \quad (9.6)$$

при этом активации выходных блоков определяются как

$$y_k = \sum_j w_{kj}z_j + w_{k0}. \quad (9.7)$$

Предположим, что необходимо выполнить линейное преобразование входных данных:

$$x_i \rightarrow \tilde{x}_i = ax_i + b. \quad (9.8)$$

В этом случае для неизменности отображения, выполняемого сетью, можно произвести соответствующее линейное преобразование весов и смещений от входов к элементам скрытого слоя (см. упражнение 9.4):

$$w_{ji} \rightarrow \tilde{w}_{ji} = \frac{1}{a}w_{ji}, \quad (9.9)$$

$$w_{j0} \rightarrow \tilde{w}_{j0} = w_{j0} - \frac{b}{a} \sum_i w_{ji}. \quad (9.10)$$

По аналогии линейное преобразование выходных переменных сети выглядит как

$$y_k \rightarrow \tilde{y}_k = cy_k + d \quad (9.11)$$

и может быть достигнуто преобразованием весов и смещений второго слоя с помощью

$$w_{kj} \rightarrow \tilde{w}_{kj} = cw_{kj}, \quad (9.12)$$

$$w_{k0} \rightarrow \tilde{w}_{k0} = cw_{k0} + d. \quad (9.13)$$

Если одну сеть обучить на исходных данных, а другую – на данных, для которых входные и/или целевые переменные были преобразованы одним из указанных выше линейных преобразований, то для согласованности необходимо получить эквивалентные сети, которые отличаются только линейным преобразованием весов, как это и было предусмотрено. Любой регуляризатор должен соответствовать этому свойству, иначе он будет произвольно отдавать предпочтение одному решению в ущерб другому, эквивалентному. Очевидно, что простое уменьшение весов в (9.1), где все веса и смещения рассматриваются на равных основаниях, не удовлетворяет этому свойству.

Поэтому необходимо подобрать регуляризатор, инвариантный к линейным преобразованиям (9.9), (9.10), (9.12) и (9.13). В соответствии с ними регуляризатор должен быть инвариантен к изменению масштаба весов и сдвигау смещений. Такой регуляризатор имеет вид:

$$\frac{\lambda_1}{2} \sum_{w \in \mathcal{W}_1} w^2 + \frac{\lambda_2}{2} \sum_{w \in \mathcal{W}_2} w^2, \quad (9.14)$$

где \mathcal{W}_1 обозначает набор весов в первом слое, \mathcal{W}_2 – набор весов во втором слое, а смещения исключены из суммирования. Этот регуляризатор останется неизменным при преобразованиях весов, если параметры регуляризации будут перескастабированы посредством $\lambda_1 \rightarrow a^{1/2}\lambda_1$ и $\lambda_2 \rightarrow c^{-1/2}\lambda_2$.

Регуляризатору (9.14) соответствует априорное распределение по параметрам вида

$$p(\mathbf{w} | \alpha_1, \alpha_2) \propto \exp\left(-\frac{\alpha_1}{2} \sum_{w \in \mathcal{W}_1} w^2 - \frac{\alpha_2}{2} \sum_{w \in \mathcal{W}_2} w^2\right). \quad (9.15)$$

Обратите внимание, что априорные параметры такой формы являются некорректными (они не могут быть нормализованы), поскольку параметры смещения не ограничены. Использование неправильных априорных параметров может привести к трудностям при выборе коэффициентов регуляризации и при сравнении моделей в байесовской концепции. Поэтому обычно для смещений (которые при этом нарушают инвариантность сдвига) используются отдельные априорные параметры со своими собственными гиперпараметрами.

Эффект от применения четырех гиперпараметров можно проиллюстрировать взятием выборок из априорных распределений и построением соответствующих сетевых функций, как показано на рис. 9.4. Здесь априорные распределения управляются четырьмя гиперпараметрами, α_1^b , α_1^w , α_2^b и α_2^w , которые представляют собой точности (precisions) гауссовых распределений смещений первого слоя, весов первого слоя, смещений второго слоя и весов второго слоя соответственно. Как видно на рисунке, параметр α_2^w регулирует вертикальный масштаб функций (обратите внимание на разные диапазоны вертикальной оси на двух верхних диаграммах), α_1^w отвечает за горизонтальный масштаб вариаций значений функций, а α_1^b – это горизонтальный диапазон, в котором происходят вариации. Параметр α_2^b , влияние которого здесь не показано, регулирует диапазон вертикальных смещений функций.

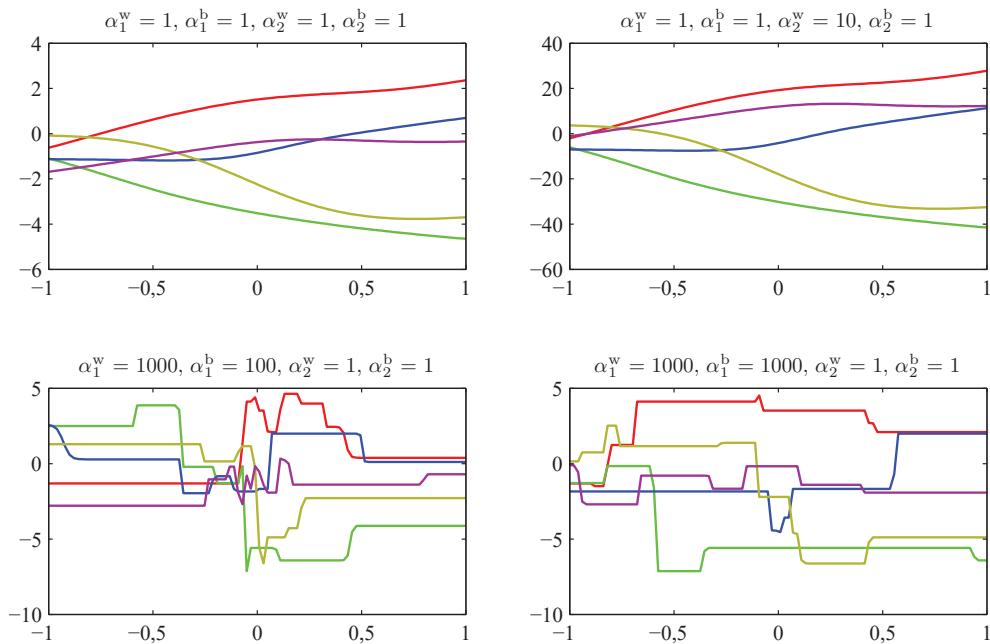


РИС. 9.4 Иллюстрация влияния гиперпараметров для определения априорного распределения весов и смещений в двухслойной сети с одним входом, одним линейным выходом и 12 скрытыми блоками с функциями активации \tanh

В более общем случае можно использовать регуляризаторы, в которых веса делятся на любое количество групп \mathcal{W}_k так, что

$$\Omega(\mathbf{w}) = \frac{1}{2} \sum_k \alpha_k \|\mathbf{w}\|_k^2, \quad (9.16)$$

где

$$\|\mathbf{w}\|_k^2 = \sum_{j \in \mathcal{W}_k} w_j^2. \quad (9.17)$$

Например, для каждого слоя многослойной сети можно использовать свой регуляризатор.

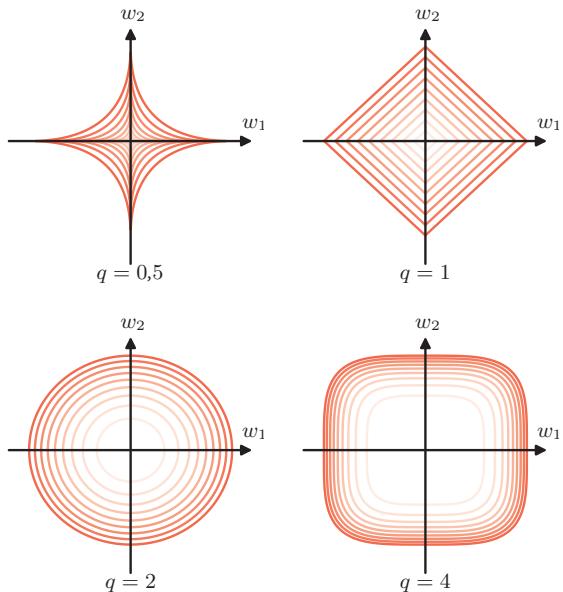
9.2.2. Обобщенное уменьшение весов

Иногда используется обобщение простого квадратичного регуляризатора:

$$\Omega(\mathbf{w}) = \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q, \quad (9.18)$$

где $q = 2$ соответствует квадратичному регуляризатору в (9.1). На рис. 9.5 показаны контуры функции регуляризации для различных значений q .

РИС. 9.5 Контуры члена регуляризации в (9.18) для различных значений параметра q



Регуляризатор вида (9.18) с $q = 1$ известен в статистической литературе как *лассо* (*lasso*) (Tibshirani, 1996). Для квадратичных функций ошибки он обладает тем свойством, что при достаточно большом λ некоторые коэффициенты w_j стремятся к нулю, что приводит к *разреженной модели* (*sparse model*), где соответствующие базисные функции не играют никакой роли. Чтобы убедиться в этом, сначала примем во внимание, что минимизация регуляризованной функции ошибки

$$E(\mathbf{w}) + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q \quad (9.19)$$

эквивалентна минимизации нерегуляризованной функции ошибки $E(\mathbf{w})$ при условии ограничения (см. упражнение 9.5)

$$\sum_{j=1}^M |w_j|^q \leq \eta \quad (9.20)$$

для соответствующего значения параметра η , где эти два подхода могут быть связаны с помощью множителей Лагранжа (см. приложение С). Источник разреженности можно увидеть на рис. 9.6, где показан минимум функции ошибки при условии ограничения (9.20). По мере увеличения λ все больше параметров будут стремиться к нулю. Для сравнения: квадратичный регуляризатор оставляет оба весовых параметра с ненулевыми значениями.

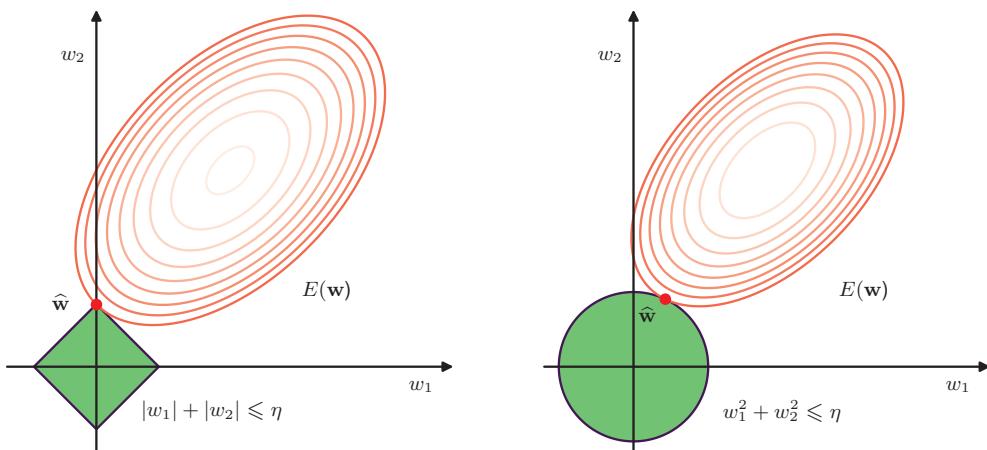


РИС. 9.6 График контуров нерегуляризованной функции ошибки (красный) вместе с областью ограничений (9.20) для регуляризатора лasso $q = 1$ слева и квадратичного регуляризатора $q = 2$ справа, в которых оптимальное значение вектора параметров w обозначается \hat{w} . Лasso дает разреженное решение, в котором $\hat{w}_1 = 0$, тогда как квадратичный регуляризатор просто уменьшает w_1 до меньшего значения

Регуляризация позволяет обучать сложные модели на наборах данных ограниченного размера без чрезмерной подгонки, по сути, ограничивая эффективную сложность модели. Однако в этом случае проблема определения оптимальной сложности модели переходит от поиска соответствующего количества обучаемых параметров к определению подходящего значения коэффициента регуляризации λ . Вопрос сложности модели будет рассмотрен в следующем разделе.

9.3. Кривые обучения

Ранее уже рассматривался вопрос о том, как меняется эффективность обобщения модели при изменении количества ее параметров, размера набора данных и коэффициента регуляризации уменьшения весов. Каждый из этих параметров дает возможность найти компромисс между смещением

и дисперсией для минимизации ошибки обобщения. Еще одним фактором, влияющим на этот баланс, является сам процесс обучения. При оптимизации функции ошибки с помощью градиентного спуска обычно происходит ее уменьшение по мере обновления параметров модели, в то время как ошибка для удержаных данных может быть неоднородной. Такое поведение можно визуализировать с помощью *кривых обучения* (*learning curves*), которые позволяют построить график показателей эффективности, таких как ошибка обучающего и проверочного наборов данных, в зависимости от количества итераций в процессе итеративного обучения, например стохастического градиентного спуска. Эти кривые дают представление о ходе обучения, а также предоставляют практическую методику контроля эффективной сложности модели.

9.3.1. Ранняя остановка

Альтернативой регуляризации в качестве способа управления эффективной сложностью сети является *ранняя остановка* (*early stopping*). Обучение моделей глубокого обучения включает в себя итерационное уменьшение функции ошибки, определенной относительно набора обучающих данных. Хотя функция ошибки, оцениваемая по обучающему набору, часто показывает в целом равномерное уменьшение в зависимости от числа итераций, ошибка, измеренная по отношению к удержаным данным, обычно называемым валидационным набором, часто показывает сначала уменьшение, а затем увеличение, поскольку сеть начинает перестраиваться. По этой причине для получения сети с хорошими обобщающими характеристиками обучение должно быть остановлено в точке наименьшей ошибки по отношению к набору валидационных данных, как показано на рис. 9.7 (вертикальные пунктирные линии).

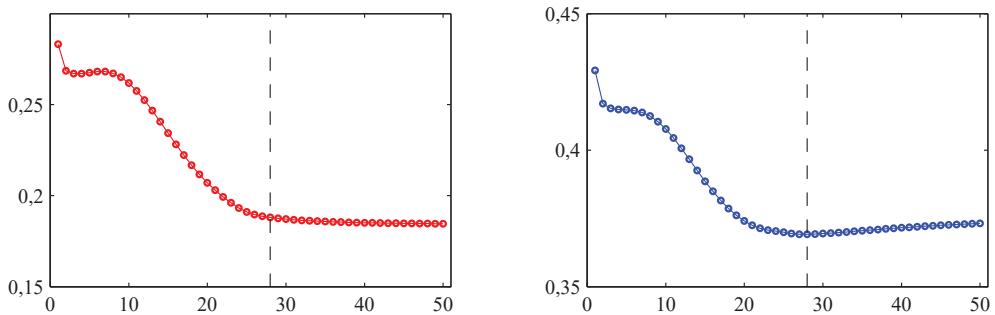


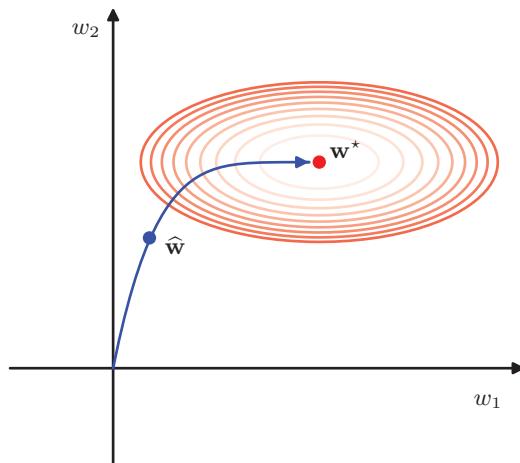
РИС. 9.7 Иллюстрация поведения ошибки обучающего набора (слева) и ошибки валидационного набора (справа) во время типичного сеанса обучения в зависимости от шага итерации для набора синусоидальных данных

Такое поведение кривых обучения иногда лучше всего объясняется с учетом эффективного числа параметров в сети. Это число начинается с малого

и затем в процессе обучения увеличивается, что соответствует постоянному росту эффективной сложности модели. Остановка обучения до достижения минимума ошибки обучения является одним из способов ограничения эффективной сложности сети.

Данное предположение можно проверить для квадратичной функции ошибки и доказать, что ранняя остановка должна демонстрировать поведение, аналогичное регуляризации с использованием простого члена уменьшения весов (Bishop, 1995a). Это видно на рис. 9.8, где оси в пространстве весов повернуты так, чтобы быть параллельными собственным векторам матрицы Гессе (см. раздел 7.1.1). Эллипсы показывают контуры постоянной ошибки, а w^* обозначает решение максимального правдоподобия, соответствующее минимуму нерегуляризованной функции ошибки. Если в отсутствие уменьшения весов вектор весов начинается в начале координат и во время обучения движется по траектории, которая следует за локальным вектором отрицательного градиента, то изначально он будет двигаться параллельно оси w_2 через точку, приблизительно соответствующую \hat{w} (показано кривой), а затем будет перемещаться к минимуму функции ошибки w_{ML} . Это следует из формы поверхности ошибки и сильно различающихся собственных значений матрицы Гессе. Таким образом, остановка в точке, близкой к \hat{w} , похожа на уменьшение весов. Если прекратить обучение раньше времени, то можно найти вектор весов \hat{w} , который качественно похож на тот, что получается при использовании простого регуляризатора с уменьшением весов вместе с обучением до минимума регуляризованной ошибки, как видно из сравнения с рис. 9.3. Связь между ранней остановкой и уменьшением весов можно выразить количественно (см. упражнение 9.6), доказав тем самым, что величина $t\eta$ (где t – это индекс итерации, а η – это параметр скорости обучения) действует как обратная величина параметра регуляризации λ . Вследствие этого эффективное число параметров в сети растет в процессе обучения.

РИС. 9.8 Схематическая иллюстрация результатов ранней остановки для квадратичной функции ошибки, схожих с уменьшением весов



9.3.2. Двойной спуск

Компромисс между смещением и дисперсией (см. раздел 4.3) дает представление об эффективности обобщения обучаемой модели при изменении количества ее параметров. Модели со слишком малым числом параметров будут характеризоваться высокой ошибкой тестового набора из-за ограниченной репрезентативной способности (высокое смещение), и при увеличении числа параметров вероятность ошибки тестового набора будет снижаться. Однако при дальнейшем увеличении числа параметров ошибка тестового набора снова возрастает из-за чрезмерной подгонки (высокая дисперсия). Это приводит к распространенному в классической статистике предположению, что число параметров в модели должно быть ограничено в зависимости от размера набора данных и что для определенного набора обучающих данных очень большие модели будут иметь низкую производительность.

Тем не менее вопреки этому предположению современные глубокие нейронные сети могут демонстрировать отличную производительность, даже если число параметров значительно превышает количество, необходимое для достижения идеального соответствия обучающим данным (Zhang et al., 2016), и общее мнение сообщества специалистов по глубокому обучению заключается в том, что большие модели эффективнее. Несмотря на то что иногда используется ранняя остановка, модели могут быть обучены до нулевой ошибки и при этом показывать хорошие результаты на тестовых данных.

Эти, казалось бы, противоречивые позиции могут быть согласованы при помощи кривых обучения и другого вида построения графиков зависимости эффективности обобщения от сложности модели. Они показывают более тонкое изменение, называемое *двойным спуском* (*double descent*) (Belkin et al., 2019). На рис. 9.9 показаны ошибки обучающего и тестового множеств в зависимости от сложности модели, определяемой количеством обучаемых параметров, для большой нейронной сети ResNet18 (He et al., 2015a) с 18 слоями параметров, которая была обучена на задаче классификации изображений. Количество весов и смещений в сети варьируется путем изменения *параметра ширины* (*width parameter*), который регулирует количество скрытых элементов в каждом слое. Горизонтальная ось представляет собой гиперпараметр, определяющий количество скрытых блоков и, следовательно, общее количество весов и смещений в сети. Вертикальная пунктирная линия, обозначенная как «порог интерполяции» (*interpolation threshold*), указывает на уровень сложности модели, при котором она в принципе способна достичь нулевой ошибки на обучающем множестве. Как и ожидалось, ошибка обучения монотонно уменьшается с ростом сложности модели. Однако ошибка тестового набора сначала уменьшается, затем снова увеличивается и, наконец, снова уменьшается. Это уменьшение ошибки тестового набора для очень больших моделей продолжается даже после того, как ошибка обучающего набора достигла нуля.

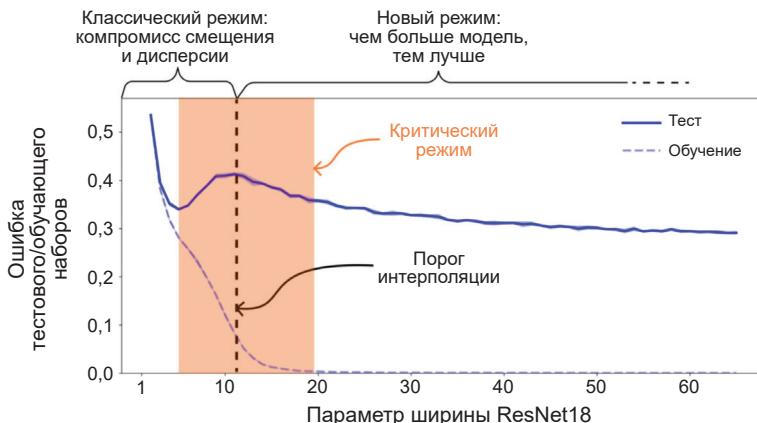


РИС. 9.9 График зависимости ошибок обучающего и тестового наборов от сложности модели для большой модели нейросети ResNet18, обученной на задаче классификации изображений. [Из (Nakkiran et al., 2019) с разрешения авторов]

Это удивительное поведение сложнее, чем можно было бы ожидать на основании обычного распределения смещения и дисперсии в классической статистике. Оно демонстрирует два различных режима подгонки моделей, как схематично показано на рис. 9.9: соответствующий классическому компромиссу смещения и дисперсии для моделей малой и средней сложности, за которым следует дальнейшее снижение ошибки тестового набора по мере перехода в режим очень больших моделей. Переход между двумя режимами происходит приблизительно в момент увеличения числа параметров в модели настолько, что модель способна точно соответствовать обучающим данным (Belkin et al., 2019). В работе (Nakkiran et al., 2019) эффективная сложность модели (effective model complexity) определяется как максимальный размер обучающего набора данных, на котором модель может достичь нулевой ошибки обучения, и поэтому двойной спуск появляется в том случае, если эффективная сложность модели превышает количество точек данных в обучающем наборе.

Аналогичное поведение можно наблюдать при управлении сложностью модели с помощью ранней остановки, как показано на рис. 9.10. Увеличение числа эпох обучения повышает эффективную сложность модели, и в случае достаточно большой модели также наблюдается двойной спуск. Для таких моделей существует множество возможных решений, включая варианты с чрезмерной подгонкой данных. В связи с этим, как представляется, стохастический градиентный спуск обладает свойством, благодаря которому вносимые им неявные смещения приводят к хорошим показателям обобщения.

Аналогичные результаты получаются и при использовании регуляризационного члена в функции ошибки для контроля сложности. Здесь ошибка

тестового набора большой модели, обучавшейся до схождения, показывает двойной спуск по отношению к $1/\lambda$, обратному параметру регуляризации, поскольку высокий показатель λ соответствует низкой сложности (Yilmaz and Heckel, 2022).

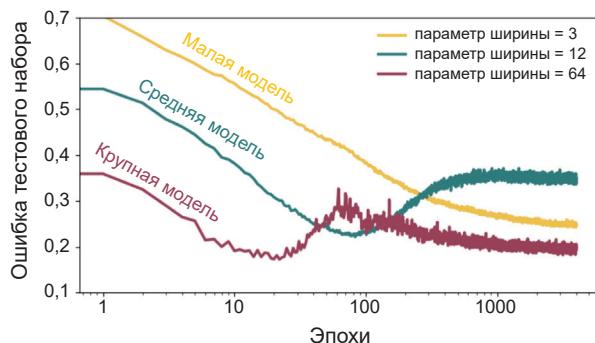


РИС. 9.10 График зависимости ошибки тестового набора от количества эпох обучения градиентным спуском для моделей ResNet18 разного размера. Эффективная сложность модели увеличивается с ростом числа эпох обучения, и для достаточно большой модели наблюдается феномен двойного спуска. [Из (Nakkiran et al., 2019) с разрешения авторов]

Одним из парадоксальных последствий двойного спуска является возможность работы в режиме, когда увеличение размера обучающего набора данных может фактически снизить производительность, что противоречит общепринятому мнению о том, что большое количество данных всегда к лучшему. Для модели в критическом режиме, показанном на рис. 9.9, увеличение размера обучающего набора может сдвинуть порог интерполяции вправо, что приведет к увеличению ошибки тестового набора. Это подтверждается на рис. 9.11, где показана ошибка тестового набора для модели преобразователя в зависимости от размерности входного пространства (см. главу 12), что называется размерностью встраивания (embedding dimension). Увеличение размерности встраивания увеличивает количество весов и смещений в модели и, следовательно, повышает сложность модели. Как видно, увеличение размера обучающего набора с 4000 до 18 000 точек данных приводит к уменьшению ошибки тестового множества, и кривая в целом становится значительно ниже, но для некоторых промежуточных значений сложности модели может наблюдаться увеличение ошибки, как показано вертикальными красными стрелками.

Для диапазона размерностей встраивания, соответствующих моделям в режиме критической сложности, увеличение размера набора данных может фактически снизить эффективность обобщения.

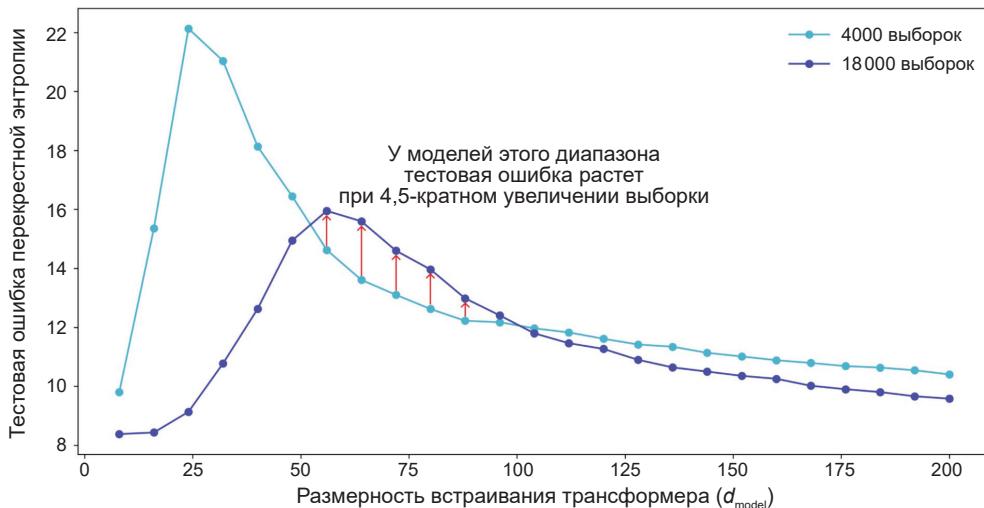


РИС. 9.11 График ошибки тестового набора для большой модели преобразователя в зависимости от размерности встраивания, которая контролирует количество параметров в модели. [Из (Nakkiran et al., 2019) с разрешения авторов]

9.4. Совместное использование параметров

Регуляризационные члены, такие как L_2 -регуляризатор $\|w\|^2$, способствуют уменьшению чрезмерной подгонки, стимулируя приближение значений весов к нулю. Еще одним способом уменьшения сложности сети является наложение жестких ограничений на значения весов путем их объединения в группы и установления требования об одинаковом значении всех весов в каждой группе, при этом общее значение определяется из данных. Этот способ известен как *совместное использование весов* (*weight sharing*), *совместное использование параметров* (*parameter sharing*) или *связывание параметров* (*parameter tieing*). При этом число степеней свободы становится меньше числа связей в сети. Обычно такой способ вводится для кодирования индуктивного смещения в сети с целью выражения каких-либо известных инвариантностей. Оценка градиентов функции ошибки для таких сетей может быть выполнена с помощью небольшой модификации обратного распространения (см. упражнение 9.7), хотя на практике это осуществляется в неявном виде с помощью автоматического дифференцирования. Совместное использование параметров будет активно применяться при изучении сверточных нейронных сетей (см. главу 10). Однако совместное использование параметров применимо только к конкретным задачам, где форма ограничений может быть определена заранее.

9.4.1. Мягкое разделение весов

В качестве альтернативы использования жестких ограничений, которые приводят к равенству наборов параметров модели, в работе (Nowlan and Hinton, 1992) была представлена форма мягкого разделения весов (soft weight sharing), при котором член регуляризации стимулирует группы весов к схожим значениям. Более того, разделение весов на группы, среднее значение веса для каждой группы и разброс значений внутри групп определяются в процессе обучения.

Напомним, что регуляризатор с простым уменьшением веса в (9.1) можно рассматривать как отрицательный логарифм гауссова априорного распределения для весов. Это способствует тому, что все веса сходятся к единственному значению, равному нулю. Вместо этого значения весов могут образовывать несколько групп, а не только одну, если рассматривать распределение вероятностей в виде смеси гауссовых распределений (см. раздел 3.2.9). Средние значения $\{\mu_j\}$ и дисперсии $\{\sigma_j^2\}$ гауссовых компонент, а также коэффициенты смещивания $\{\pi_j\}$ можно считать настраиваемыми параметрами, которые будут определяться в процессе обучения.

Таким образом, получается плотность распределения вероятности вида

$$p(\mathbf{w}) = \prod_i \left\{ \sum_{j=1}^K \pi_j \mathcal{N}(w_i | \mu_j, \sigma_j^2) \right\}, \quad (9.21)$$

где K – это число компонентов в смеси. Взятие отрицательного логарифма приводит к функции регуляризации вида

$$\Omega(\mathbf{w}) = -\sum_i \ln \left(\sum_{j=1}^K \pi_j \mathcal{N}(w_i | \mu_j, \sigma_j^2) \right). \quad (9.22)$$

Тогда функция общей ошибки задается как

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \Omega(\mathbf{w}), \quad (9.23)$$

где λ – это коэффициент регуляризации.

Эта ошибка минимизируется совместно по отношению к весам $\{w_i\}$ и по отношению к параметрам $\{\pi_j, \mu_j, \sigma_j\}$ модели смеси. Это можно сделать с помощью градиентного спуска, который требует оценки производных $\Omega(\mathbf{w})$ по всем обучаемым параметрам. Для этого удобно рассматривать $\{\pi_j\}$ в качестве априорных вероятностей того, что каждый компонент сгенерировал значение веса, и ввести соответствующие апостериорные вероятности (см. упражнение 9.8), которые определяются по теореме Байеса:

$$\gamma_i(w_i) = \frac{\pi_j \mathcal{N}(w_i | \mu_j, \sigma_j^2)}{\sum_k \pi_k \mathcal{N}(w_i | \mu_k, \sigma_k^2)}. \quad (9.24)$$

Тогда производные функции суммарной ошибки по весам определяются (см. упражнение 9.9) в виде

$$\frac{\partial \tilde{E}}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda \sum_j \gamma_i(w_i) \frac{(w_i - \mu_j)}{\sigma_j^2}. \quad (9.25)$$

Таким образом, эффект регуляризации заключается в вытягивании каждого веса к центру j -го гауссова распределения с силой, пропорциональной апостериорной вероятности этого гауссова распределения для данного веса. Это именно то воздействие, которое нужно для достижения поставленной цели.

Производные ошибки по центрам гауссовых распределений также легко вычисляются (см. упражнение 9.10), и в результате получаем:

$$\frac{\partial \tilde{E}}{\partial \mu_j} = \lambda \sum_i \gamma_i(w_i) \frac{(\mu_j - w_i)}{\sigma_i^2}, \quad (9.26)$$

что имеет простую понятную интерпретацию, поскольку оно продвигает μ_j к среднему значению весов, взвешенному по апостериорным вероятностям того, что соответствующие весовые параметры были сгенерированы компонентом j .

Чтобы гарантировать, что дисперсии $\{\sigma_j^2\}$ остаются положительными, необходимо ввести новые переменные $\{\xi_j\}$, определяемые как

$$\sigma_j^2 = \exp(\xi_j), \quad (9.27)$$

и выполнить минимизацию без ограничений относительно $\{\xi_j\}$. Тогда соответствующие производные (упражнение 9.11) имеют вид:

$$\frac{\partial \tilde{E}}{\partial \xi} = \frac{\lambda}{2} \sum_j \gamma_i(w_i) \left(1 - \frac{(w_i - \mu_j)^2}{\sigma_j^2} \right). \quad (9.28)$$

Этот процесс приближает σ_j к взвешенному среднему значению квадратичных отклонений весов относительно соответствующего центра μ_j , где весовые коэффициенты вновь задаются апостериорной вероятностью того, что каждый вес генерируется компонентом j .

Для производных по коэффициентам смешивания π_j необходимо учитывать ограничения

$$\sum_j \pi_j = 1, \quad 0 \leq \pi_i \leq 1, \quad (9.29)$$

которые вытекают из интерпретации π_j в качестве априорных вероятностей. Для этого необходимо выразить коэффициенты смешивания в виде набора вспомогательных переменных $\{\eta_j\}$ с помощью функции *softmax*, которая имеет вид:

$$\pi_j = \frac{\exp(\eta_j)}{\sum_{k=1}^K \exp(\eta_k)}. \quad (9.30)$$

Тогда производные регуляризованной функции ошибки по $\{\eta_j\}$ (см. упражнение 9.12) имеют вид:

$$\frac{\partial \tilde{E}}{\partial \eta_j} = \lambda \sum_i \{\pi_j - \gamma_j(w_i)\}. \quad (9.31)$$

Отсюда видно, что π_j стремится к средней апостериорной вероятности для компонента смеси j .

Другой вариант применения мягкого разделения весов (Lasserre, Bishop and Minka, 2006) представляет собой принципиально новый подход, сочетающий обучение генеративной модели без наблюдения с обучением соответствующей дискриминантной модели под наблюдением. Это может пригодиться в ситуациях с большим количеством немаркированных данных и нехваткой маркированных данных. Преимущество генеративной модели заключается в том, что для определения ее параметров можно использовать все данные, в то время как параметры дискриминативной модели определяются только на основе маркированных примеров. Однако дискриминативная модель может достичь лучшего обобщения при неправильной спецификации модели, т. е. когда модель не совсем точно описывает истинное распределение, порождающее данные, что обычно и происходит. Благодаря мягкому связыванию параметров двух моделей можно получить точно заданный гибрид генеративного и дискриминативного подходов, который может быть устойчив к неправильной спецификации модели и при этом получает преимущества от обучения на немаркированных данных.

9.5. Остаточные связи

Репрезентативные возможности глубоких нейронных сетей во многом обусловлены использованием нескольких слоев обработки, и ранее было установлено, что увеличение числа слоев в сети может значительно повысить эффективность обобщения.

Ранее также было показано, как пакетная нормализация (см. раздел 7.4.2) наряду с тщательной инициализацией весов и смещений (см. раздел 7.2.5) позволяют решить проблему исчезающих или взрывающихся градиентов в глубоких сетях. Однако даже при пакетной нормализации обучать сети с большим количеством слоев становится все труднее.

Одно из объяснений этого явления получило название «рассыпающиеся градиенты» (*shattered gradients*) (Balduzzi et al., 2017). Ранее уже было отмечено, что репрезентативные возможности нейронных сетей растут экспоненциально с увеличением глубины. При использовании функций активации ReLU наблюдается экспоненциальное увеличение количества линейных областей, которые сеть может представлять (см. раздел 6.3).

Однако следствием этого является увеличение числа разрывов в градиенте функции ошибки. Это показано на рис. 9.12 для сетей с одной входной

и одной выходной переменной. Здесь производная выходной переменной по отношению к входной переменной (якобиан сети) построена как функция входной переменной. Согласно цепному правилу вычислений эти производные определяют градиенты поверхности функции ошибки. Как видно, для глубоких сетей крайне малые изменения весовых параметров в ранних слоях сети могут привести к значительным изменениям градиента. Итерационные алгоритмы оптимизации на основе градиента основаны на предположении, что градиент плавно изменяется в пространстве параметров, и поэтому эффект «рассыпающегося градиента» может привести к неэффективному обучению в очень глубоких сетях.

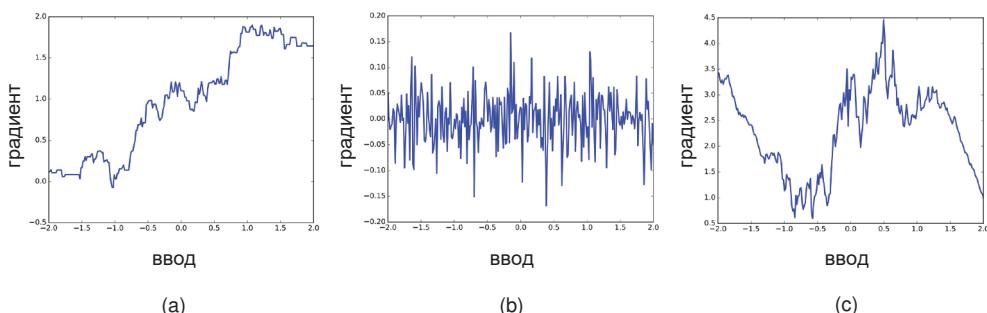


РИС. 9.12 Графики якобиана для сетей с одним входом и одним выходом, показывающие (a) сеть с двумя слоями весов, (b) сеть с 25 слоями весов и (c) сеть с 51 слоем весов вместе с остаточными связями. [Из (Balduzzi et al., 2017) с разрешения авторов]

Важной модификацией архитектуры нейронных сетей, которая в значительной степени способствует обучению очень глубоких сетей, является использование *остаточных связей* (*residual connections*) (He et al., 2015a), которые представляют собой особую форму *связей пропущенного слоя* (*skip-layer connections*). Рассмотрим нейронную сеть из последовательности трех слоев вида

$$\mathbf{z}_1 = \mathbf{F}_1(\mathbf{x}), \quad (9.32)$$

$$\mathbf{z}_2 = \mathbf{F}_2(\mathbf{z}_1), \quad (9.33)$$

$$\mathbf{y} = \mathbf{F}_3(\mathbf{z}_2). \quad (9.34)$$

Здесь функции $\mathbf{F}_i(\cdot)$ могут состоять из простого линейного преобразования, за которым следует функция активации ReLU, или они могут быть более сложными с несколькими линейными слоями, слоями функции активации и нормализации. Остаточная связь сводится к тому, что вход каждой функции добавляется обратно к выходу, чтобы получить

$$\mathbf{z}_1 = \mathbf{F}_1(\mathbf{x}) + \mathbf{x}, \quad (9.35)$$

$$\mathbf{z}_2 = \mathbf{F}_2(\mathbf{z}_1) + \mathbf{z}_1, \quad (9.36)$$

$$\mathbf{y} = \mathbf{F}_3(\mathbf{z}_2) + \mathbf{z}_2. \quad (9.37)$$

Каждая комбинация функции и остаточной связи, например $\mathbf{F}_1(\mathbf{x}) + \mathbf{x}$, называется *остаточным блоком* (*residual block*). Сеть с остаточными связями (*residual network*), которую также называют ResNet, состоит из нескольких последовательных слоев подобных блоков. Модифицированная сеть с остаточными связями показана на рис. 9.13. Остаточный блок может достаточно просто генерировать преобразование тождества, если параметры нелинейной функции достаточно малы, чтобы выходы функции приближались к нулю.

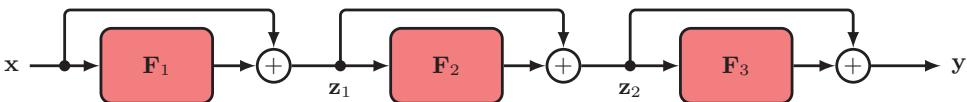


РИС. 9.13 Сеть с остаточными связями, включающая три остаточных блока, которые соответствуют последовательности преобразований (9.35)–(9.37)

Термин «остаточный» означает, что в каждом блоке функция определяет остаток между картой тождества и желаемым выходом, что подтверждается перестановкой преобразования остатков:

$$\mathbf{F}_l(\mathbf{z}_{l-1}) = \mathbf{z}_l - \mathbf{z}_{l-1}. \quad (9.38)$$

Градиенты в сети с остаточными связями гораздо менее чувствительны к входным значениям по сравнению со стандартной глубокой сетью, как показано на рис. 9.12(с).

В работе (Li et al., 2017) был предложен способ прямой визуализации поверхностей ошибок, который показал, что эффект остаточных связей заключается в создании более гладких поверхностей функции ошибки, как показано на рис. 9.14. Обычно в остаточную сеть включают слои пакетной нормализации, поскольку вместе они значительно уменьшают проблему исчезающих и взрывающихся градиентов. В работе (He et al., 2015а) было показано, что включение остаточных связей позволяет эффективно обучать очень глубокие сети, которые потенциально могут иметь сотни слоев.

Дополнительное представление о том, как остаточные связи способствуют сглаживанию поверхностей ошибок, можно получить при объединении (9.35), (9.36) и (9.37) для получения одного общего уравнения для всей сети:

$$\mathbf{y} = \mathbf{F}_3(\mathbf{F}_2(\mathbf{F}_1(\mathbf{x}) + \mathbf{x}) + \mathbf{z}_1) + \mathbf{z}_2. \quad (9.39)$$

Теперь можно подставить промежуточные переменные \mathbf{z}_1 и \mathbf{z}_2 и получить выражение для выхода сети в зависимости от входа \mathbf{x} (см. упражнение 9.13):

$$\begin{aligned} \mathbf{y} = & \mathbf{F}_3(\mathbf{F}_2(\mathbf{F}_1(\mathbf{x}) + \mathbf{x}) + \mathbf{F}_1(\mathbf{x}) + \mathbf{x}) \\ & + \mathbf{F}_2(\mathbf{F}_1(\mathbf{x}) + \mathbf{x}) \\ & + \mathbf{F}_1(\mathbf{x}) + \mathbf{x}. \end{aligned} \quad (9.40)$$

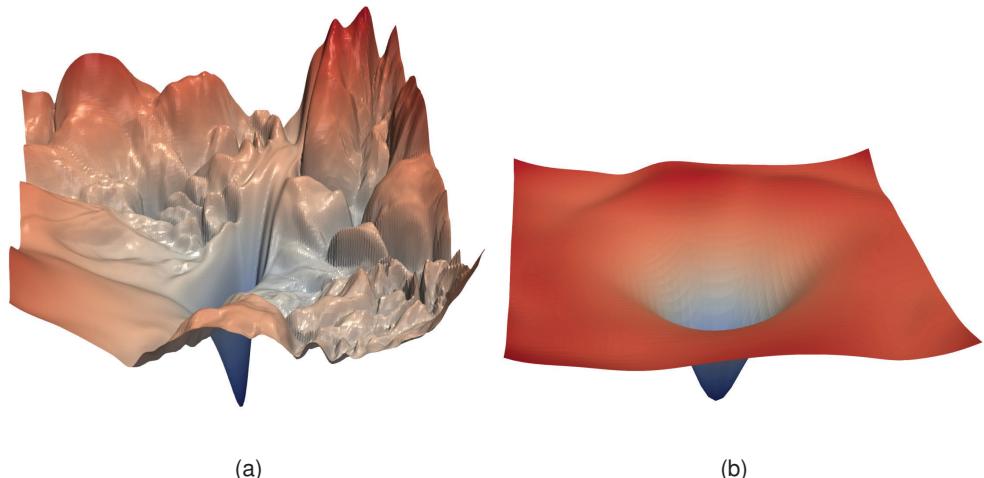


РИС. 9.14 (a) Визуализация поверхности ошибки для сети с 56 слоями. (b) Та же сеть с включением остаточных связей показывает эффект сглаживания, возникающий благодаря остаточным связям. [Из (Li et al., 2017) с разрешения авторов]

Эта расширенная форма остаточной сети изображена на рис. 9.15. Здесь видно, что общая функция состоит из нескольких параллельно действующих сетей, которые включают в себя сеть с меньшим количеством слоев. Сеть обладает репрезентативной способностью глубокой сети, поскольку включает в себя такую сеть в качестве частного случая. Однако поверхность ошибок сглаживается благодаря сочетанию неглубоких и глубоких вспомогательных сетей.

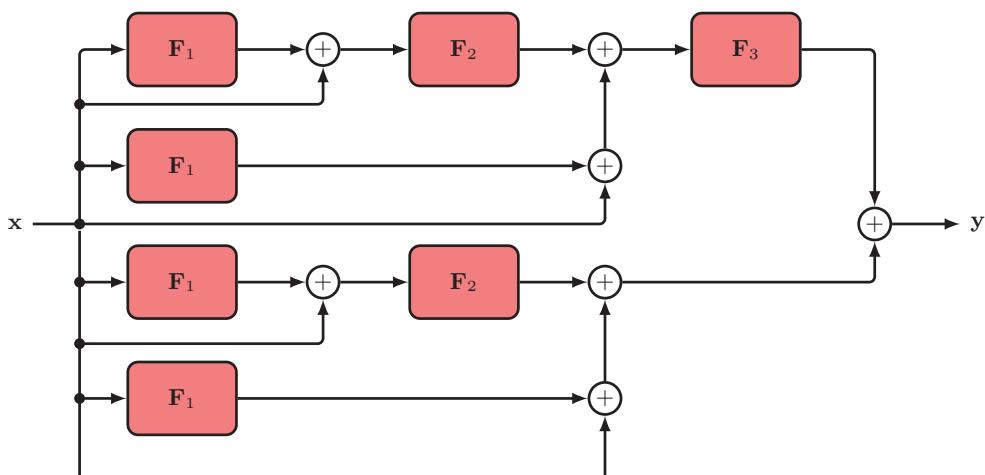


РИС. 9.15 Расширенный вид сети, показанной на рис. 9.13

Обратите внимание, что связи с пропуском слоя, определяемые (9.40), подразумевают одинаковые размерности входных и всех промежуточных переменных для возможности их добавления. В какой-то момент можно изменить размерность сети, включив в нее неквадратную матрицу \mathbf{W} обучаемых параметров в виде

$$\mathbf{z}_l = \mathbf{F}_l(\mathbf{z}_{l-1}) + \mathbf{W}\mathbf{z}_{l-1}. \quad (9.41)$$

До сих пор форма обучаемых нелинейных функций $\mathbf{F}_l(\cdot)$ не была конкретизирована. Самым простым вариантом была бы стандартная нейронная сеть, в которой чередуются слои с обучаемым линейным преобразованием и фиксированной нелинейной функцией активации, такой как ReLU. Это открывает две возможности для размещения остаточных связей, как показано на рис. 9.16. В варианте (a) добавляемые величины всегда неотрицательны, поскольку они задаются выходами слоев ReLU, и поэтому для того, чтобы можно было использовать как положительные, так и отрицательные значения, чаще всего используется вариант (b).

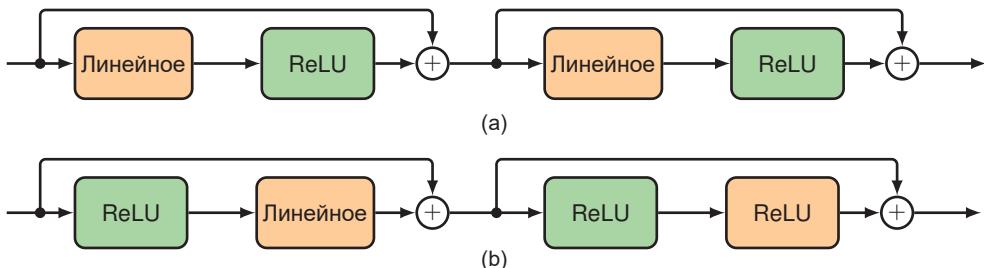


РИС. 9.16 Два альтернативных способа включения остаточных связей в стандартную сеть с прямой связью с чередованием обучаемых линейных слоев и нелинейных функций активации ReLU

9.6. Усреднение модели

При наличии нескольких различных моделей, обученных для решения одной и той же задачи, вместо выбора одной лучшей модели можно улучшить обобщение путем усреднения прогнозов, сделанных отдельными моделями. Такие комбинации моделей иногда называют *комитетами* (*committees*) или *ансамблями* (*ensembles*). Для моделей, выдающих вероятностные результаты, предсказанное распределение является средним значением предсказаний каждой модели:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{L} \sum_{l=1}^L p_l(\mathbf{y}|\mathbf{x}), \quad (9.42)$$

где $p_l(\mathbf{y}|\mathbf{x})$ – это выход модели l , а L – это общее число моделей.

Этот процесс усреднения можно описать на примере компромисса между смещением и дисперсией (см. раздел 4.3). Вспомните из рис. 4.7, что при обучении нескольких полиномов на синусоидальных данных и последующем усреднении полученных функций вклад, обусловленный дисперсионным членом, как правило, аннулируется, что приводит к более эффективному составлению прогнозов.

На практике, разумеется, используется только один набор данных, поэтому необходимо найти способ привнести вариативность между различными моделями в пределах комитета. Одним из способов может быть использование так называемых загрузочных (*bootstrap*) наборов данных, когда несколько наборов данных создаются следующим образом. Пусть исходный набор данных состоит из N точек данных $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. Новый набор данных \mathbf{X}_B создается путем случайной выборки N точек из \mathbf{X} с заменой так, что некоторые точки из \mathbf{X} могут повторяться в \mathbf{X}_B , в то время как другие точки из \mathbf{X} могут отсутствовать в \mathbf{X}_B . Этот процесс можно повторить L раз для создания L наборов данных размером N каждый, и все они созданы путем выборки из исходного набора данных \mathbf{X} . Затем каждый набор данных можно использовать для обучения модели, а прогнозы полученных моделей усредняются. Эта процедура известна как *улучшение агрегации* (*bootstrap aggregation*), или *создание ансамбля* (*Bagging*, в контексте *создание мультимножества*. – Прим. перев.) (Breiman, 1996). Альтернативным способом формирования ансамбля может быть использование исходного набора данных для обучения нескольких различных моделей с разными архитектурами.

Проанализировать преимущества ансамблевого прогнозирования можно на примере задачи регрессии с входным вектором \mathbf{x} и одной выходной переменной y . Пусть имеется набор обученных моделей $y_1(\mathbf{x}), \dots, y_M(\mathbf{x})$, и для формирования ансамблевого прогнозирования используется формула:

$$y_{\text{COM}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}). \quad (9.43)$$

Если точная функция, значение которой необходимо предсказать, задается как $h(\mathbf{x})$, то выход каждой из моделей может быть записан как истинное значение плюс ошибка:

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x}). \quad (9.44)$$

Тогда средняя ошибка суммы квадратов принимает вид:

$$\mathbb{E}_{\mathbf{x}}[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2] = \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2], \quad (9.45)$$

где $\mathbb{E}_{\mathbf{x}}[\cdot]$ обозначает частотное ожидание относительно распределения входного вектора \mathbf{x} . Таким образом, средняя ошибка для моделей, функционирующих по отдельности, равна:

$$E_{\text{AV}} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2]. \quad (9.46)$$

Точно так же ожидаемая ошибка по комитету (9.43) определяется как

$$\begin{aligned} E_{\text{COM}} &= \mathbb{E}_{\mathbf{x}} \left[\left\{ \frac{1}{M} \sum_{m=1}^M y_m(\mathbf{x}) - h(\mathbf{x}) \right\}^2 \right] \\ &= \mathbb{E}_{\mathbf{x}} \left[\left\{ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right\}^2 \right]. \end{aligned} \quad (9.47)$$

Если предположить, что ошибки имеют нулевое среднее значение и не являются коррелированными, то

$$\mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})] = 0, \quad (9.48)$$

$$\mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})] = 0, \quad m \neq l, \quad (9.49)$$

и тогда (см. упражнение 9.14) получается:

$$E_{\text{COM}} = \frac{1}{M} E_{\text{AV}}. \quad (9.50)$$

Этот, казалось бы, впечатляющий результат говорит о том, что средняя ошибка модели может быть уменьшена в M раз посредством простого усреднения M версий модели. К сожалению, данный результат зависит от ключевого предположения, что ошибки отдельных моделей не коррелированы. На практике, как правило, ошибки сильно коррелированы, и сокращение общей ошибки обычно оказывается значительно меньшим. Однако всегда можно доказать, что ожидаемая ошибка комитета не будет превышать ожидаемую ошибку входящих в него моделей (см. упражнение 9.15), так что $E_{\text{COM}} \leq E_{\text{AV}}$.

Несколько иной подход к сочетанию моделей, известный как *усиление* («подкачка», или просто *бустинг*, *boosting*) (Freund and Schapire, 1996), предполагает объединение нескольких «базовых» классификаторов для создания комитета, производительность которого может быть значительно выше, чем у любого из них. Метод бустинга может дать хорошие результаты даже в том случае, если базовые классификаторы работают лишь немного лучше, чем произвольные. Принципиальное отличие бустинга от методов комитетов, таких как рассмотренный выше метод создания мульти множеств (bagging), заключается в том, что базовые классификаторы обучаются последовательно, и каждый базовый классификатор обучается с использованием взвешенной формы набора данных, где весовой коэффициент, связанный с каждой точкой данных, зависит от производительности предыдущих классификаторов. В частности, точки, которые неправильно классифицированы одним из базовых классификаторов, имеют больший вес при обучении следующего классификатора в последовательности. После обучения всех классификаторов их прогнозы объединяются по схеме взвешенного мажоритарного голосования.

На практике основным недостатком всех методов объединения моделей является необходимость обучения нескольких моделей и последующей оценки прогнозов для всех моделей, что увеличивает вычислительные издержки

как на обучение, так и на вывод. Насколько эти затраты значимы, зависит от конкретного сценария применения.

9.6.1. Прореживание

Широко используемая и очень эффективная форма регуляризации, известная как *прореживание*, или *отсев* (*dropout*) (Srivastava et al., 2014), может быть рассмотрена в качестве неявного способа приближенного усреднения модели по экспоненциально большому числу моделей без необходимости обучения нескольких моделей по отдельности. Этот способ имеет широкое применение и является недорогим с точки зрения вычислительных затрат. Прореживание – это одна из наиболее эффективных форм регуляризации, и она активно внедряется во многих практических приложениях.

Основная идея прореживания заключается в удалении узлов из сети, включая их связи, в произвольном порядке в процессе обучения. Каждый раз, когда в сеть поступает новая точка данных, происходит новый случайный выбор узлов, которые следует исключить. На рис. 9.17 показана простая сеть, а также примеры прореженных сетей, в которых подмножества узлов были исключены.

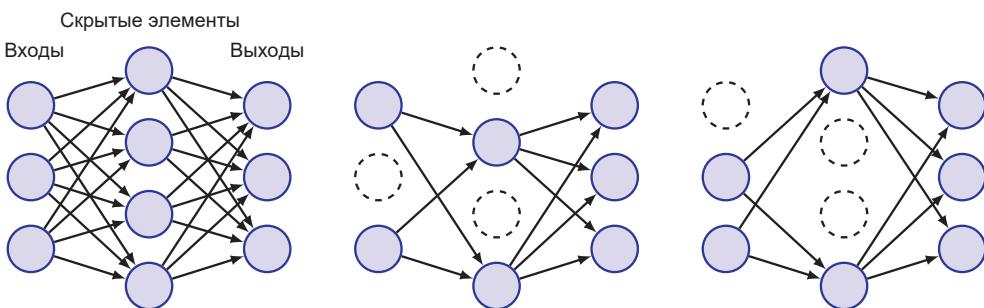


РИС. 9.17 Нейронная сеть (слева) и два примера прореженных сетей с пропуском произвольного подмножества узлов

Прореживание может применяться как к скрытым узлам, так и к входным узлам, но не к выходам, что эквивалентно установке выхода прореженного узла в ноль. Это можно реализовать посредством определения векторной маски $R_i \in \{0, 1\}$, которая перемножает активацию невыходного узла i для точки данных n , значения которой устанавливаются в 1 с вероятностью ρ . Для скрытых узлов обычно используется значение $\rho = 0,5$, тогда как для входов обычно используется значение $\rho = 0,8$.

В процессе обучения, когда каждая точка данных передается в сеть, происходит создание новой маски, и этапы прямого и обратного распространения применяются к этой прореженной сети для создания градиентов функции ошибки, которые затем используются для обновления весов, например, с помощью стохастического градиентного спуска. Если точки данных сгруппиро-

ваны в мини-батчи (см. раздел 7.2.4), то градиенты усредняются по точкам данных в каждом мини-батче перед применением обновления весов. Для сети с M невыходными узлами существует $2M$ прореженных сетей, и поэтому лишь небольшая часть этих сетей будет когда-либо рассматриваться в процессе обучения. В этом отличие от обычных ансамблевых методов, где каждая из сетей в ансамбле обучается независимо до сходимости. Еще одно отличие заключается в том, что экспоненциально большое количество сетей, которые неявно обучаются с прореживанием, не являются независимыми, но при этом совместно используют значения своих параметров с полной сетью, а значит, и друг с другом. Обратите внимание, что обучение с прореживанием может занять много времени, поскольку обновления отдельных параметров становятся очень зашумленными. Кроме того, поскольку функция ошибки по своей природе является шумной, становится сложнее убедиться в корректности работы алгоритма оптимизации лишь на основании наблюдения за уменьшением функции ошибки в процессе обучения.

После завершения обучения можно, в принципе, получить прогнозы с помощью правила ансамбля (9.42), которое в данном случае имеет вид:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{R}} p(\mathbf{R})p(\mathbf{y}|\mathbf{x}, \mathbf{R}), \quad (9.51)$$

где сумма определяется по экспоненциальному большому пространству масок, а $p(\mathbf{y}|\mathbf{x}, \mathbf{R})$ – это прогнозируемое распределение по сети с маской \mathbf{R} . Поскольку такое суммирование трудновыполнимо, его можно аппроксимировать путем выборки небольшого числа масок, и в практическом плане для достижения хороших результатов может быть достаточно всего 10 или 20 масок. Эта процедура известна как *отсев Монте-Карло* (*Monte Carlo dropout*).

Еще более простой подход заключается в составлении прогнозов с использованием обученной сети без маскированных узлов и изменении весов в сети так, чтобы ожидаемый входной сигнал для каждого узла был примерно таким же во время тестирования, как и во время обучения, с учетом того что во время обучения часть узлов будет отсутствовать. Таким образом, если во время обучения узел присутствует с вероятностью ρ , то во время тестирования выходные веса этого узла будут умножены на ρ до использования сети для составления прогнозов.

Еще одна причина для использования прореживания вытекает из байесовской концепции. При полностью байесовском подходе (см. раздел 2.6) прогнозирование осуществлялось бы путем усреднения по всем возможным $2M$ сетевым моделям, при этом каждая сеть взвешивалась бы по своей апостериорной вероятности. С вычислительной точки зрения это было бы непомерно затратно как во время обучения при оценке апостериорных вероятностей, так и во время тестирования при вычислении взвешенных прогнозов. Прореживание позволяет аппроксимировать усреднение моделей, придавая равный вес каждой из них.

Дальнейшие соображения, связанные с прореживанием, обусловлены его ролью в уменьшении избыточной подгонки. В стандартной сети параметры

могут подстраиваться под шум в отдельных точках данных, а скрытые узлы становятся слишком специализированными. Каждый узел настраивает свои веса для минимизации ошибки с учетом выходов других узлов. Это приводит к совместной адаптации узлов, которая не может быть обобщена на новые данные. При прореживании каждый узел не может полагаться на присутствие других специфических узлов и вместо этого ему приходится вносить полезный вклад в широкий спектр различных контекстных факторов, что снижает степень совместной адаптации и специализации. Для простой линейной регрессионной модели, обученной по методу наименьших квадратов, регуляризация прореживания (см. упражнение 9.18) эквивалентна модифицированной форме квадратичной регуляризации.

Упражнения

- 9.1** (*) Рассматривая по очереди каждую из четырех групповых аксиом, докажите, что множество всех возможных поворотов квадрата на (положительные или отрицательные) кратные 90° , а также бинарная операция составления поворотов образуют группу (см. раздел 9.1.3). Аналогичным образом докажите, что множество всех непрерывных перемещений объекта в двумерной плоскости также образует группу.
- 9.2** (**) Рассмотрим линейную модель вида

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^D w_i x_i \quad (9.52)$$

вместе с функцией ошибки в виде суммы квадратов, имеющей вид

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2. \quad (9.53)$$

Теперь предположим, что гауссов шум ϵ_i с нулевым средним значением и дисперсией σ^2 добавляется независимо к каждой из входных переменных x_i . Используя $\mathbb{E}[\epsilon_i] = 0$ и $\mathbb{E}[\epsilon_i \epsilon_j] = \delta_{ij} \sigma^2$, докажите, что минимизация E_D , усредненного по распределению шума, эквивалентна минимизации суммы квадратов ошибки для свободных от шума входных переменных при добавлении члена регуляризации с уменьшением весов, при этом параметр смещения w_0 в регуляризаторе опущен.

- 9.3** (**) Рассмотрим функцию ошибки, которая состоит только из квадратичного регуляризатора:

$$\Omega(\mathbf{w}) = -\frac{1}{2} \mathbf{w}^T \mathbf{w} \quad (9.54)$$

вместе с формулой обновления градиентного спуска:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau+1)} - \eta \nabla \Omega(\mathbf{w}). \quad (9.55)$$

Рассматривая предел бесконечно малых обновлений, запишите соответствующее дифференциальное уравнение для процесса изменения \mathbf{w} . Запишите решение этого уравнения, начиная с начального значения \mathbf{w}_0 , и докажите, что элементы \mathbf{w} экспоненциально убывают до нуля.

- 9.4** (*) Докажите, что функция сети, определяемая (9.6) и (9.7), инвариантна при преобразовании (9.8), применяемом к входам, при условии, что веса и смещения одновременно преобразуются с помощью (9.9) и (9.10). Аналогично докажите, что выходы сети могут быть преобразованы в соответствии с (9.11) путем применения преобразований (9.12) и (9.13) к весам и смещениям второго слоя.
- 9.5** (**) Используя множители Лагранжа (см. приложение С), докажите, что минимизация регуляризованной функции ошибки, заданной в (9.19), эквивалентна минимизации нерегуляризованной функции ошибки $E(\mathbf{w})$ при условии ограничения (9.20). Объясните взаимосвязь между параметрами η и λ .
- 9.6** (***) Рассмотрим квадратичную функцию ошибки вида

$$E = E_0 + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H}(\mathbf{w} - \mathbf{w}^*), \quad (9.56)$$

где \mathbf{w}^* представляет собой минимум, а матрица Гессе \mathbf{H} является положительно определенной и постоянной. Предположим, что начальный весовой вектор $\mathbf{w}^{(0)}$ находится в начале координат и обновляется с помощью простого градиентного спуска:

$$\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \rho \nabla E, \quad (9.57)$$

где τ обозначает номер шага, а ρ – это скорость обучения (которая предполагается небольшой). Докажите, что после τ шагов компоненты весового вектора, параллельные собственным векторам \mathbf{H} , могут быть записаны в виде

$$w_j^{(\tau)} = \{1 - (1 - \rho \eta_j)^{\tau}\} w_j^*, \quad (9.58)$$

где $w_j = \mathbf{w}^T \mathbf{u}_j$, а \mathbf{u}_j и η_j – это собственные векторы и собственные значения \mathbf{H} , соответственно, определяемые

$$\mathbf{H}_{uj} = \eta_j \mathbf{u}_j. \quad (9.59)$$

Докажите, что при $\tau \rightarrow \infty$ это дает $\mathbf{w}^{(\tau)} \rightarrow \mathbf{w}^*$, как и предполагалось, при условии $|1 - \rho \eta_j| < 1$. Теперь предположим, что обучение прекращается после конечного числа τ шагов. Докажите, что компоненты весового вектора, параллельные собственным векторам матрицы Гессе, удовлетворяют следующим условиям:

$$w_j^{(t)} \simeq w_j^* \text{ при } \eta_j \gg (\rho\tau)^{-1}, \quad (9.60)$$

$$|w_j^{(t)}| \ll |w_j^*| \text{ при } \eta_j \ll (\rho\tau)^{-1}. \quad (9.61)$$

Этот результат свидетельствует о том, что $(\rho\tau)^{-1}$ играет роль, аналогичную роли параметра регуляризации λ при уменьшении весов.

- 9.7** (**) Рассмотрим нейронную сеть, в которой несколько весов ограничены одним и тем же значением. Определите, как необходимо модифицировать стандартный алгоритм обратного распространения, чтобы обеспечить выполнение таких ограничений при вычислении производных функции ошибки по регулируемым параметрам сети.
- 9.8** (*) Рассмотрим распределение смеси, определяемое

$$p(w) = \sum_{j=1}^M \pi_j \mathcal{N}(w | \mu_j, \sigma_j^2), \quad (9.62)$$

где $\{\pi_j\}$ можно рассматривать как априорные вероятности $p(j)$ для соответствующих гауссовых компонент. Используя теорему Байеса, докажите, что соответствующие апостериорные вероятности $p(j|w)$ определяются в (9.24).

- 9.9** (**) Используя (9.21), (9.22), (9.23) и (9.24), проверьте результат (9.25).
- 9.10** (**) Используя (9.21), (9.22), (9.23) и (9.24), проверьте результат (9.26).
- 9.11** (**) Используя (9.21), (9.22), (9.23) и (9.24), проверьте результат (9.28).
- 9.12** (**) Докажите, что производные от коэффициентов смешивания $\{\pi_k\}$, определяемых в (9.30) по вспомогательным параметрам $\{\eta_j\}$, имеют вид:

$$\frac{\partial \pi_k}{\partial \eta_j} = \delta_{jk} \pi_j - \pi_j \pi_k. \quad (9.63)$$

Теперь, используя ограничение $\sum_k \gamma_k(w_i) = 1$ для всех i , получите результат (9.31).

- 9.13** (*) Подтвердите, что объединение (9.35), (9.36) и (9.37) дает одно общее уравнение для всей сети в виде (9.40).
- 9.14** (**) Ожидаемая среднеквадратичная ошибка E_{AV} для простой модели комитета может быть определена в (9.46), а ожидаемая ошибка самого комитета определяется в (9.47). Предполагая, что отдельные ошибки удовлетворяют (9.48) и (9.49), выведите результат (9.50).
- 9.15** (**) Используя неравенство Йенсена (2.102) для частного случая выпуклой функции $f(x) = x^2$, докажите, что средняя ожидаемая среднеквадратичная ошибка E_{AV} членов простой модели комитета, заданная в (9.46), и ожидаемая ошибка E_{COM} самого комитета, заданная в (9.47), удовлетворяют

$$E_{\text{COM}} \leq E_{\text{AV}}. \quad (9.64)$$

9.16 (**) Используя неравенство Йенсена в формуле (2.102), докажите, что результат (9.64), полученный в предыдущем упражнении, справедлив для любой функции ошибки $E(y)$, а не только для суммы квадратов, при условии, что она является выпуклой функцией от y .

9.17 (**) Рассмотрим комитет, где допустимы неравные веса составляющих его моделей, так что

$$y_{\text{COM}}(\mathbf{x}) = \sum_{m=1}^M \alpha_m y_m(\mathbf{x}). \quad (9.65)$$

Чтобы прогнозы $y_{\text{COM}}(\mathbf{x})$ оставались в разумных пределах, предположим, что при каждом значении \mathbf{x} они должны быть ограничены минимальным и максимальным значениями, указанными любым из членов комитета, так что

$$y_{\min}(\mathbf{x}) \leq y_{\text{COM}}(\mathbf{x}) \leq y_{\max}(\mathbf{x}). \quad (9.66)$$

Докажите, что необходимым и достаточным условием для этого ограничения являются коэффициенты α_m , удовлетворяющие следующим условиям:

$$\alpha_m \geq 0, \quad \sum_{m=1}^M \alpha_m = 1. \quad (9.67)$$

9.18 (***) В данном упражнении исследуется влияние регуляризации прореживания на простую модель линейной регрессии, обученную по методу наименьших квадратов. Рассмотрим модель вида

$$y_k = \sum_{i=1}^D w_{ki} x_i \quad (9.68)$$

вместе с функцией ошибки в виде суммы квадратов, представленной

$$E(\mathbf{W}) = \sum_{n=1}^N \sum_{k=1}^K \left\{ t_{nk} - \sum_{i=1}^D w_{ki} R_{ni} x_{ni} \right\}^2, \quad (9.69)$$

где элементы $R_{ni} \in \{0, 1\}$ матрицы прореживания выбираются случайным образом из распределения Бернулли с параметром ρ . Теперь возьмем ожидание по распределению случайных параметров прореживания. Докажите, что

$$\mathbb{E}[R_{ni}] = \rho, \quad (9.70)$$

$$\mathbb{E}[R_{ni} R_{nj}] = \delta_{ij} \rho + (1 - \delta_{ij}) \rho^2. \quad (9.71)$$

Отсюда следует, что ожидаемая функция ошибки для этой модели отсева имеет вид:

$$\mathbb{E}[E(\mathbf{W})] = \sum_{n=1}^N \sum_{k=1}^K \left\{ y_{nk} - \rho \sum_{i=1}^D w_{ki} x_{ni} \right\}^2 \quad (9.72)$$

$$+ \rho(1-\rho) \sum_{n=1}^N \sum_{k=1}^K \sum_{i=1}^D w_{ki}^2 x_{ni}^2. \quad (9.73)$$

Получается, что ожидаемая функция ошибки соответствует сумме квадратов ошибки с квадратичным регуляризатором, в котором коэффициент регуляризации масштабируется отдельно для каждой входной переменной в соответствии со значениями данных, наблюдаемых на этом входе. Наконец, запишите решение в замкнутой форме для весовой матрицы, которая минимизирует эту регуляризованную функцию ошибки.

Глава 10

Сверточные сети

При использовании простейших моделей машинного обучения предполагается, что наблюдаемые значения данных не структурированы, т. е. элементы векторов данных $x = (x_1, \dots, x_D)$ рассматриваются в том виде, будто заранее ничего не известно о возможных связях отдельных элементов между собой. Если произвести произвольную перестановку порядка следования этих переменных и последовательно применить эту фиксированную перестановку ко всем обучающим и тестовым данным, то никаких отличий в производительности рассмотренных до сих пор моделей не будет.

Однако во многих приложениях машинного обучения используются *структурированные данные* с дополнительными взаимосвязями между входными переменными. Например, слова в естественном языке образуют *последовательность* (*sequence*) (см. главу 12), и, если моделировать язык в виде генеративного авторегрессионного процесса, можно было бы ожидать, что каждое слово будет сильнее зависеть от ближайших к нему слов и меньше – от слов, находящихся значительно раньше в этой последовательности. Точно так же пиксели изображения имеют четко определенную пространственную взаимосвязь, в которой входные переменные расположены в двумерной сетке, а близлежащие пиксели имеют высокую корреляцию значений.

Ранее уже отмечалось, что наши знания о структуре конкретных модальностей данных (см. раздел 9.1) могут быть использованы путем добавления регуляризующего члена к функции ошибки в обучающей задаче путем дополнения данных или модификации архитектуры модели. Такие методы могут помочь модели соблюдать определенные свойства (см. раздел 9.1.4), такие как инвариантность и эквивариантность по отношению к преобразованиям входных данных. В этой главе рассматривается архитектурный принцип под названием *сверточная нейронная сеть* (*Convolutional neural network*, *CNN*). Иногда также встречается вариант *конволюционная нейронная сеть*, что является прямой калькой с английского языка. – Прим. перев.). Такую архитектуру можно рассматривать в качестве многослойной сети с малым числом связей и общим доступом к параметрам, которая предназначена для кодирования инвариантности и эквивариантности, характерных для визуальных данных.

10.1. Компьютерное зрение

Автоматический анализ и интерпретация данных в виде изображений являются основными задачами в области компьютерного зрения и представляют собой одну из основных прикладных сфер машинного обучения (Szeliski, 2022). Исторически компьютерное зрение в значительной степени основывалось на трехмерной проекционной геометрии. Созданные вручную признаки использовались в качестве входных данных для простых алгоритмов обучения (Hartley and Zisserman, 2004). Однако эта область стала одной из первых, куда пришла революция глубокого обучения, главным образом благодаря архитектуре CNN. Хотя эта архитектура изначально была разработана специально для анализа изображений, затем она нашла широкое применение и в других областях, например для анализа последовательных данных (см. главу 12). Совсем недавно конкуренцию сверточным сетям при решении некоторых задач смогли составить альтернативные архитектуры на основе преобразователей.

Существует множество вариантов применения машинного обучения в области компьютерного зрения, среди которых наиболее часто встречаются следующие:

- 1) **классификация** изображений (см. рис. 1.1 1.), например классификация снимков поражений кожи в качестве доброкачественных или злокачественных. Иногда это называют «распознаванием образов»;
- 2) **распознавание** объектов на изображении (см. рис. 10.19 2) и определение их местоположения на изображении, например обнаружение пешеходов по данным камеры автомобиля с автономным управлением;
- 3) **сегментация** изображений (см. рис. 10.26), при которой каждый пиксель классифицируется отдельно, тем самым разделяя изображение на области с общей меткой. Например, природная сцена может быть сегментирована на небо, траву, деревья и здания, в то время как снимок медицинского сканирования может быть сегментирован и разделен на злокачественную структуру и нормальную ткань;
- 4) **создание подписи** (см. рис. 12.27), когда на основе изображения автоматически генерируется текстовое описание;
- 5) **синтез** новых изображений (см. рис. 1.3), например генерирование изображений человеческих лиц. Изображения также могут быть синтезированы на основе текстового ввода с описанием желаемого содержания картинки;
- 6) **ретуширование**, или инпайнтинг (inpainting) (см. рис. 20.9), в процессе которого отдельные участки заменяются синтезированными пикселями, согласующимися с остальной частью изображения. Такая технология используется, например, для удаления нежелательных объектов при редактировании изображений;
- 7) **передача стиля** (style transfer) (см. рис. 10.32), когда входное изображение одного стиля, например фотография, преобразуется в соответ-

ствующее изображение другого стиля, например в живопись масляными красками;

- 8) **сверхвысокое разрешение** (рис. 20.8), когда разрешение изображения повышается за счет увеличения числа пикселей и синтеза связанной с ними высокочастотной информации;
- 9) **предсказание глубины**, при котором один или несколько ракурсов используются для прогноза расстояния сцены от камеры до каждого пикселя на целевом изображении;
- 10) **реконструкция сцены**, в процессе которой одно или несколько двухмерных изображений сцены используются для восстановления трехмерного образа.

10.1.1. Данные изображений

Изображение представляет собой прямоугольный массив пикселей, где каждый пиксель характеризуется яркостью серого цвета или – чаще всего – тройкой из красного, зеленого и синего каналов, каждый из которых имеет свое собственное значение яркости. Значения этой яркости являются неотрицательными числами с максимальными значениями в соответствии с предельными характеристиками камеры или другого аппаратного устройства, используемого для захвата изображения. В дальнейшем значения яркости будут рассматриваться в качестве непрерывных переменных, но на практике они представлены в виде 8-битных чисел с конечной точностью, например в виде целых чисел в диапазоне 0, ..., 255. Некоторые изображения, например снимки магнитно-резонансной томографии (МРТ), используемые в медицинской диагностике, представляют собой трехмерные сетки *вокселей* (*Voxels*, сокращение от англ. *volume pixel*, объемный пиксель. – Прим. перев.). Аналогичным образом видео состоит из последовательности двухмерных изображений, поэтому их также можно рассматривать как трехмерные структуры, в которых последовательные кадры расположены на временной шкале.

Теперь рассмотрим возможность использования нейронных сетей для работы с изображениями с целью решения некоторых из перечисленных выше задач. Изображения, как правило, имеют высокую размерность, а типичные камеры позволяют получать изображения размером в десятки мегапикселей. Таким образом, если рассматривать данные изображения как неструктурированные, то может потребоваться модель с огромным количеством параметров, обучение которой будет практически невозможным. Более того, такой подход не учитывает высокоструктурированную природу данных изображения, в которой важную роль играет взаимное расположение различных пикселей. Если взять пиксели изображения и произвольно переместить их, то результат перестанет быть похожим на естественное изображение. Точно так же, если создать синтетическое изображение методом случайного выбора значений яркости пикселей, независимо для каждого из них, то вероятность получить что-либо похожее на естественное изображение будет практически нулевой (см. рис. 6.8). Локальные корреляции играют важную роль, поэтому

в естественном изображении вероятность того, что два близлежащих пикселя будут иметь схожие цвета и значения яркости, гораздо выше, чем два пикселя, находящихся на большом расстоянии друг от друга. Это мощное предварительное знание может использоваться для кодирования сильных индуктивных смещений в нейронной сети, что позволяет получить модели с гораздо меньшим количеством параметров и гораздо более высокой точностью обобщения.

10.2. Сверточные фильтры

Одна из причин появления сверточных сетей заключается в том, что для данных изображений, для которых и были разработаны такие сети, применение стандартной архитектуры с полной связностью потребовало бы огромного количества параметров из-за высокой размерности изображений. Чтобы убедиться в этом, рассмотрим цветное изображение размером $10^3 \times 10^3$ пикселей, каждый из которых имеет три значения, соответствующие яркости красного, зеленого и синего цветов. Если первый скрытый слой сети имеет, например, 1000 скрытых элементов, то уже в первом слое будет 3×10^9 весов. Кроме того, такая сеть должна будет обучаться любым инвариантностям и эквивариантностям на примерах, что потребует огромных наборов данных. Если создать архитектуру с учетом наших индуктивных предпочтений относительно структуры изображений, то можно значительно понизить объем требуемых наборов данных, а также улучшить обобщение с учетом симметрий в пространстве изображений.

Чтобы использовать двумерную структуру данных изображений для создания индуктивных смещений, можно задействовать четыре взаимосвязанных принципа: иерархию, локальность, эквивариантность и инвариантность. Рассмотрим задачу распознавания лиц на изображениях. Существует естественная иерархическая структура, поскольку одно изображение может содержать несколько лиц, и каждое лицо включает такие элементы, как глаза; при этом каждый глаз имеет такую структуру, как радужная оболочка, которая, в свою очередь, имеет такую структуру, как контуры (края). На самом низком уровне иерархии узел нейронной сети может обнаружить наличие такого элемента, как контуры, основываясь на информации локального характера, относящейся к небольшой области изображения, и поэтому ему необходимо просмотреть лишь небольшое подмножество пикселей изображения. Более сложные структуры, расположенные дальше по иерархии, могут быть обнаружены путем сочетания нескольких признаков, обнаруженных на предыдущих уровнях. Ключевым моментом тем не менее является тот факт, что, несмотря на стремление заложить в модель общую концепцию иерархии, ее детали, включая тип признаков, извлекаемых на каждом уровне, должны быть получены из данных, а не закодированы вручную. Иерархические модели вполне органично вписываются в рамки технологии глубокого обучения, которая как раз и дает возможность извлекать очень сложные

концепции из необработанных данных путем последовательной, возможно, очень многоуровневой обработки, в ходе которой вся система обучается от начала до конца.

10.2.1. Детекторы признаков

Для удобства вначале рассмотрим только изображения в градациях шкалы серого (т. е. изображения с одним каналом). Рассмотрим один блок в первом слое нейронной сети, который принимает на вход только значения пикселей из небольшой прямоугольной области, или фрагмента, полученного из изображения, как показано на рис. 10.1а. Этот фрагмент называется *рецептивным (рецепторным) полем* (*receptive field*) этого блока, что отражает принцип локальности. В дальнейшем желательно, чтобы значения весов, связанные с этим блоком, обучились определять какой-либо полезный низкоуровневый признак. Выход этого блока задается обычной функциональной формой, включающей взвешенную линейную комбинацию входных значений, которая впоследствии преобразуется с помощью нелинейной функции активации:

$$z = \text{ReLU}(\mathbf{w}^T \mathbf{x} + w_0), \quad (10.1)$$

где \mathbf{x} – это вектор значений пикселей рецептивного поля, и здесь подразумевается функция активации ReLU. Поскольку с каждым входным пикселием связан один вес, то значения весов образуют небольшую двумерную сетку, известную как *фильтр (filter)*, иногда также называемую *ядром (kernel)*, которое само по себе можно представить в виде изображения. Этот пример показан на рис. 10.1. Здесь на рис. 10.1а изображен блок в скрытом слое сети,

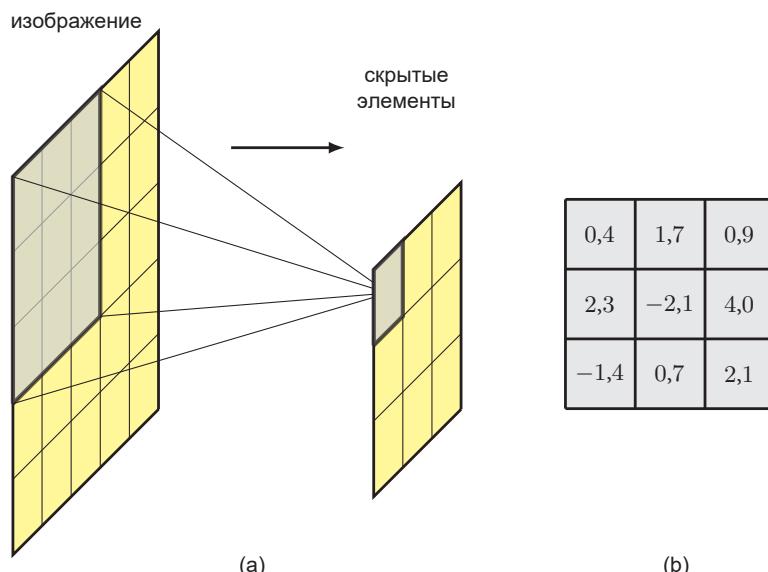


РИС. 10.1 Схема рецептивного поля

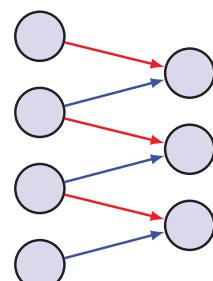
который получает входные данные от пикселей на участке изображения размером 3×3 . Пиксели этой области образуют рецептивное поле для данного блока. На рис. 10.1b приведены весовые значения, связанные с этим скрытым блоком, которые можно представить в виде небольшой матрицы 3×3 . Именно это и называется ядром. Существует также дополнительный параметр смещения, который на рисунке не показан.

Предположим, что w и w_0 в (10.1) фиксированы, и зададимся вопросом, при каком значении входного фрагмента изображения x этот скрытый блок даст наибольший выходной отклик. Чтобы ответить на этот вопрос, необходимо каким-то образом ограничить x , поэтому предположим, что его норма $\|x\|^2$ фиксирована. Тогда решение для x , которое максимизирует $w^T x$, а значит, максимизирует отклик скрытого блока (см. упражнение 10.1), имеет вид $x = \alpha w$ при некотором коэффициенте α . Это означает, что максимальный выходной отклик этого скрытого блока происходит в том случае, если он определяет участок изображения, который вплоть до общего масштабирования выглядит как образ ядра. Обратите внимание, что ReLU генерирует ненулевой выход только тогда, когда $w^T x$ превышает порог $-w_0$, и поэтому блок работает как детектор признаков, подающий сигнал при нахождении достаточно высокого совпадения со своим ядром.

10.2.2. Эквивариантный перенос

Теперь следует отметить, что если небольшой участок на изображении лица соответствует, например, глазу в этом месте, то тот же самый набор значений пикселей в другой части изображения обязательно должен представлять глаз в новом месте. Наша нейронная сеть должна уметь обобщать то, что она узнала в одном месте, на все возможные места на изображении без необходимости находить в обучающем наборе примеры соответствующего признака в каждом возможном месте. Чтобы добиться этого, мы можем просто повторить одни и те же значения весов скрытых элементов во всех местах изображения, как показано на рис. 10.2 для одномерного входного пространства. Здесь связи разрежены и являются общими для скрытых элементов, как показано красными и синими стрелками, где связи одного цвета имеют одинаковые значения веса. Таким образом, эта сеть имеет шесть связей, но только два независимых обучаемых параметра.

РИС. 10.2 Иллюстрация свертки для одномерного массива входных значений и ядра шириной 2



Элементы скрытого слоя образуют *карту признаков* (*feature map*), в которой все элементы имеют одинаковые веса. Следовательно, если локальный участок изображения вызывает определенный отклик в элементе, связанном с этим участком, то тот же набор значений пикселей в другом месте будет вызывать аналогичный отклик в соответствующей *перенесенной* (*translated*) точке карты признаков (см. раздел 9.1.3). Именно так и выглядит *эквивариантность* (*equivariance*). В этой сети можно заметить разреженные связи, так как большинство связей отсутствуют. Кроме того, значения весов являются общими (*shared*) для всех скрытых элементов, о чем свидетельствуют цвета связей. Такое преобразование является примером *свертки* (*convolution*).

Концепцию свертки можно распространить на двумерные изображения следующим образом (Dumoulin and Visin, 2016). Для изображения I с яркостью пикселей $I(j, k)$ и фильтра K со значениями пикселей $K(l, m)$ карта признаков C имеет значения активации, заданные как

$$C(j, k) = \sum_l \sum_m I(j + l, k + m)K(l, m), \quad (10.2)$$

где нелинейная функция активации для наглядности опущена. Это также пример *свертки*, которая иногда выражается как $C = I * K$. Обратите внимание, что, строго говоря, в (10.2) используется *взаимная корреляция*, что несколько отличается от обычного математического определения свертки (см. упражнение 10.4), но здесь и далее будем следовать общепринятой практике в литературе по машинному обучению и называть (10.2) сверткой. Взаимосвязь (10.2) проиллюстрирована на рис. 10.3 для изображения 3×3 и фильтра 2×2 (см. раздел 7.4.2). Важно отметить, что при использовании пакетной нормализации в сверточной сети необходимо использовать одинаковые значения среднего значения и дисперсии в каждом пространственном положении на карте признаков при нормализации состояний элементов, чтобы статистика карты признаков не зависела от их места.

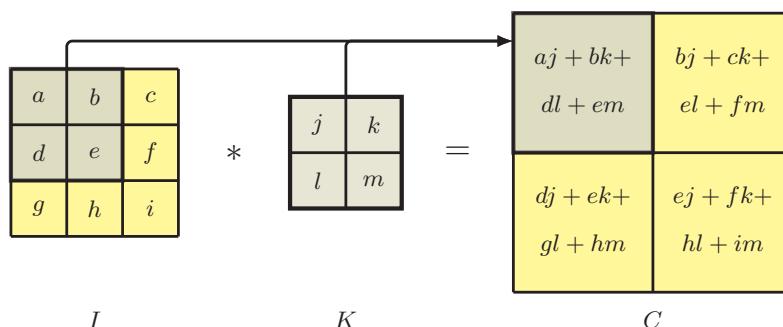


РИС. 10.3 Пример изображения 3×3 , свернутого с фильтром 2×2 для получения карты признаков 2×2

В качестве примера применения свертки рассмотрим задачу распознавания краев на изображениях с помощью фиксированного, созданного вручную

сверточного фильтра. На первый взгляд, вертикальный край можно рассматривать как момент, когда происходит значительное локальное изменение уровня яркости между пикселями при горизонтальном перемещении по изображению. Это можно измерить, свернув изображение с фильтром 3×3 вида

$$\begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad (10.3)$$

Аналогичным образом можно обнаружить горизонтальные края, свернув изображение с транспонированием этого фильтра:

$$\begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad (10.3)$$

На рис. 10.4 показаны результаты применения этих двух сверточных фильтров к образцу изображения. Обратите внимание, что на рис. 10.4b, когда вертикальный край соответствует увеличению яркости пикселя, соответствующая точка на карте признаков положительна (обозначена светлым оттенком), тогда как в случае, когда вертикальный край соответствует уменьшению яркости пикселя, соответствующая точка на карте признаков отрицательна (обозначена темным оттенком), с аналогичными свойствами на рис. 10.4c для горизонтальных краев

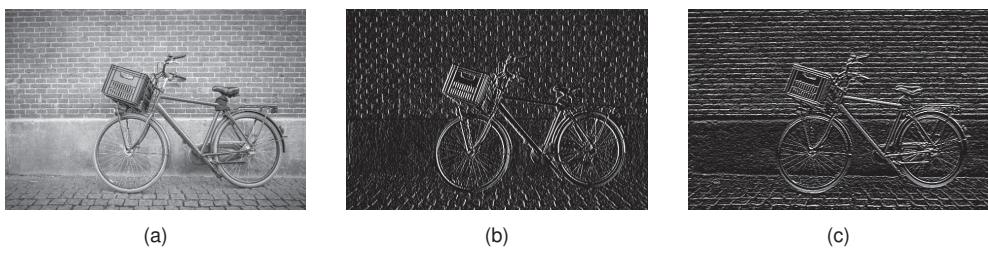


РИС. 10.4 Иллюстрация определения краев с помощью сверточных фильтров: (a) исходное изображение, (b) результат свертки с фильтром (10.3 по вертикальным краям) и (c) результат свертки с фильтром (10.4) по горизонтальным краям

Сравнивая эту сверточную структуру со стандартной полноценной сетью, можно выделить несколько преимуществ: (i) связи разрежены, что приводит к гораздо меньшему количеству весов даже для больших изображений; (ii) значения весов разделяются, что значительно сокращает количество независимых параметров и, соответственно, уменьшает размер обучающего набора, необходимого для обучения этих параметров; (iii) одну и ту же сеть можно применять к изображениям разного размера без необходимости переобучения. К последнему пункту вернемся позже (см. раздел 10.4.3),

а пока просто отметим, что изменение размера входного изображения приводит к простому изменению размера карты признаков, но не изменяет количество весов или число независимых обучаемых параметров в модели. И последнее замечание относительно сверточных сетей: благодаря использованию массивного параллелизма графических процессоров (GPU) для достижения высокой вычислительной производительности сверточные сети могут быть реализованы чрезвычайно эффективным образом.

10.2.3. Заполнение

На рис. 10.3 показано, что карта свертки меньше исходного изображения. Если изображение имеет размерность $J \times K$ пикселей и оно свернуто с ядром размерности $M \times M$ (фильтры обычно выбираются квадратными), то результирующая карта признаков имеет размерность $(J - M + 1) \times (K - M + 1)$. В некоторых случаях желательно, чтобы карта признаков имела те же размеры, что и исходное изображение. Для этого исходное изображение *заполняется (padding)* дополнительными пикселями по периметру, как показано на рис. 10.5. Если добавить P пикселей, то выходная карта будет иметь размерность $(J + 2P - M + 1) \times (K + 2P - M + 1)$. Если никакого заполнения нет, т. е. $P = 0$, то такая свертка называется *правильной (valid convolution)*. Если значение P выбрано таким образом, что выходной массив имеет тот же размер, что и входной (см. упражнение 10.6), что соответствует $P = (M - 1)/2$, то такая свертка называется *идентичной (same convolution)*, поскольку изображение и карта признаков имеют одинаковые размеры. В компьютерном зрении фильтры обычно используют нечетные значения M , чтобы заполнение было симметричным по всем сторонам изображения и чтобы существовал четко определенный центральный пиксель, связанный с расположением фильтра. Наконец, необходимо выбрать подходящее значение яркости, связанное с пикселями заполнения. Типичным вариантом является установка нулевого значения яркости пикселей после предварительного вычитания среднего значения из каждого изображения, чтобы ноль представлял собой среднее значение яркости пикселей. Кроме того, заполнение может применяться к картам признаков для обработки последующими сверточными слоями.

РИС. 10.5 Иллюстрация изображения 4×4 , которое было заполнено дополнительными пикселями для создания изображения 6×6

0	0	0	0	0	0
0	X_{11}	X_{12}	X_{13}	X_{14}	0
0	X_{21}	X_{22}	X_{23}	X_{24}	0
0	X_{31}	X_{32}	X_{33}	X_{34}	0
0	X_{41}	X_{42}	X_{43}	X_{44}	0
0	0	0	0	0	0

10.2.4. Свертки со сдвигом

В стандартных задачах обработки изображений количество пикселей на них может быть очень большим, и, поскольку ядра часто относительно малы, так что $M \ll J, K$, сверточная карта признаков будет иметь размер, близкий к исходному изображению, и будет такого же размера, если используется такое же заполнение. Иногда требуется использовать карты признаков, которые значительно меньше исходного изображения, чтобы обеспечить гибкость при проектировании архитектур сверточных сетей. Один из способов добиться этого – использовать *свертки с шагом* (*strided convolutions*), в которых вместо перемещения фильтра по изображению по одному пикселию за раз, он перемещается большими шагами размером S , который, собственно, и называется *шагом*, или *страйдом* (*stride*). Если использовать одинаковый страйд по горизонтали и вертикали (см. упражнение 10.7), то количество элементов в карте признаков будет равно

$$\left\lfloor \frac{J + 2P - M}{S} - 1 \right\rfloor \times \left\lfloor \frac{K + 2P - M}{S} - 1 \right\rfloor, \quad (10.5)$$

где $\lfloor x \rfloor$ обозначает «нижний уровень» x , т. е. наибольшее целое число, которое меньше или равно x . Для больших изображений и малых размеров фильтров карта изображений будет примерно в $1/S$ раз меньше исходного изображения.

10.2.5. Многомерные свертки

До сих пор рассматривались свертки для одного изображения в серой цветовой гамме. В случае с цветным изображением необходимо использовать три канала, соответствующие красному, зеленому и синему цветам. Конфигурацию сверток можно легко расширить для охвата нескольких каналов за счет увеличения размерности фильтра. Изображение с $J \times K$ пикселей и C каналами будет описываться тензором размерности $J \times K \times C$ (см. раздел 6.3.7). Теперь можно ввести фильтр, описываемый тензором размерности $M \times M \times C$, который включает в себя отдельный фильтр $M \times M$ для каждого из C каналов. Предполагая отсутствие заполнений и ширину страйда, равную 1, это вновь приводит к карте признаков размером $(J - M + 1) \times (K - M + 1)$, как показано на рис. 10.6.

Теперь рассмотрим еще одно важное расширение сверток. До сих пор речь шла о создании единой карты признаков, в которой все ее точки имеют один и тот же набор обучаемых параметров. Для фильтра размерности $M \times M \times C$ это будет $M^2 C$ весовых параметров независимо от размера изображения. Кроме того, с этим элементом будет связан параметр смещения. Такой фильтр аналогичен одному скрытому узлу в полносвязной сети, он может научиться обнаруживать только один вид признаков и поэтому очень ограничен. Для построения более гибких моделей необходимо просто включить несколько таких фильтров, при этом каждый фильтр будет иметь свой собственный

независимый набор параметров, порождающий свою собственную независимую карту признаков, как показано на рис. 10.7. Эти отдельные карты признаков в дальнейшем будут называться *каналами (channels)*. Тензор фильтров теперь будет иметь размерность $M \times M \times C \times C_{\text{OUT}}$, где C – это количество входных

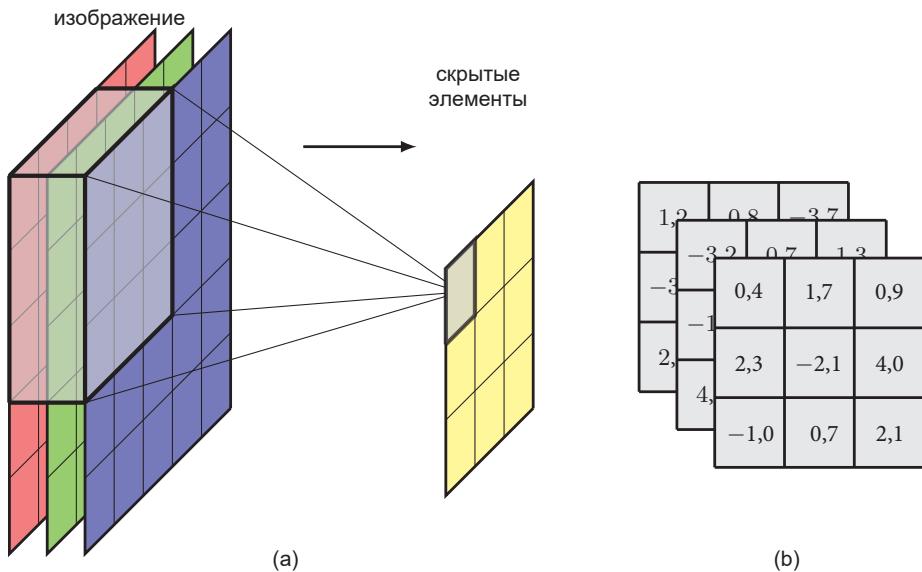


РИС. 10.6 (a) Схема многомерного фильтра с входами из каналов R, G и B. (b) Ядро имеет 27 весов (плюс параметр смещения, который не показан) и может быть визуализировано как тензор $3 \times 3 \times 3$

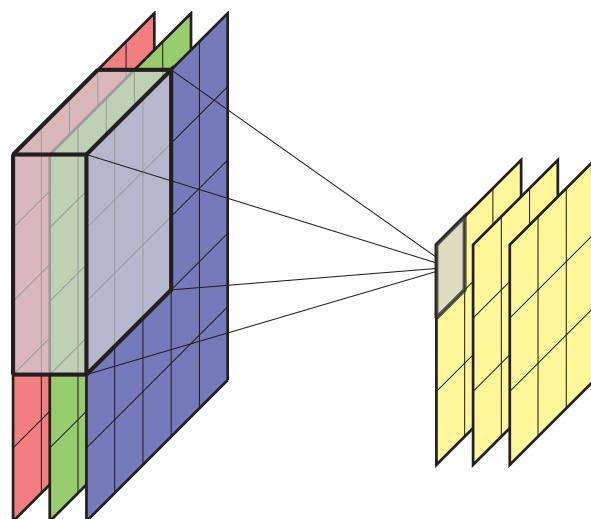


РИС. 10.7 Многомерный слой сверточного фильтра, показанный на рис. 10.6, можно расширить, включив в него несколько независимых каналов фильтрации

каналов, а C_{OUT} – это количество выходных каналов. Каждый выходной канал будет иметь свой собственный параметр смещения, поэтому общее количество параметров составит $(M^2C + 1)C_{\text{OUT}}$.

Полезной концепцией при проектировании сверточных сетей является *свертка 1×1* (Lin, Chen and Yan, 2013), представляющая собой простой сверточный слой, в котором размер фильтра составляет один пиксель. Фильтры имеют C весов, по одному для каждого входного канала, плюс смещение. Одним из вариантов применения сверток 1×1 является простое изменение количества каналов (обычно в сторону их уменьшения) без изменения размера карт признаков путем задания количества выходных каналов, отличного от количества входных. Таким образом, он дополняет свертки или пулинг в том плане, что уменьшает количество каналов, а не их размерность.

10.2.6. Пулинг

В сверточном слое кодируется эквивариантность преобразования, вследствие чего при перемещении небольшого участка пикселей, представляющего рецептивное поле скрытого элемента, в другое место изображения соответствующие выходы карты признаков будут перемещаться в соответствующее место карты признаков. Это свойство особенно эффективно при решении таких задач, как определение местоположения объекта на изображении. Для других приложений, таких как классификация изображений, важно, чтобы выходные данные были инвариантны к преобразованиям входных данных. Однако во всех случаях необходимо обеспечить способность сети к обучению иерархической структуре, где сложные признаки определенного уровня формируются из более простых признаков предыдущего уровня. Во многих случаях важны пространственные отношения между этими более простыми признаками. Например, определение лица по относительному расположению глаз, носа и рта, а не только по наличию этих признаков в произвольных местах изображения. Однако небольшие изменения в относительном расположении не влияют на классификацию, и поэтому необходимо обеспечить инвариантность к таким небольшим перемещениям отдельных признаков. Этого можно достичь с помощью *объединения в пул*, или *пулинга* (*pooling*), применяемого к выходу сверточного слоя.

Метод пулинга похож на использование сверточного слоя, поскольку массив блоков расположен в виде сетки, и каждый блок получает входные данные от рецептивного поля в предыдущем слое карты признаков. Здесь также есть выбор размера фильтра и длины шага. Разница между ними заключается в том, что выход блока пулинга является простой, фиксированной функцией его входов, и поэтому в пулинге нет обучаемых параметров. Распространенным примером функции пулинга является *подвыборка с определением максимального значения*, или *максимальный пулинг* (*max pooling*) (Zhou and Chellappa, 1988), где каждый элемент просто выводит функцию *max*, примененную ко входным значениям. Это показано на простом примере на рис. 10.8. Здесь длина шага равна ширине фильтра, поэтому рецептивные поля не перекрываются.

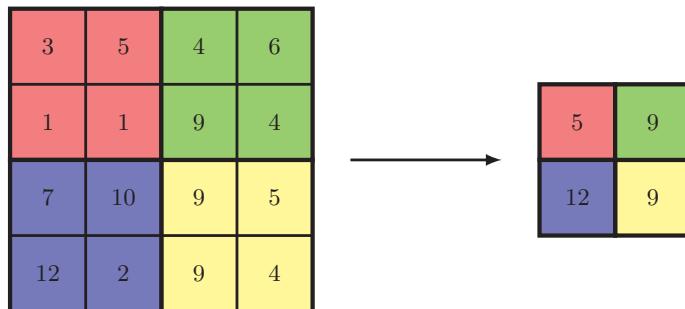


РИС. 10.8 Иллюстрация метода максимального пулинга, при котором блоки пикселей размером 2×2 в карте признаков объединяются с помощью оператора *max* для создания новой карты признаков меньшей размерности

Помимо создания некоторой локальной инвариантности преобразования, пулинг можно также использовать для уменьшения размерности представления путем понижения выборки карты признаков. Обратите внимание, что использование страйдов размером более 1 в сверточном слое также приводит к уменьшению выборки карт признаков.

Активацию элемента в карте признаков можно интерпретировать как меру эффективности выявления соответствующего признака, так что функция максимального пулинга сохраняет информацию о наличии признака и его выраженности, но отбрасывает часть информации о положении. Существует множество других вариантов функции пулинга, например *средний* (усредненный) пулинг (*average pooling*), при котором функция пулинга вычисляет среднее значение показателей соответствующего рецептивного поля в карте признаков. Все они обеспечивают определенную степень локальной инвариантности преобразования.

Обычно пулинг применяется к каждому каналу карты признаков по отдельности. Например, если есть карта признаков с 8 каналами, каждый из которых имеет размерность 64×64 , и к ней применяется максимальный пулинг с рецептивным полем размером 2×2 и страйдом (шагом) 2, то результатом операции пулинга будет тензор размерности $32 \times 32 \times 8$.

Кроме того, пулинг также можно применить к нескольким каналам карты признаков, что открывает перед сетью возможности для обучения другим инвариантам, помимо простой инвариантности преобразования. Например, если несколько каналов в сверточном слое учатся обнаруживать один и тот же признак, но в разных ориентациях, то максимальный пулинг по этим картам признаков будет приближенно инвариантен к поворотам.

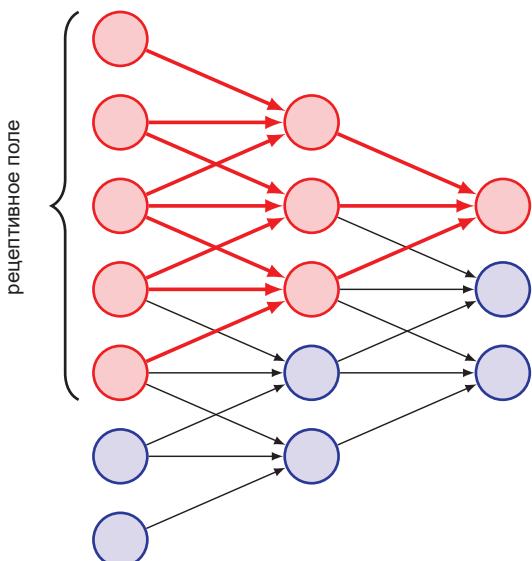
Пулинг также позволяет сверточной сети обрабатывать изображения разного размера. В конечном итоге выходной и, как правило, некоторые промежуточные слои сверточной сети должны быть определенного фиксированного размера. Входные сигналы различного размера можно обрабатывать путем изменения длины страйда пулинга в зависимости от размера изображения, при этом количество объединенных выходов остается постоянным.

10.2.7. Многослойные свертки

Структура сверточных сетей, описанная ранее, аналогична одному слою в обычной полносвязной нейронной сети. Однако для того, чтобы сеть была способна обнаруживать и представлять иерархическую структуру данных, необходимо расширить ее архитектуру и рассмотреть возможность использования нескольких описанных выше слоев. Каждый сверточный слой характеризуется тензором фильтров размерности $M \times M \times C_{\text{IN}} \times C_{\text{OUT}}$, в котором число независимых параметров веса и смещения равно $(M^2 C_{\text{IN}} + 1)C_{\text{OUT}}$. За каждым таким сверточным слоем может опционально следовать слой пулинга. В результате можно использовать несколько таких слоев свертки и пулинга поочередно, при этом выходные каналы C_{OUT} определенного слоя, аналогичные каналам RGB входного изображения, формируют входные каналы следующего слоя. Обратите внимание, что количество каналов в карте признаков иногда называют *глубиной* (*depth*) карты признаков, но обычно под глубиной все же подразумевают количество слоев многослойной сети.

Ключевым свойством, заложенным в сверточную структуру, является локальность: каждый элемент карты признаков получает информацию только от небольшого участка, *рецептивного поля* (*receptive field*) предыдущего слоя. При создании глубокой нейронной сети, в которой каждый слой является сверточным, эффективное рецептивное поле элемента в более поздних слоях сети становится намного больше, чем в более ранних, как показано на рис. 10.9. Здесь показано, что красный элемент в верхней части выходного слоя получает входные сигналы от рецептивного поля в среднем слое элементов, каждый из которых имеет рецептивное поле в первом слое элементов. Таким образом, активация красного элемента в выходном слое зависит от выходов 3 элементов в среднем слое и 5 элементов во входном слое.

РИС. 10.9 Иллюстрация увеличения эффективного рецептивного поля с глубиной в многослойной сверточной сети



Во многих случаях выходные блоки сети должны делать прогнозы относительно изображения в целом, например в задаче классификации, и поэтому им необходимо объединить информацию по всему входному изображению. Обычно это достигается введением одного или двух стандартных полно связных слоев на финальных участках сети, в которых каждый элемент связан с каждым элементом предыдущего слоя. Количество параметров в такой архитектуре может быть управляемым, поскольку финальный сверточный слой обычно имеет гораздо меньшую размерность, чем входной слой, из-за промежуточных объединяющих слоев. Тем не менее конечные полно связные слои могут включать большую часть независимых степеней свободы в сети, даже если количество (совместных) связей в сети больше в сверточных слоях.

Таким образом, полная сверточная нейронная сеть состоит из нескольких слоев сверток, которые чередуются с операциями пулинга, и зачастую с обычными полно связными слоями на финальных участках сети. При проектировании такой архитектуры необходимо принимать многочисленные решения, включая вопрос о количестве слоев, количестве каналов в каждом слое, размерах фильтров, ширине страйдов и множестве других гиперпараметров. К настоящему времени уже изучено множество различных архитектур, однако на практике систематическое сравнение значений гиперпараметров с использованием удержаных данных затруднено из-за больших вычислительных затрат на обучение каждой конфигурации-кандидата.

10.2.8. Примеры сетевых архитектур

Сверточные сети являются первыми глубокими нейронными сетями (т. е. сетями с более чем двумя обучамыми слоями параметров), которые были успешно применены в прикладных целях. Одним из первых примеров стала сеть *LeNet*, которая использовалась для классификации монохромных изображений рукописных цифр низкого разрешения (LeCun et al., 1989; LeCun et al., 1998). Разработка более мощных сверточных сетей значительно ускорилась с появлением крупного эталонного набора данных *ImageNet* (Deng et al., 2009), состоящего из 14 млн естественных изображений, каждое из которых было вручную маркировано по одной из почти 22 000 категорий. Этот набор данных значительно превышал по объему все ранее использовавшиеся, и успехи в этой области, достигнутые благодаря *ImageNet*, подчеркнули важность крупномасштабных данных, а также хорошо разработанных моделей с соответствующими индуктивными смещениями для создания эффективных решений в области глубокого обучения.

Подмножество изображений, состоящее из 1000 непересекающихся категорий, легло в основу ежегодного крупномасштабного конкурса визуального распознавания *ImageNet Large Scale Visual Recognition Challenge*. И вновь это число категорий оказалось намного больше по сравнению с теми несколькими десятками классов, которые обычно рассматривались ранее. Наличие такого количества категорий значительно усложняет задачу, поскольку, если бы классы были распределены равномерно, случайное угадывание привело

бы к погрешности в 99,9 %. Набор данных насчитывает чуть более 1,28 млн обучающих, 50 000 проверочных и 100 000 тестовых изображений. Классификаторы предназначены для создания ранжированного списка прогнозируемых классов на тестовых изображениях, а результаты представлены в виде коэффициентов ошибок топ-1 и топ-5. Это означает, что изображение считается правильно классифицированным в том случае, если истинный класс находится в верхней части списка или в одном из пяти самых высоких рейтингов прогнозируемых классов. Ранние результаты, полученные на этом наборе данных, позволили достичь уровня погрешности топ-5 порядка 25,5 %. Важным достижением стало появление сверточной сетевой архитектуры *AlexNet* (Krizhevsky, Sutskever and Hinton, 2012), которая победила в конкурсе 2012 года и снизила уровень ошибок в топ-5 до нового рекорда в 15,3 %. Ключевыми аспектами этой модели стали использование функции активации ReLU, применение графических процессоров для обучения сети и использование регуляризации отсева (см. раздел 9.6.1). В последующие годы был достигнут дальнейший прогресс, в результате чего процент ошибок достиг примерно 3 %, что даже несколько лучше, чем у человека, который при работе с теми же данными допускает около 5 % ошибок (Dodge and Karam, 2017). Такой результат можно объяснить трудностями, с которыми сталкивается человек при выделении слабо различимых классов (например, нескольких разновидностей грибов).

В качестве примера типичной архитектуры сверточной сети рассмотрим в деталях модель VGG-16 (Simonyan and Zisserman, 2014), где VGG – это компания Visual Geometry Group, разработавшая модель, а 16 – это количество обучаемых слоев в модели. У VGG-16 довольно простые принципы построения, что позволило создать относительно однородную архитектуру, как показано на рис. 10.10, и минимизировать количество вариантов выбора гиперпараметров. На вход поступает изображение размером 224×224 пикселя с тремя цветовыми каналами, затем следуют наборы сверточных слоев, чередующиеся с понижающей дискретизацией. Все сверточные слои имеют фильтры размером 3×3 со страйдом 1, одинаковым заполнением и функцией активации ReLU, в то время как операции максимального пулинга используют страйд 2 и размер фильтра 2×2 , тем самым уменьшая количество элементов в 4 раза. Первый обучаемый слой – это сверточный слой, в котором каждый элемент получает входные данные от «кубика» размером $3 \times 3 \times 3$ из стека входных каналов с 28 параметрами, включая смещение. Эти параметры разделяются между всеми элементами карты признаков для каждого канала. В первом слое 64 таких канала признаков, что дает на выходе тензор размером $224 \times 224 \times 64$. Второй слой также является сверточным и также имеет 64 канала. Затем следует максимальный пулинг с картами признаков размером 112×112 . Слои 3 и 4 также являются сверточными с размерностью 112×112 , и в каждом из них имеется 128 каналов. Такое увеличение числа каналов в некоторой степени компенсирует понижающую дискретизацию в слое максимального пулинга, чтобы число переменных в представлении на каждом слое не уменьшалось слишком быстро по всей сети. За этими

слоями снова следует операция максимального пулинга для создания карты признаков размером 56×56 . Далее следуют еще три сверточных слоя с 256 каналами каждый, что вновь приводит к удвоению числа каналов в результате уменьшения дискретизации. Затем следует еще один максимальный пулинг для получения карты признаков размером 28×28 , после чего еще три сверточных слоя по 512 каналов, далее еще один максимальный пулинг, который понижает выборку до карты признаков размером 14×14 . За этим следуют еще три сверточных слоя, но количество карт признаков в этих слоях сохраняется на уровне 512, после чего следует еще один максимальный пулинг, который уменьшает размер карт признаков до 7×7 . Наконец, есть еще три *полносвязных* слоя, т. е. стандартных нейросетевых слоя с полной связностью и без обмена параметрами. Последний слой максимального пулинга имеет 512 каналов размером 7×7 , что в сумме дает 25 088 элементов. Первый полно связный слой имеет 4096 элементов, каждый из которых соединен с каждым из элементов максимального пулинга. За ним следует второй полно связный слой опять же с 4096 элементами, и, наконец, третий полно связный слой с 1000 элементами, чтобы сеть можно было применить к задаче классификации с использованием 1000 классов. Все обучаемые слои сети работают с нелинейными функциями активации ReLU, за исключением выходного слоя, где используется функция активации softmax. Всего в VGG-16 насчитывается около 138 млн независимо обучаемых параметров, большинство из которых (почти 103 млн) находятся в первом полно связном слое, а большинство связей приходится на первый сверточный слой (см. упражнение 10.8).

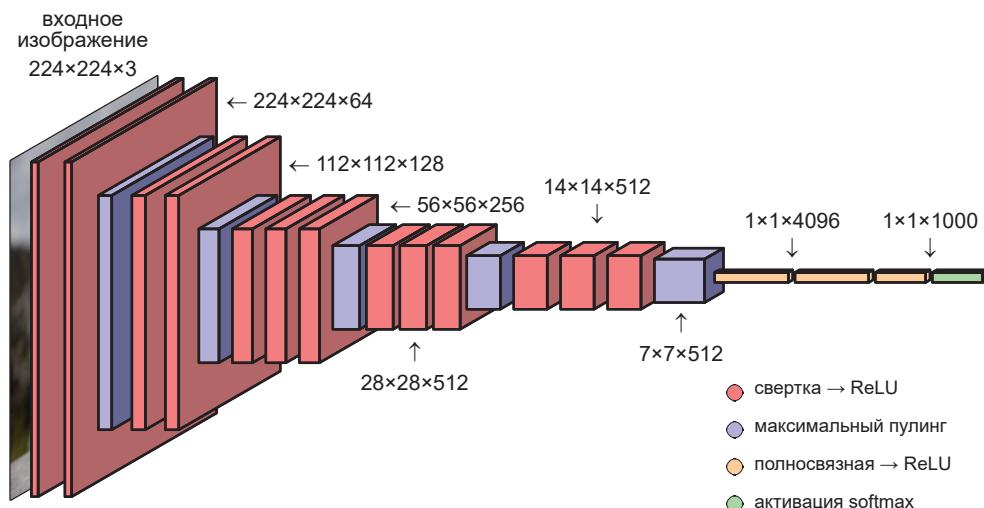


РИС. 10.10 Архитектура типичной сверточной сети, в данном случае модели VGG-16

Ранние CNN обычно имели меньшее количество сверточных слоев, поскольку у них были большие receptive поля. Например, Alexnet (Krizhevsky, Sutskever and Hinton, 2012) характеризуется receptive полями 11×11

с шагом 4. На рис. 10.9 было показано, что больших рецептивных полей можно также добиться неявным образом, используя несколько слоев с меньшими рецептивными полями в каждом. Преимущество последнего подхода заключается в том, что он требует значительно меньшего количества параметров, фактически накладывая индуктивное смещение на большие фильтры, поскольку их приходится составлять из сверточных субфильтров. Несмотря на сложность такой архитектуры, только сама функция сети должна быть закодирована в явном виде, поскольку производные функции издержек (см. раздел 8.2) могут быть оценены с помощью автоматического дифференцирования, а функция издержек оптимизирована с помощью стохастического градиентного спуска.

10.3. Визуализация обученных CNN

Далее рассмотрим особенности современных глубоких CNN, которые имеют удивительное сходство с характеристиками зрительной коры млекопитающих.

10.3.1. Зрительная кора головного мозга

Исторически сложилось так, что основой для создания сверточных нейронных сетей стали передовые исследования в области нейронауки, которые позволили разобраться в природе обработки зрительных сигналов у млекопитающих, в том числе и у человека. Электрические сигналы от сетчатки преобразуются через ряд обрабатывающих слоев в зрительной коре, которая находится в задней части мозга. Там нейроны организованы в виде двумерных листов, каждый из которых формирует карту двумерного зрительного поля. В своей новаторской работе (Hubel and Wiesel, 1959) авторы провели измерения электрических откликов отдельных нейронов в зрительной коре кошек при подаче зрительных стимулов на их глаза. Они обнаружили, что некоторые нейроны, называемые «простыми клетками», вызывают сильную ответную реакцию на зрительные сигналы с простым контуром, ориентированным под определенным углом и расположенным в определенном положении в поле зрения, в то время как другие стимулирующие сигналы приводят к значительно меньшему реагированию этих нейронов. Более детальные исследования показали, что реакция этих простых клеток может быть смоделирована с помощью *фильтров Габора* (*Gabor filters*), которые представляют собой двумерные функции и которые определяются как

$$G(x, y) = A \exp(-\alpha \tilde{x}^2 - \beta \tilde{y}^2) \sin(\omega \tilde{x} + \varphi), \quad (10.6)$$

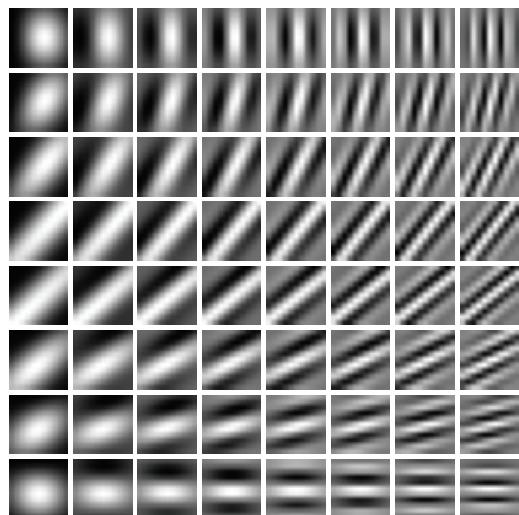
где

$$\tilde{x} = (x - x_0) \cos(\theta) + (y - y_0) \sin(\theta), \quad (10.7)$$

$$\tilde{y} = -(x - x_0) \sin(\theta) + (y - y_0) \cos(\theta). \quad (10.8)$$

Уравнения (10.7) и (10.8) описывают поворот системы координат на угол θ , поэтому член $\sin(\cdot)$ в (10.6) представляет собой синусоидальное пространственное колебание, ориентированное в направлении, определяемом полярным углом θ , с частотой ω и фазовым углом φ . Экспоненциальный множитель в (10.6) формирует огибающую затухания, которая локализует фильтр поблизости от положения (x_0, y_0) и со скоростями затухания, регулируемыми α и β . Примеры фильтров Габора показаны на рис. 10.11. Здесь угол ориентации θ изменяется от 0 в верхней строке до $\pi/2$ в нижней строке, а частота изменяется от $\omega = 1$ в левом столбце до $\omega = 10$ в правом столбце.

РИС. 10.11 Примеры фильтров Габора в соответствии с (10.6)



В своей работе (Hubel and Wiesel, 1959) авторы также сообщили об открытии «сложных клеток» (*complex cells*), которые реагируют на более сложные стимулирующие воздействия и, по-видимому, образуются путем объединения и обработки выходных сигналов простых клеток. Эти реакции демонстрируют некоторую степень инвариантности к небольшим изменениям входного сигнала, таким как смещение местоположения, что аналогично объединению элементов в глубокой сверточной сети. Более глубокие уровни системы обработки зрительных сигналов млекопитающих имеют еще более специфические ответные реакции и еще большую инвариантность по отношению к преобразованиям зрительного сигнала. Такие клетки были названы «клетками бабушки» (*grandmother cells*), поскольку такая клетка может концептуально реагировать тогда и только тогда, когда визуальный сигнал соответствует изображению бабушки человека, независимо от местоположения, масштаба, освещения или других изменений сцены. Эта работа непосредственно вдохновила на создание ранней формы глубокой нейронной сети под названием «неокогнитрон» (*neocognitron*) (Fukushima, 1980), которая стала предтечей сверточных нейронных сетей. Неокогнитрон обладал несколькими слоями обработки сигналов, включающими локальные рецептивные поля с общими

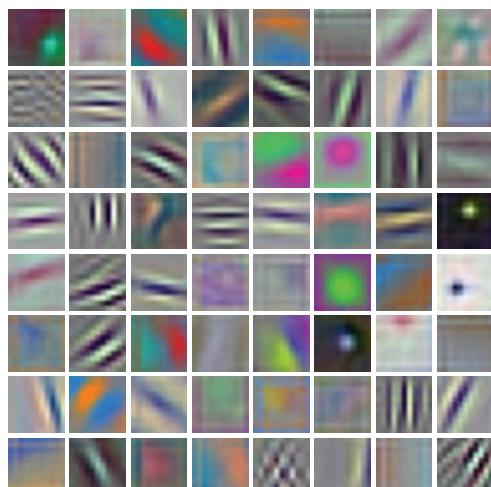
весами, за которыми следовало локальное усреднение или максимальный пулинг для передачи позиционной инвариантности. Однако в нем отсутствовала процедура сквозного обучения, поскольку он был создан до появления метода обратного распространения, и вместо этого он основывался на «жадном» алгоритме бесконтрольного кластеризованного послойного обучения.

10.3.2. Визуализация обученных фильтров

Пусть в наличии есть обученная глубокая CNN, и необходимо выяснить, что именно научились определять ее скрытые элементы. Для фильтров в первом сверточном слое это сделать относительно несложно, поскольку они соответствуют небольшим фрагментам в пространстве оригинального входного изображения. Поэтому можно визуализировать веса сети, связанные с этими фильтрами, непосредственно в виде небольших изображений. Первый сверточный слой выполняет вычисление внутренних произведений между фильтрами и соответствующими фрагментами изображения, и в этом случае данный элемент будет иметь большое значение активации при большом значении внутреннего произведения.

На рис. 10.12 показаны примеры фильтров из первого слоя CNN, обученной на наборе данных ImageNet. Можно заметить значительное сходство между этими фильтрами и фильтрами Габора на рис. 10.11, которые соответствуют признакам, распознаваемым живыми нейронами в зрительной коре млекопитающих. Однако это не означает, что сверточная нейронная сеть является удачной моделью работы головного мозга, поскольку очень похожие результаты могут быть получены с помощью самых разных статистических методов (Hyvärinen, Hurri and Hoyer, 2009). Это объясняется тем, что такие характерные фильтры являются общим свойством статистики естественных изображений и поэтому оказываются полезными для анализа изображений как в естественных, так и в виртуальных условиях.

РИС. 10.12 Примеры обученных фильтров из первого слоя AlexNet



Несмотря на возможность непосредственной визуализации фильтров первого слоя, последующие слои сети интерпретировать сложнее, поскольку их входом являются не фрагменты изображений, а группы ответных реакций фильтров. Один из подходов, аналогичный использованному в работе (Hubel and Wiesel, 1959), заключается в предоставлении сети большого количества фрагментов изображений и последующем анализе тех из них, которые приводят к наибольшему значению активации в определенном скрытом элементе. На рис. 10.13 показаны примеры, полученные с помощью сети с пятью сверточными слоями, за которыми следуют два полно связанных слоя, обученных на 1,3 млн точек данных ImageNet с охватом 1000 классов. Здесь прослеживается естественная иерархическая структура: первый слой реагирует на контуры, второй – на текстуры и простые формы, третий – на компоненты объектов (например, колеса), а пятый – на объекты в целом. Девять лучших активаций в каждой карте признаков расположены в виде сетки 3×3 для четырех случайно выбранных каналов в каждом из соответствующих слоев. По мере увеличения глубины наблюдается устойчивая прогрессия сложности: от простых кромок в слое 1 до полноценных объектов в слое 5.



РИС. 10.13 Примеры фрагментов изображений из проверочного набора, которые вызывают наибольшую активацию в скрытых элементах сети с пятью сверточными слоями, обученной на данных ImageNet. [Из (Zeiler and Fergus, 2013) с разрешения авторов]

Эту методику можно расширить, не ограничиваясь простой выборкой фрагментов изображений из валидационного набора, а выполнив численную оптимизацию входных переменных с целью максимизации активации конкретного элемента (Zeiler and Fergus, 2013; Simonyan, Vedaldi and Zisserman, 2013; Yosinski et al., 2015). Если выбрать один из элементов в качестве одного из выходов, можно попытаться найти изображение, наиболее соответствующее метке определенного класса. Поскольку выходные элементы обычно работают с функцией активации softmax, эффективнее максимизировать значение предварительной активации, которое поступает в softmax, а не напрямую вероятность класса, так как в этом случае оптимизация будет зависеть только от одного класса. Например, если искать изображение с наибольшей реакцией на класс «собака», то при оптимизации выхода softmax можно добиться того, что изображение будет, например, менее похоже на кошку из-за знаменателя в softmax. Такой подход близок к принципу состязательного обучения (см. главу 17). Однако неограниченная оптимизация ак-

тивации выходного элемента приводит к тому, что отдельные значения пикселей устремляются к бесконечности, а также к появлению высокочастотной структуры, которую сложно интерпретировать. По этой причине для поиска решений, более близких к естественным изображениям, требуется некоторая регуляризация. В работе (Yosinski et al., 2015) использовалась функция регуляризации, состоящая из суммы квадратов значений пикселей, а также процедура поочередного обновления значений пикселей изображения на основе градиента с операцией размытия для удаления высокочастотной структуры и операцией обрезки, которая обнуляет те значения пикселей, которые вносят лишь небольшой вклад в метку класса. Примеры результатов показаны на рис. 10.14. Для каждого из четырех классов объектов показаны четыре различных решения, полученные при различных настройках параметров регуляризации. [Из (Yosinski et al., 2015) с разрешения авторов.]

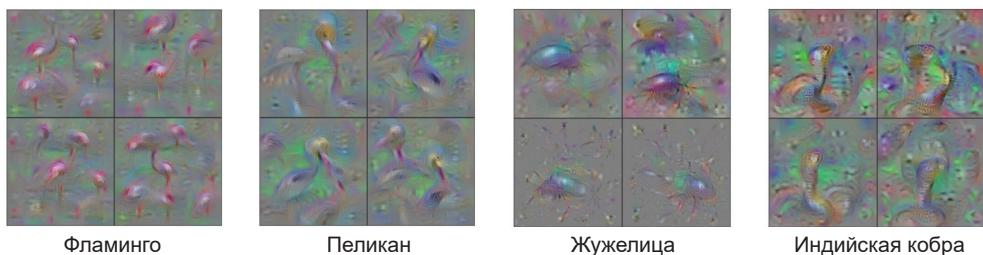


РИС. 10.14 Примеры синтетических изображений, полученных путем максимизации вероятности класса по отношению к значениям каналов пикселей изображения для обученного сверточного классификатора

10.3.3. Карты значимости

Еще одним способом получения информации о функциях, задействованных в сверточной сети, являются *карты значимости* (*saliency maps*), с помощью которых можно идентифицировать те области изображения, которые являются наиболее значимыми для определения метки класса. Для этого лучше всего исследовать последний сверточный слой, поскольку в нем все еще сохраняется пространственная локализация, которая теряется в последующих полносвязных слоях, и при этом он обладает самым высоким уровнем семантического представления. С помощью метода Grad-CAM (Gradient Class Activation Mapping) (Selvaraju et al., 2016) для определенного входного изображения сначала вычисляются производные предварительной активации выходного элемента $a^{(c)}$ определенного класса с еще до применения функции softmax по отношению к предварительным активациям $a_{ij}^{(k)}$ всех элементов конечного сверточного слоя для канала k . После этого по каждому каналу в этом слое вычисляется среднее значение этих производных, что дает

$$\alpha_k = \frac{1}{M_k} \sum_i \sum_j \frac{\partial a^{(c)}}{\partial a_{ij}^{(k)}}, \quad (10.9)$$

где i и j обозначают строки и столбцы канала k , а M_k – это общее количество элементов в этом канале. Далее эти средние значения используются для формирования взвешенной суммы в виде

$$\mathbf{L} = \sum_k \alpha_k \mathbf{A}^{(k)}, \quad (10.10)$$

где $\mathbf{A}^{(k)}$ – это матрица с элементами $a_{ij}^{(k)}$. Полученный массив имеет ту же размерность, что и конечный сверточный слой, например 14×14 для сети VGG, показанной на рис. 10.10, и может быть наложен на исходное изображение в виде «тепловой карты», как показано на рис. 10.15.

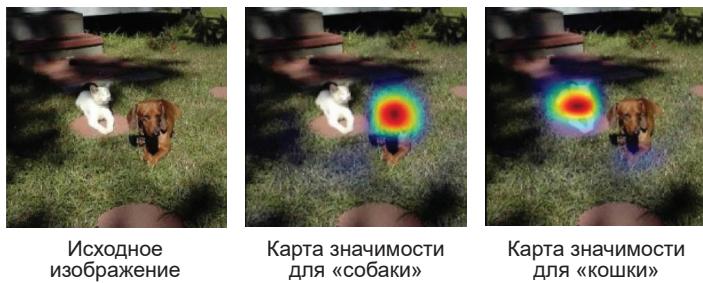


РИС. 10.15 Карты значимости сети VGG-16 для категорий «собака» и «кошка». [Из (Selvaraju et al., 2016) с разрешения авторов]

10.3.4. Состязательные атаки

Градиенты изменения значений пикселей входного изображения также могут быть использованы для создания так называемых *состязательных атак* (*adversarial attacks*) на сверточные сети (Szegedy et al., 2013). Такие атаки подразумевают внесение очень небольших изменений в изображение на мало-заметном для человека уровне, которые приводят к ошибочной классификации изображения нейронной сетью. Один из наиболее простых способов создания искаженных изображений называется *знакомым методом быстрого градиента* (*fast gradient sign method*) (Goodfellow, Shlens and Szegedy, 2014). Он заключается в изменении значения каждого пикселя в изображении \mathbf{x} на фиксированную величину со знаком, определяемым градиентом функции ошибки $E(\mathbf{x}, t)$ в зависимости от значений пикселей. В результате получается модифицированное изображение, определяемое как

$$\mathbf{x}' = \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} E(\mathbf{x}, t)). \quad (10.11)$$

Здесь t соответствует истинной метке \mathbf{x} , а ошибка $E(\mathbf{x}, t)$ может быть, например, отрицательным логарифмом правдоподобия \mathbf{x} . Требуемый градиент можно эффективно вычислить с помощью метода обратного распространения (см. главу 8). При обычном обучении нейронной сети параметры сети настраиваются так, чтобы минимизировать эту ошибку, в то время как мо-

дификация, определяемая в (10.11), изменяет изображение (при сохранении фиксированных параметров обученной сети) так, чтобы увеличить ошибку. Сохраняя небольшую погрешность, можно добиться того, что изменения изображения будут неразличимы для человеческого глаза. Примечательно, что в результате этого воздействия можно получить изображения с высокой степенью достоверности, которые сеть классифицирует неправильно, как показано в примере на рис. 10.16. Изображение слева классифицировано как панда с вероятностью 57,7 %. Добавление небольшого уровня случайного возмущения (которое само по себе классифицируется как нематода с вероятностью 8,2 %) приводит к изображению справа, которое классифицируется как гибbon с достоверностью 99,3 %. [Из (Goodfellow, Shlens and Szegedy, 2014) с разрешения авторов.]

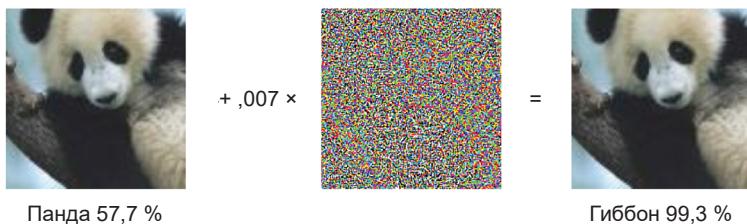


РИС. 10.16 Пример состязательной атаки на обученную сверточную сеть

Возможность обмана нейронных сетей подобным образом вызывает потенциальные проблемы с безопасностью, поскольку создает возможности для атак на обученные классификаторы. Может показаться, что эта проблема возникает из-за чрезмерной подгонки, когда сверхмощная модель точно адаптируется к конкретному изображению настолько, что небольшие изменения входных данных приводят к большим изменениям в прогнозируемых вероятностях классов. Однако выяснилось, что изображение, которое было адаптировано для получения ложного результата для конкретной обученной сети, может давать такие же ложные результаты при подаче в другие сети (Goodfellow, Shlens and Szegedy, 2014). Более того, аналогичный неблагоприятный результат может быть получен с помощью гораздо менее гибких линейных моделей. Возможно даже создание таких физических артефактов, что обычное, неиспорченное изображение артефакта будет давать ошибочные предсказания при подаче на обученную нейронную сеть, как показано на рис. 10.17. Изображения этих объектов на снимках уверенно классифицируются CNN как знаки ограничения скорости до 45 миль/ч. Несмотря на то что с основными видами подобных атак можно справиться путем внесения простых изменений в процесс обучения сети, с более сложными способами бороться намного сложнее. Понимание последствий этих результатов и смягчение их потенциальных проблем по-прежнему представляют собой открытые области для исследований.

РИС. 10.17 Два примера измененных физических знаков остановки. [Из (Eykholt et al., 2018) с разрешения авторов]



10.3.5. Синтетические изображения

В качестве последнего примера модификации изображения, позволяющего получить дополнительное представление о принципах работы обученной сверточной сети, рассмотрим технику под названием *DeepDream* (Mordvintsev, Olah and Tyka, 2015). Ее задача заключается в создании синтетического изображения с утрированными характеристиками. Для этого необходимо определить, какие узлы в конкретном скрытом слое сети сильно реагируют на определенное изображение, а затем изменить изображение для усиления подобных реакций. Например, если представить сеть, обученной распознаванию объектов, изображение облаков, и при этом какой-либо узел обнаружит в определенной области изображения паттерн, напоминающий кошку, то можно изменить изображение таким образом, чтобы оно стало еще более «похожим на кошку» в этой области. Для этого на вход сети подается изображение, которое затем распространяется по определенному слою. Затем для этого слоя задаются переменные δ обратного распространения (см. раздел 8.1.2), равные предварительным активациям узлов, и выполняется обратное распространение до входного слоя, в результате чего получается вектор градиента по пикселям изображения. Наконец, нужно изменить изображение, сделав небольшой шаг в направлении вектора градиента. Эту процедуру можно рассматривать как градиентный метод (см. упражнение 10.10) для увеличения функции:

$$F(\mathbf{I}) = \sum_{i,j,k} a_{ijk}(\mathbf{I})^2, \quad (10.12)$$

где $a_{ijk}(\mathbf{I})$ – это предварительная активация блока в строке i и столбце j канала k в выбранном слое, когда входным изображением является \mathbf{I} , а сумма складывается из всех блоков и всех каналов в этом слое. Для получения плавных изображений применяется некоторая регуляризация в виде пространственного сглаживания и отсечения пикселей. Этот процесс можно повторить несколько раз для достижения более выраженного эффекта. Примеры результирующего изображения показаны на рис. 10.18. В верхнем ряду показаны результаты применения алгоритма с использованием активаций из 7-го сверточного слоя сети VGG-16 после пяти итераций и после 30 итераций

ций. В нижнем ряду также показаны примеры с использованием 10-го слоя после пяти итераций и после 30 итераций. Примечательно, что, несмотря на обученность сверточных сетей различать классы объектов, они, похоже, способны улавливать как минимум часть информации, необходимой для создания изображений из этих классов.

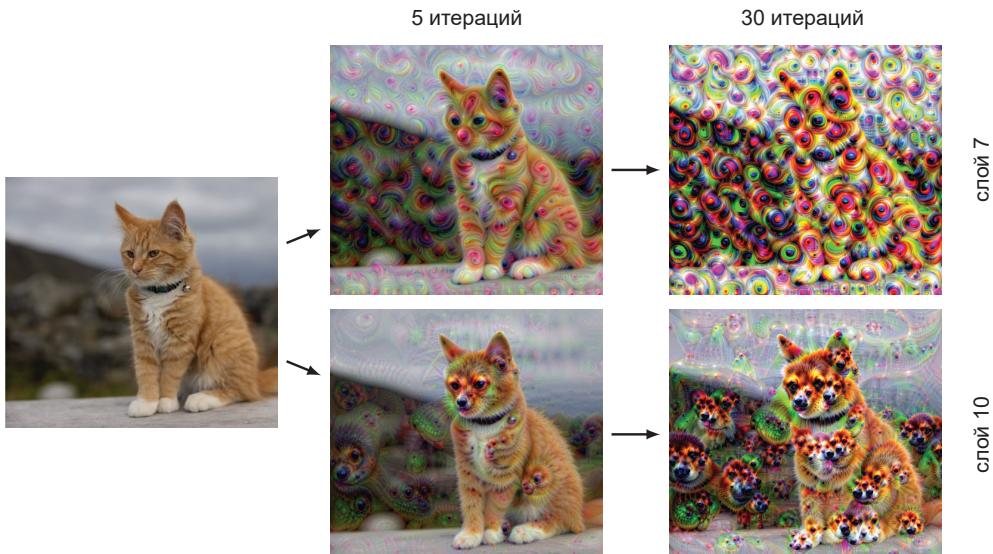


РИС. 10.18 Примеры применения DeepDream к изображению

Эту технику можно применить к фотографии, а можно начать с входных данных, состоящих из случайного шума, чтобы получить изображение, полностью генерированное обученной сетью. Несмотря на то что DeepDream позволяет получить некоторое представление о работе обученной сети, он в основном используется для создания эффектных изображений в виде произведений искусства.

10.4. Определение объектов

При разработке CNN в качестве мотивации прежде всего выступает задача классификации изображений, в которой все изображение относится к одному классу, например «кошка» или «велосипед». Это целесообразно для таких наборов данных, как ImageNet, в которых по своему замыслу каждое изображение характеризуется доминированием одного объекта. Однако существует множество других применений CNN, в которых можно использовать встроенные индуктивные смещения. В более общем случае сверточные слои CNN, обученные на большой базе данных изображений для конкретной задачи, могут научиться внутренним представлениям, которые имеют широкое

применение, и поэтому CNN можно точно настроить для решения широкого спектра конкретных задач (см. раздел 6.3.4). Ранее уже рассматривался пример сверточной сети, обученной на данных ImageNet, которая благодаря трансферному обучению смогла достичь эффективности классификации поражений кожи на уровне человека (см. раздел 1.1.1).

10.4.1. Ограничительные рамки

На многих изображениях есть несколько объектов, принадлежащих к одному или нескольким классам, и поэтому может возникнуть необходимость определить наличие и класс каждого объекта. Более того, во многих прикладных задачах компьютерного зрения требуется определить местоположение всех обнаруженных объектов на изображении. Например, автономному транспортному средству, использующему RGB-камеры, может потребоваться определить наличие и местоположение пешеходов, а также идентифицировать дорожные знаки, другие транспортные средства и т. д.

Рассмотрим проблему определения местоположения объекта на изображении. Широко распространенным подходом является определение *ограничительной рамки* (*bounding box*), которая представляет собой прямоугольник, плотно прилегающий к границам объекта, как показано на рис. 10.19. Здесь синие рамки соответствуют классу «автомобиль», красные – классу «пешеход», а оранжевые – классу «светофор». Границная область может быть задана координатами ее центра, а также шириной и высотой в виде вектора $\mathbf{b} = (b_x, b_y, b_w, b_h)$. Здесь элементы \mathbf{b} могут быть заданы в виде пикселей или непрерывных чисел, где, в соответствии с принятым соглашением, левый верхний угол изображения имеет координаты $(0, 0)$, а правый нижний – координаты $(1, 1)$.

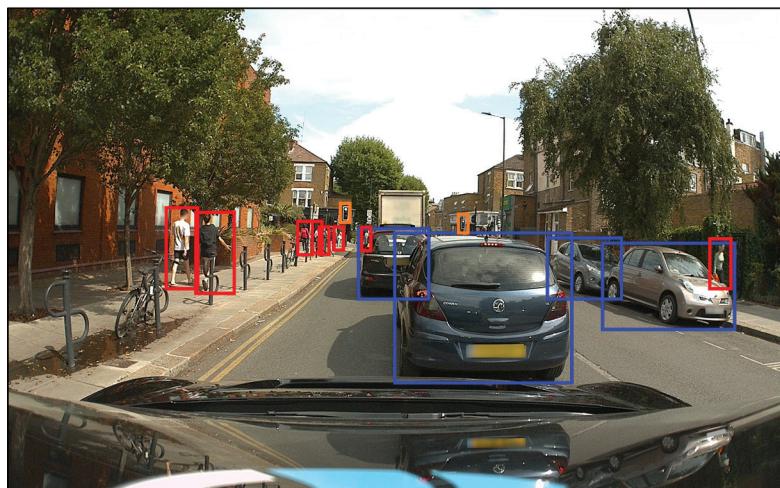


РИС. 10.19 Изображение с некоторыми объектами разных классов, в котором местоположение каждого объекта обозначено плотно прилегающим прямоугольником (ограничительной рамкой). [Оригинальное изображение предоставлено компанией Wayve Technologies Ltd.]

Если предполагается, что изображения содержат один и только один объект из заранее определенного набора классов C , то CNN обычно имеет C выходных блоков, функции активации которых определяются функцией softmax (см. раздел 5.3). Для локализации объекта можно использовать еще четыре выхода с линейными функциями активации, обученными прогнозированию координат ограничительной рамки (b_x, b_y, b_w, b_h). Поскольку эти величины непрерывны, может подойти функция ошибки суммы квадратов для соответствующих выходов. Так, например, было сделано в работе (Redmon et al., 2015), где изображение сначала делится на сетку 7×7 . Для каждой ячейки сетки используется сверточная сеть, которая выводит класс и координаты ограничительной рамки любого объекта, связанного с этой ячейкой сетки, на основе признаков, взятых из всего изображения.

10.4.2. Пересечение по объединению

Для определения эффективности обученной сети, которая может предсказывать ограничительные рамки, необходим эффективный способ. При классификации изображений результатом работы сети является распределение вероятностей по меткам классов, поэтому для оценки ее эффективности можно использовать логарифмическое правдоподобие для истинных меток классов на тестовом наборе. Тем не менее для локализации объектов нужен способ оценки точности ограничительной рамки относительно некоторой базовой истины, которая может быть получена, например, с помощью маркировки человеком. В качестве основы для такого измерения можно использовать степень перекрытия предсказанных и целевых границ, но при этом площадь перекрытия будет зависеть от размера объекта на изображении. Кроме того, предсказанная ограничительная рамка должна быть оштрафована за область предсказания, которая расположена за пределами истинной ограничительной рамки. Лучшая метрика, которая решает обе эти проблемы, называется «пересечение по объединению» (*intersection-over-union*, или *IoU*) и представляет собой простой коэффициент отношения площади пересечения двух ограничительных рамок к площади их объединения, как показано на рис. 10.20. Если предсказанная граница показана синим прямоугольником, а истинная – красным, то показатель IoU определяется как отношение площади пересечения рамок, показанных зеленым цветом слева, к площади их объединения, показанного зеленым цветом справа. Обратите внимание, что показатель IoU лежит в диапазоне от 0 до 1. Прогнозы могут быть признаны верными, если показатель IoU превышает пороговое значение, которое обычно задается на уровне 0,5. Обратите внимание, что IoU обычно не используется непосредственно в качестве функции потерь для обучения, поскольку ее трудно оптимизировать методом градиентного спуска. Поэтому обучение обычно проводится с использованием центрированных объектов, а показатель IoU используется преимущественно в качестве оценочной метрики.

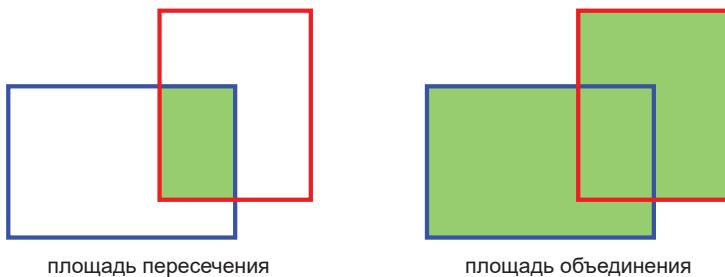


РИС. 10.20 Иллюстрация метрики «пересечение по объединению» для количественной оценки точности предсказания ограничительной рамки

10.4.3. Скользящие окна

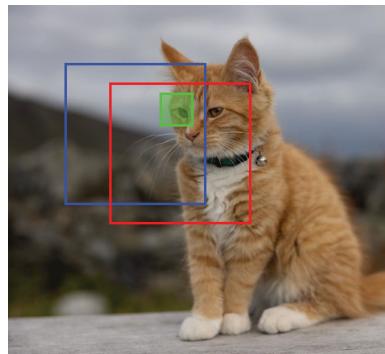
Один из подходов к обнаружению и локализации объектов заключается в создании обучающего набора, состоящего из строго отсеченных примеров объекта, который необходимо обнаружить, а также примеров аналогично отсеченных участков изображений, не содержащих никаких объектов (класс «фон»). Этот набор данных используется для обучения классификатора, например глубокой CNN, выходы которого определяют вероятность присутствия объекта каждого конкретного класса во входном окне. Обученная модель далее используется для выявления объектов на новом изображении путем «сканирования» входного окна по всему изображению и получения для каждой локации результирующего подмножества изображения в качестве входных данных для классификатора. Это называется методом *скользящего окна* (*sliding window*). В случае выявления объекта с высокой вероятностью связанное с ним местоположение окна определяет соответствующую ограничительную рамку.

Очевидный недостаток этого подхода заключается в значительных вычислительных издержках вследствие большого количества потенциальных положений окна на изображении. Кроме того, может возникнуть необходимость повторения процесса с использованием окон разного масштаба для учета различных размеров объектов на изображении. Экономия может быть достигнута за счет перемещения входного окна по горизонтали и вертикали со страйдом (шагом) более одного пикселя. Однако существует компромисс между точностью определения местоположения при использовании небольшого страйда и снижением вычислительных затрат при использовании большего страйда. Вычислительные затраты при использовании скользящего окна могут быть приемлемыми для простых классификаторов, но для глубоких нейронных сетей, потенциально содержащих миллионы параметров, стоимость такого наивного подхода может оказаться непомерно высокой.

К счастью, сверточная структура нейронной сети позволяет значительно повысить ее эффективность (Sermanet et al., 2013). Следует отметить, что сверточный слой в такой сети сам по себе предусматривает поэтапное скольжение определителя признаков с общими весами по входному изображению. Следовательно, когда скользящее окно используется для нескольких пря-

мых проходов через сверточную сеть, возникает значительная избыточность в вычислениях, как показано на рис. 10.21.

РИС. 10.21 Иллюстрация повторяющихся вычислений при использовании CNN для обработки данных со скользящим входным окном, где красная и синяя рамки показывают два перекрывающихся положения входного окна. Зеленая рамка обозначает одно из мест расположения рецептивного поля скрытого элемента в первом сверточном слое, и оценка активации соответствующего скрытого элемента разделяется между двумя положениями окна



Поскольку вычислительная структура скользящих окон повторяет структуру сверточек, эффективная реализация скользящих окон в сверточной сети оказалась чрезвычайно простой. Рассмотрим упрощенную сверточную сеть на рис. 10.22, состоящую из сверточного слоя, за которым следует слой с максимальным пулингом, а затем полно связанный слой. Для простоты в каждом слое показан только один канал, но расширение до нескольких каналов не представляет сложности. Входное изображение для сети имеет размер 6×6 , фильтры в сверточном слое имеют размер 3×3 со страйдом 1, а слой максимального пулинга имеет неперекрывающиеся рецептивные поля размером 2×2 со страйдом 1. После этого следует полно связанный слой с одним выходным элементом. Обратите внимание, что этот последний слой также можно рассматривать как еще один сверточный слой с размером фильтра 2×2 , так что для фильтра существует только одна позиция и, следовательно, один выход.

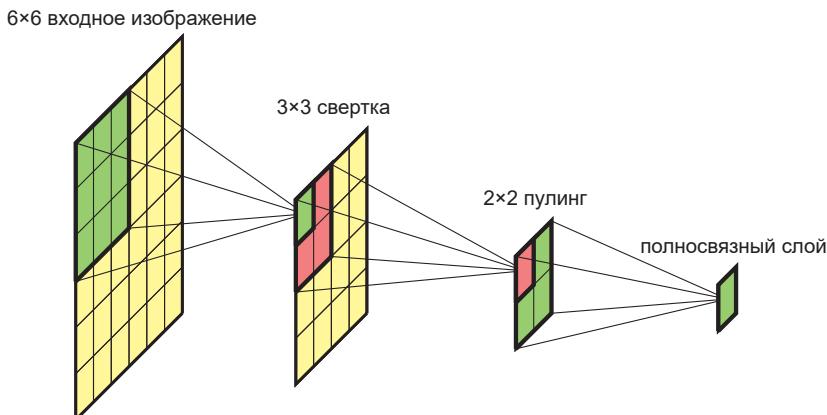


РИС. 10.22 Пример простой сверточной сети с одним каналом в каждом слое, которая служит иллюстрацией концепции скользящего окна для обнаружения объектов на изображениях

Теперь предположим, что эта сеть обучена на центрированных изображениях объектов, а затем используется для работы с более крупным изображением размером 8×8 , как показано на рис. 10.23, где сеть просто расширена за счет увеличения размеров сверточного слоя и слоя максимального пулинга. Слой свертки теперь имеет размер 6×6 , а слой объединения – 3×3 . Теперь есть четыре выходных блока, каждый из них со своей собственной функцией softmax. Веса в этом блоке распределяются между четырьмя блоками. Здесь видно, что вычисления, необходимые для обработки входного сигнала, соответствующего позиции окна в левом верхнем углу входного изображения, аналогичны тем, которые использовались для обработки исходных входных сигналов 6×6 , использованных при обучении. Для остальных позиций окна требуется лишь небольшой объем дополнительных вычислений, как показано синими квадратами, что приводит к значительному повышению эффективности по сравнению с наивным повторным применением полной сверточной сети (см. упражнение 10.12). Обратите внимание, что сами полностью связанные слои теперь имеют сверточную структуру.

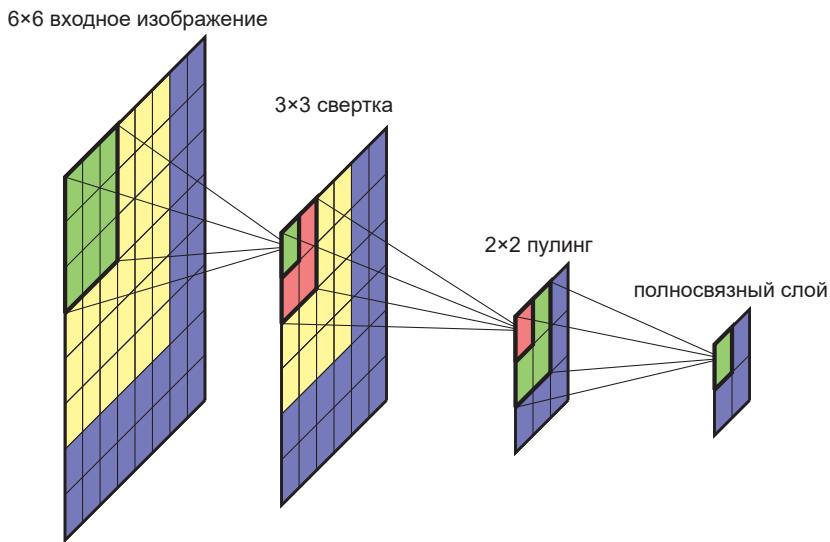


РИС. 10.23 Применение сети, показанной на рис. 10.22, к более крупному изображению, где требуемые дополнительные вычисления соответствуют синим областям

10.4.4. Обнаружение в разных масштабах

Помимо поиска объектов в различных положениях, также необходим поиск объектов разного масштаба и с разным соотношением сторон. Например, узкая ограничительная рамка, нарисованная вокруг кошки, будет иметь разное соотношение сторон в случае, когда кошка сидит в вертикальном положении, и в случае, когда она лежит. Вместо использования нескольких детекторов с различными размерами и формами входного окна более прос-

тым, но в то же время эквивалентным способом является использование фиксированного входного окна и создание нескольких копий входного изображения, каждая с различными парами коэффициентов масштабирования по горизонтали и вертикали. Затем входное окно сканируется по каждой из копий изображения для обнаружения объектов, а соответствующие коэффициенты масштабирования используются для преобразования координат ограничительной рамки обратно в пространство исходного изображения, как показано на рис. 10.24. Здесь исходное изображение (а) копируется несколько раз, и каждая копия масштабируется в горизонтальном и/или вертикальном направлениях, как показано для горизонтального масштабирования в (б). Затем окно фиксированного размера сканируется поверх масштабированных изображений. Если объект обнаружен с высокой вероятностью, как показано красной рамкой в (б), соответствующие координаты окна могут быть спроецированы обратно в пространство исходного изображения для определения соответствующей ограничительной рамки, как показано в (с).

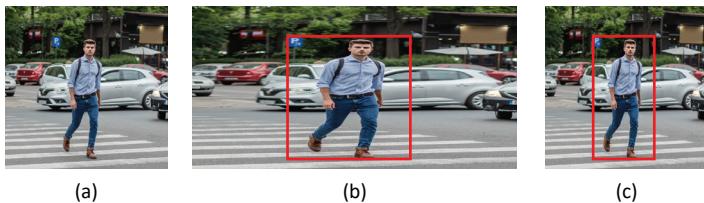


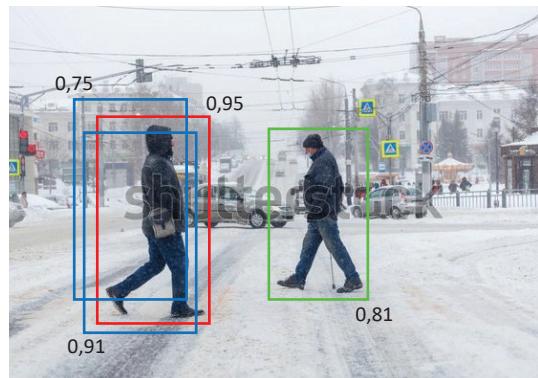
РИС. 10.24 Иллюстрация процесса обнаружения и локализации объектов в различных масштабах и соотношениях сторон с использованием фиксированного входного окна

10.4.5. Немаксимальное подавление

При сканировании изображения обученной сверточной сетью можно обнаружить на нем несколько экземпляров объектов одного класса, а также объекты других классов. При этом также возникает тенденция к многократному обнаружению одного и того же объекта в схожих локациях, как показано на рис. 10.25. Эту проблему можно решить с помощью *немаксимального подавления* (*non-max suppression*), при котором для каждого класса объектов по очереди выполняется следующая операция. Сначала скользящее окно проходит по всему изображению и оценивает вероятность присутствия объекта данного класса в каждой локации. Затем удаляются все связанные с этими объектами ограничительные рамки, вероятность которых ниже некоторого порога, например 0,7, что приводит к результату, показанному на рис. 10.25. Здесь красная ограничительная рамка соответствует наибольшей общей вероятности. В результате немаксимального подавления устраняются другие перекрывающиеся границы-кандидаты, показанные синим цветом, но при этом сохраняется выявление еще одного экземпляра того же класса объектов, показанного зеленой рамкой. Рамка с наибольшей вероятностью считается успешным обнаружением, а соответствующая ограничительная рамка запи-

сыается в качестве прогноза. Далее отбрасываются все остальные рамки, чей IoU с успешной рамкой распознавания превышает некоторый порог, например 0,5. Так можно исключить несколько близлежащих распознаваний одного и того же объекта. Затем из оставшихся рамок выбирается та, которая с наибольшей вероятностью является еще одним успешным распознаванием, и этап исключения повторяется. Процесс продолжается до тех пор, пока все ограничительные рамки не будут либо отброшены, либо объявлены успешными обнаружениями.

РИС. 10.25 Схематическое изображение нескольких случаев распознавания одного и того же объекта в соседних локациях с указанием соответствующих вероятностей



10.4.6. Быстрая региональная CNN

Еще одним способом повышения скорости распознавания и локализации объектов является использование подхода сканирующего окна, при котором все ресурсы глубокой сверточной сети применяются ко всем областям изображения, даже если в некоторых из них вероятность найти объект минимальна. Вместо этого можно задействовать какую-нибудь технику с меньшими вычислительными затратами, например алгоритм сегментации, для выявления участков изображения с более высокой вероятностью нахождения объекта и затем задействовать все возможности сети только для таких участков. Это привело к появлению таких методик, как *быстрое предложение регионов* (*fast region proposal*) с помощью CNN, или *быстрая региональная CNN* (*fast R-CNN*) (Girshick, 2015). Кроме того, сверточную сеть с предложением регионов можно использовать для определения наиболее перспективных регионов, что привело к созданию более быстрой R-CNN (*faster R-CNN*) (Ren et al., 2015), которая допускает сквозное обучение как самой сети с предложением регионов, так и сети для обнаружения и локализации.

Недостатком подхода со скользящим окном является необходимость учитывать большое количество положений окна с точным интервалом, что приводит к большим вычислительным нагрузкам. Более эффективным подходом является сочетание скользящих окон с прямым прогнозированием ограничительных рамок, о котором шла речь в начале этого раздела (Sermanet et al., 2013). В этом случае непрерывные выходы прогнозируют положение

ограничительной рамки относительно положения окна и, таким образом, обеспечивают возможность точной настройки прогнозируемого положения.

10.5. Сегментация изображений

В задаче классификации изображений метка одного класса присваивается всему изображению. Ранее было показано, что более подробную информацию дает распознавание нескольких объектов и регистрация их положения с помощью ограничительных рамок (см. раздел 10.4). Еще более детальный анализ получается при *семантической сегментации* (*semantic segmentation*), когда каждому пикселю изображения присваивается один из заранее определенных классов. Это означает, что выходное пространство будет иметь ту же размерность, что и входное изображение, и поэтому его удобнее представить в виде изображения с тем же количеством пикселей. Хотя входное изображение обычно имеет три канала для R, G и B, выходной массив будет иметь C каналов в случае существования C классов, отражающих вероятность для каждого класса. Если каждому классу соответствует свой (произвольно выбранный) цвет, то прогноз сети сегментации можно представить в виде изображения, в котором каждый пиксель окрашен в соответствии с классом, имеющим наибольшую вероятность, как показано на рис. 10.26. Например, синие пиксели соответствуют классу «автомобиль», красные – классу «пешеход», а оранжевые – классу «светофор».



РИС. 10.26 Пример изображения и соответствующей ему семантической сегментации, где каждый пиксель окрашен в соответствии с его классом. [Предоставлено компанией Wayve Technologies Ltd.]

10.5.1. Сверточная сегментация

Простым способом решения проблемы семантической сегментации является построение сверточной сети, на вход которой подается прямоугольный участок изображения по центру пикселя, а единственным выходом является функция softmax, которая классифицирует этот пиксель. Применяя такую сеть к каждому пикселю поочередно, можно сегментировать все изображе-

ние (для этого потребуется заполнение границ вокруг изображения в зависимости от размера входного окна). Однако такой подход будет крайне неэффективным из-за избыточности вычислений (рис. 10.21), вызванной перекрывающимися фрагментами. Как уже было показано, устранить такую неэффективность можно за счет объединения вычислений прямого прохода для разных мест входа в одну сеть, что приводит к модели, в которой конечные полносвязные слои также являются сверточными (рис. 10.23). В результате можно создать CNN, в которой каждый слой имеет ту же размерность, что и входное изображение, благодаря наличию страйда 1 в каждом слое с одинаковым заполнением и без пулинга. Каждый выходной элемент имеет функцию активации softmax с весами, общими для всех выходов. Несмотря на работоспособность такой сети, для обучения сложным внутренним представлениям, необходимым для достижения высокой точности, потребуется множество слоев с несколькими каналами в каждом слое, и в целом это будет непомерно затратно для изображений с приемлемым разрешением.

10.5.2. Повышающая дискретизация

Как уже было замечено, большинство сверточных сетей работают с несколькими уровнями поникающей дискретизации, так что с увеличением числа каналов размер карт признаков уменьшается, в результате чего общий размер и общая стоимость сети остаются на приемлемом уровне, а сама сеть может извлекать из изображения семантически значимые признаки высокого порядка. На основе этой концепции можно создать более эффективную архитектуру для семантической сегментации, взяв стандартную глубокую сверточную сеть и добавив дополнительные обучаемые слои, которые получают низкоразмерное внутреннее представление и преобразуют его обратно вплоть до исходного разрешения изображения (Long, Shelhamer and Darrell, 2014; Noh, Hong and Han, 2015; Badrinarayanan, Kendall and Cipolla, 2015), как показано на рис. 10.27. Здесь показано уменьшение размерности карт

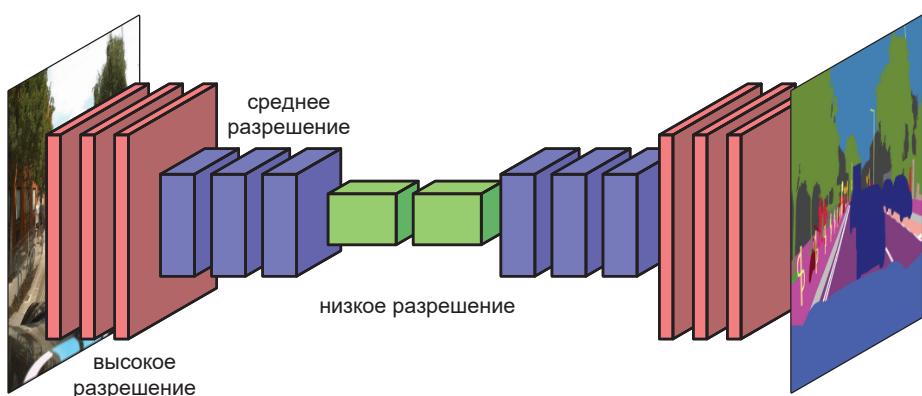


РИС. 10.27 Иллюстрация сверточной нейронной сети, используемой для семантической сегментации изображений

признаков с помощью серии сверток со страйдами и/или операций пулинга, за которыми следует серия сверток с переносами и/или операций *отмены пулинга (unpooling)*, которые увеличивают размерность до уровня исходного изображения.

Для этого необходим способ обратить эффект понижающей дискретизации, возникающий при свертках со страйдами и операциях пулинга. Сначала рассмотрим *повышающий (up-sampling)* аналог пулинга, когда выходной слой имеет большее число элементов, чем входной; например, каждому входному элементу соответствует блок выходных элементов 2×2 . Вопрос в том, какие значения использовать для выходов. Чтобы найти повышающий аналог среднего пулинга, достаточно скопировать каждое входное значение во все соответствующие выходные блоки, как показано на рис. 10.28а. Как видно, применение среднего пулинга к выходу этой операции приводит к регенерации входа.

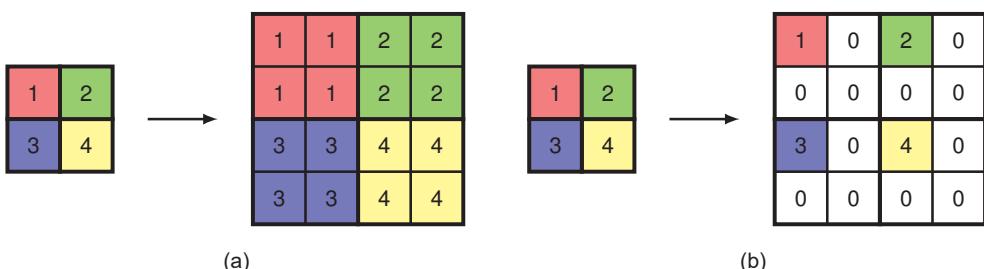


РИС. 10.28 Иллюстрация операций отмены пулинга: (a) аналог среднего пулинга и (b) аналог максимального пулинга

Для максимального пулинга можно рассмотреть операцию, показанную на рис. 10.28б, где каждое входное значение копируется в первый блок соответствующего выходного блока, а остальные значения в каждом блоке устанавливаются в ноль. И вновь получается, что применение операции максимального пулинга к выходному слою регенерирует входной слой. Иногда это также называют *максимальной отменой пулинга (max-unpooling)*. Присвоение ненулевого значения первому элементу выходного блока представляется произвольным, поэтому можно использовать модифицированный подход, который также сохраняет больше пространственной информации от слоев понижающей дискретизации (Badrinarayanan, Kendall and Cipolla, 2015). Для этого выбирается архитектура сети, в которой каждый слой понижающей дискретизации с максимальным пулингом имеет соответствующий слой повышающей дискретизации, расположенный дальше в сети. Тогда во время понижающей дискретизации ведется запись о присвоении элементу в каждом блоке максимального значения, а затем в соответствующем повышающем слое дискретизации ненулевой элемент выбирается для того же места, как показано на рис. 10.29 для максимального пулинга 2×2 .

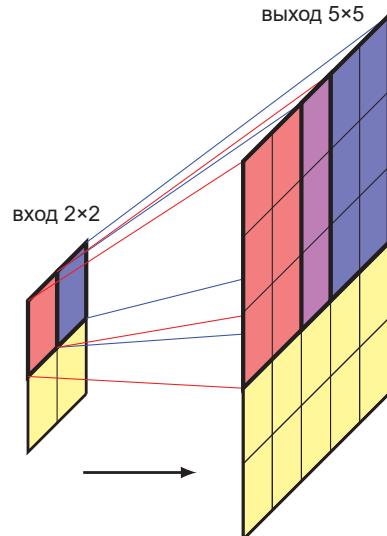


РИС. 10.29 Часть пространственной информации из слоя максимального пулинга (слева) можно сохранить, отметив положение максимального значения для каждого блока 2×2 во входном массиве, а затем в соответствующем слое повышающей дискретизации поместить ненулевую запись в соответствующее место выходного массива

10.5.3. Полностью сверточные сети

Рассмотренные выше методы повышающей дискретизации являются фиксированными функциями, как и операции поникающей дискретизации со средним и максимальным пулингом. Также можно использовать обученную повышающую дискретизацию, которая является аналогом свертки с использованием страйдов для поникающей дискретизации. При свертке со страйдом каждый блок на выходной карте связан через общие обучаемые веса с небольшим участком на входной карте, и при перемещении на один шаг по выходному массиву фильтр перемещается на два или более шагов по входному массиву. По этой причине выходной массив имеет меньшую размерность, чем входной. Для повышающей дискретизации используется фильтр, который соединяет один пиксель во входном массиве с участком выходного массива, а затем выбирается такая архитектура, чтобы при перемещении на один шаг по входному массиву фильтр перемещался на два или более шагов по выходному массиву (Dumoulin and Visin, 2016). Это показано на рис. 10.30

РИС. 10.30 Иллюстрация транспонирующей свертки для фильтра 3×3 с выходным страйдом 2. Ее можно рассматривать как обратную операцию к свертке 3×3 . Красный фрагмент на выходе получается умножением ядра на активацию красного блока во входном слое и аналогичным образом для синего фрагмента на выходе



для фильтров 3×3 и выходного страйда 2. Обратите внимание, что на выходе есть ячейки, для которых несколько позиций фильтра перекрываются, и соответствующие выходные значения можно найти либо путем суммирования, либо путем усреднения вкладов от отдельных позиций фильтра.

Такое повышение дискретизации называется *транспонирующей сверткой* (*transpose convolution*), ведь если свертка с понижением дискретизации выражена в матричной форме, то соответствующая свертка с повышением дискретизации задается транспонирующей матрицей (см. упражнение 10.13). Ее также называют *сверткой с дробным шагом (страйдом)* (*fractional strided convolution*), так как страйд стандартной свертки является отношением размера шага в выходном слое к размеру шага во входном слое. На рис. 10.30, например, это отношение равно 1/2. Обратите внимание, что иногда это также называют *деконволюцией*, или *обратной сверткой* (*deconvolution*), но лучше избегать этого термина, поскольку он широко используется в математике для обозначения операции, обратной операции свертки. Такая операция применяется в функциональном анализе, а это совсем другая концепция. Если в архитектуре сети отсутствуют объединяющие слои, а поникающая и повышающая дискретизация выполняется исключительно с помощью сверток, то такая архитектура называется *полностью сверточной сетью* (*fully convolutional network*) (Long, Shelhamer and Darrell, 2014). Она может принимать изображение произвольного размера и выдавать карту сегментации такого же размера.

10.5.4. Архитектура U-net

Как выяснилось, поникающая дискретизация за счет сверток со страйдами и пулинга позволяет увеличить число каналов без увеличения размера сети. Это также приводит к снижению пространственного разрешения и, следовательно, к потере пространственной информации по мере прохождения сигналов через сеть. Несмотря на то что для классификации изображений это вполне приемлемо, потеря пространственной информации представляет собой проблему для семантической сегментации, поскольку здесь необходимо классифицировать каждый пиксель. Одним из подходов к решению этой проблемы является архитектура *U-net* (Ronneberger, Fischer and Brox, 2015), показанная на рис. 10.31. Название этой архитектуры происходит от U-образной формы диаграммы.

Основная концепция заключается в том, что для каждого слоя поникающей выборки существует соответствующий слой повышающей выборки, и конечный набор активаций каналов в каждом слое поникающей выборки объединяется с соответствующим первым набором каналов в слое повышающей выборки, тем самым предоставляя этим слоям доступ к пространственной информации с более высоким разрешением. Обратите внимание, что в последнем слое *U-net* для уменьшения числа каналов до числа классов можно использовать свертки 1×1 , после чего следует функция активации *softmax*.

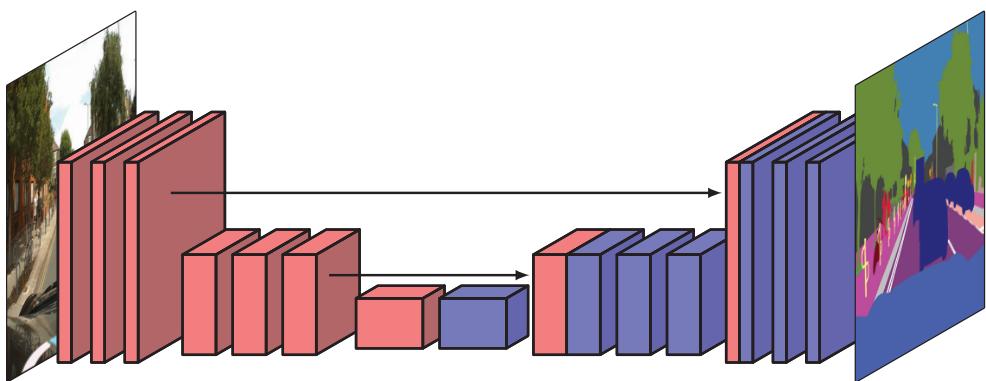


РИС. 10.31 Архитектура *U*-*net* имеет симметричное расположение слоев поникающей и повышающей дискретизации, а выход каждого слоя поникающей дискретизации соединяется в цепочку с соответствующим слоем повышающей дискретизации

10.6. Перенос стиля

Как уже было замечено, ранние слои глубокой сверточной сети учатся определять простые признаки, такие как кромки и текстуры, в то время как более поздние слои учатся определять более сложные объекты, такие как предметы (см. раздел 10.3). Это свойство можно использовать для перерисовки изображения в стиле другого изображения с помощью процесса, называемого *нейронным переносом стиля* (*neural style transfer*) (Gatys, Ecker and Bethge, 2015), как показано на рис. 10.3. Здесь представлена фотография сцены на канале (слева), которая была выполнена в стиле «Крушения транспортного судна» Дж. М. В. Тёрнера (в центре) и в стиле «Звездной ночи» Винсента Ван Гога (справа). В каждом случае изображение, использованное для создания стиля, показано на вставке.



РИС. 10.32 Пример нейронного переноса стиля. [Из (Gatys, Ecker and Bethge, 2015) с разрешения авторов]

Пусть целью является создание синтетического изображения G , «содержание» которого определяется изображением C , а «стиль» взят из другого изображения S . Это достигается путем определения функции ошибки $E(G)$,

заданной суммой двух членов, один из которых способствует тому, чтобы содержание (content) \mathbf{G} было похоже на \mathbf{C} , а другой – чтобы стиль (style) \mathbf{G} был похож на \mathbf{S} :

$$E(\mathbf{G}) = E_{\text{content}}(\mathbf{G}, \mathbf{C}) + E_{\text{style}}(\mathbf{G}, \mathbf{S}). \quad (10.13)$$

Функциональные формы этих двух терминов определяют концепции содержания и стиля в неявном виде. Затем можно найти \mathbf{G} , начав со случайно инициализированного изображения и используя градиентный спуск для минимизации $E(\mathbf{G})$.

Чтобы определить $E_{\text{content}}(\mathbf{G}, \mathbf{C})$, можно выбрать определенный сверточный слой в сети и измерить активацию блоков этого слоя, когда в качестве входного сигнала используется изображение \mathbf{G} , а также когда в качестве входного сигнала используется изображение \mathbf{C} . Затем нужно добиться того, чтобы соответствующие предварительные активации были одинаковыми, для чего используется функция ошибки суммы квадратов вида

$$E_{\text{content}}(\mathbf{G}, \mathbf{C}) = \sum_{i,j,k} \{a_{ijk}(\mathbf{G}) - a_{ijk}(\mathbf{C})\}^2, \quad (10.14)$$

где $a_{ijk}(\mathbf{G})$ обозначает предварительную активацию блока в позиции (i, j) в канале k данного слоя, когда входным изображением является \mathbf{G} , и аналогично для $a_{ijk}(\mathbf{C})$. Выбор того или иного слоя для определения предварительных активаций влияет на конечный результат: если ранние слои нацелены на поиск низкоуровневых признаков, таких как кромки, то более поздние слои – на поиск более сложных структур или даже целых объектов.

При определении $E_{\text{style}}(\mathbf{G}, \mathbf{C})$ подразумевается, что стиль определяется совпадением признаков из разных каналов в пределах сверточного слоя. Например, если стиль изображения \mathbf{S} таков, что вертикальные края обычно ассоциируются с оранжевыми пятнами, то желательно, чтобы то же самое было верно и для сгенерированного изображения \mathbf{G} . Тем не менее, хотя $E_{\text{content}}(\mathbf{G}, \mathbf{C})$ пытается сопоставить признаки \mathbf{G} в тех же местах, что и соответствующие признаки \mathbf{C} , для ошибки стиля $E_{\text{style}}(\mathbf{G}, \mathbf{S})$ необходимо, чтобы \mathbf{G} имел характеристики, совпадающие с характеристиками \mathbf{S} , но взятые из произвольного места, и поэтому используется среднее значение по точкам в карте признаков. Снова рассмотрим конкретный сверточный слой. Степень совпадения признака в канале k с соответствующим признаком в канале k' для входного изображения \mathbf{G} можно измерить путем формирования матрицы перекрестной корреляции:

$$F_{kk'}(\mathbf{G}) = \sum_{i=1}^I \sum_{j=1}^J a_{ijk}(\mathbf{G}) a_{ijk'}(\mathbf{C}), \quad (10.15)$$

где I и J – это размеры карт признаков в данном конкретном сверточном слое, а произведение $a_{ijk} a_{ijk'}$ будет значительным, если активированы оба признака. Если в этом слое имеется K каналов, то $F_{kk'}$ образуют элементы матрицы $K \times K$,

называемой *матрицей стилей* (*style matrix*). Степень совпадения стиля двух изображений \mathbf{G} и \mathbf{S} можно определить путем сравнения их матриц стилей с помощью функции:

$$E_{\text{style}}(\mathbf{G}, \mathbf{S}) = \frac{1}{(2IJK)^2} \sum_{k=1}^K \sum_{k'=1}^K \{F_{kk'}(\mathbf{G}) - F_{kk'}(\mathbf{S})\}^2. \quad (10.16)$$

Несмотря на то что можно было бы снова использовать один слой, более впечатляющие результаты получаются при использовании вкладов от нескольких слоев в виде

$$E_{\text{style}}(\mathbf{G}, \mathbf{S}) = \sum_l \lambda_l E_{\text{style}}^{(l)}(\mathbf{G}, \mathbf{S}), \quad (10.17)$$

где l обозначает сверточный слой. Коэффициенты λ_l определяют относительное весовое соотношение между различными слоями, а также весовое соотношение относительно члена ошибки контента. Эти весовые коэффициенты настраиваются эмпирически с помощью субъективной оценки.

Упражнения

- 10.1** (*) Пусть задан фиксированный весовой вектор w . Докажите, что входной вектор x , максимизирующий скалярное произведение $w^T x$, при условии, что $\|x\|^2$ постоянно, задается $x = \alpha w$ для некоторого скаляра α (см. приложение C). Это проще всего сделать с помощью множителя Лагранжа.
- 10.2** (**) Рассмотрим слой сверточной сети с одномерным входным массивом и одномерной картой признаков, как показано на рис. 10.2, где входной массив имеет размерность 5, а фильтры – ширину 3 со страйдом 1. Докажите, что это можно выразить как частный случай полносвязного слоя, записав матрицу весов, в которой отсутствующие связи заменяются нулями, а общие параметры обозначаются с помощью повторяющихся записей. Любые параметры смещения можно игнорировать.
- 10.3** (*) Вычислите в явном виде выход следующей свертки входной матрицы 4×4 с фильтром 2×2 :

2	5	-3	0
0	6	0	-4
-1	-3	0	2
5	0	0	3

*

-2	0
4	6

=

?	?	?
?	?	?
?	?	?

(10.18)

- 10.4** (**) Для изображения I с $J \times K$ пикселями и фильтра K с $L \times M$ элементами запишите пределы для двух сумм в (10.2). В математической лите-

туре операция (10.2) называется перекрестной корреляцией, а свертка определяется как

$$C(j, k) = \sum_l \sum_m I(j - l, k - m)K(l, m). \quad (10.19)$$

Запишите пределы для сумм в (10.19). Докажите, что (10.19) можно записать в эквивалентной «перевернутой» форме:

$$C(j, k) = \sum_l \sum_m I(j + l, k + m)K(l, m) \quad (10.20)$$

и снова запишите пределы для сумм.

- 10.5** (*) В математике свертка для непрерывной переменной x определяется как

$$F(x) = \int_{-\infty}^{\infty} G(y)k(x - y)dy, \quad (10.21)$$

где $k(x - y)$ – это ядерная функция. С помощью дискретной аппроксимации интеграла объясните связь со сверточным слоем, определяемым в (10.19), в CNN.

- 10.6** (*) Рассмотрим изображение размером $J \times K$, на которое со всех сторон наложено еще P пикселей и которое затем свернуто с помощью ядра размером $M \times M$, где M – это нечетное число. Докажите, что если выбрать $P = (M - 1)/2$, то результирующая карта признаков будет иметь размер $J \times K$ и, следовательно, будет иметь тот же размер, что и исходное изображение.
- 10.7** (*) Докажите, что если ядро размера $M \times M$ свернуто с изображением размера $J \times K$ с заполнением глубиной P и страйдами длины S , то размерность полученной карты признаков определяется по формуле (10.5).
- 10.8** (**) Для каждого из 16 слоев в CNN VGG-16, показанной на рис. 10.10, оцените (i) количество весов (т. е. связей), включая смещения, и (ii) количество независимо обучаемых параметров. Убедитесь, что общее число обучаемых параметров в сети составляет примерно 138 млн.
- 10.9** (**) Рассмотрим свертку вида (10.2) и предположим, что ядро является разделяемым, так что

$$K(l, m) = F(l)G(m) \quad (10.22)$$

для некоторых функций $F(\cdot)$ и $G(\cdot)$. Докажите, что вместо выполнения одной двумерной свертки теперь можно вычислить тот же самый результат с помощью двух одномерных сверток, что дает значительное повышение эффективности.

- 10.10** (*) Процедура обновления в DeepDream включает в себя установку переменных δ для обратного распространения равными предвари-

тельным активациям узлов в выбранном слое, а затем запуск обратного распространения на входной слой для получения вектора градиента по пикселям изображения. Докажите, что это можно вывести как градиентную оптимизацию относительно пикселей изображения I применительно к функции (10.12).

- 10.11** (**) При проектировании нейронной сети для распознавания объектов из C различных классов на изображении используется метка класса «1 из ($C + 1$)» с одной переменной для каждого класса объектов и одной дополнительной переменной, представляющей «фоновый» класс, т. е. область входного изображения, не содержащую объектов, принадлежащих ни к одному из определенных классов. Сеть выдает вектор вероятностей длины ($C + 1$). В качестве альтернативы для обозначения наличия или отсутствия объекта можно использовать одну двоичную переменную, а для обозначения конкретного класса объекта – отдельный вектор длины «1 из C ». В этом случае сеть выдает единственную вероятность, представляющую наличие объекта, и отдельный набор вероятностей для обозначения класса. Запишите взаимосвязь между этими двумя наборами вероятностей.
- 10.12** (**) Рассчитайте количество вычислительных шагов, необходимых для одного прямого прохода через сверточную сеть, показанную на рис. 10.22, без учета смещений и без учета оценки функций активации. Аналогичным образом рассчитайте общее количество вычислительных шагов для одного прямого прохода через расширенную сеть, показанную на рис. 10.23. Наконец, оцените соотношение девяти повторных наивных применений сети на рис. 10.22 к изображению 8×8 по сравнению с одним применением сети на рис. 10.23. Это соотношение указывает на повышение эффективности за счет использования сверточной реализации метода скользящего окна.
- 10.13** (**) В этом упражнении на примере одномерных векторов показано, почему сверточную повышающую дискретизацию иногда называют транспонирующей сверткой. Рассмотрим одномерный сверточный слой с активацией (x_1, x_2, x_3, x_4) , на вход которого поданы четыре блока, заполненные нулями, чтобы получить $(0, x_1, x_2, x_3, x_4, 0)$, и фильтр с параметрами (w_1, w_2, w_3) . Запишите одномерный вектор активации выходного слоя, предполагая, что страйд равен 2. Выразите этот выход в виде матрицы A , умноженной на вектор $(0, x_1, x_2, x_3, x_4, 0)$. Теперь рассмотрим свертку с повышающей дискретизацией, в которой входной слой имеет активации (z_1, z_2) с фильтром, имеющим значения (w_1, w_2, w_3) , и выходным страйдом 2. Запишите шестимерный выходной вектор, предполагая, что перекрывающиеся значения фильтра суммируются, а функция активации является просто тождеством. Докажите, что это можно выразить в виде матричного умножения с использованием транспонирующей матрицы A^T .

Глава 11

Структурированные распределения

Как уже было отмечено ранее, вероятность является одной из важнейших фундаментальных концепций глубокого обучения. Например, нейронная сеть, используемая для бинарной классификации, описывается условным распределением вероятности вида

$$p(t | \mathbf{x}, \mathbf{w}) = y(\mathbf{x}, \mathbf{w})^t \{1 - y(\mathbf{x}, \mathbf{w})\}^{(1-t)}, \quad (11.1)$$

где $y(\mathbf{x}, \mathbf{w})$ – это функция нейронной сети, которая принимает на вход вектор \mathbf{x} и управляемся вектором \mathbf{w} обучаемых параметров. Соответствующее значение правдоподобия перекрестной энтропии является основой для определения функции ошибки, используемой для обучения нейронной сети. Функция сети может быть очень сложной, но при этом условное распределение в (11.1) имеет простую форму. Однако существует множество важнейших моделей глубокого обучения, которые имеют гораздо более сложную вероятностную структуру, например большие языковые модели, нормализующие потоки, вариационные автокодировщики, диффузионные модели и многие другие. Для описания и использования этой структуры используется мощный фреймворк, который называется *вероятностными моделями графов* (*probabilistic graphical models*), или просто *моделями графов* (*graphical models*), которые обеспечивают возможность выражения структурированных вероятностных распределений в графической форме. Модели графов в сочетании с нейронными сетями для определения связанных распределений вероятностей обеспечивают исключительную гибкость при создании сложных моделей, которые могут быть обучены до конца с помощью стохастического градиентного спуска, с эффективной оценкой градиентов с помощью автодифференцирования. В этой главе речь пойдет об основных концепциях моделей графов для приложений глубокого обучения, в то время как более полное описание моделей графов для машинного обучения в целом можно найти в книге (Bishop, 2006).

11.1. Модели графов

Теория вероятностей может быть выражена в виде двух простых уравнений, известных как *правило суммы* и *правило произведения* (см. раздел 2.1). Все вероятностные операции, обсуждаемые в этой книге, независимо от их сложности сводятся к многократному применению этих двух уравнений. Теоретически можно было бы сформулировать и использовать сложные вероятностные модели исключительно посредством алгебраических действий. Однако в дальнейшем будет полезно дополнить этот анализ с помощью диаграммных представлений вероятностных распределений, поскольку они обладают рядом полезных свойств.

1. Они обеспечивают простой способ визуализации структуры вероятностной модели и могут быть использованы для разработки и объяснения новых моделей.
2. Свойства модели, в том числе свойства условной независимости, удобнее всего изучать на диаграммах.
3. Сложные вычисления, необходимые для выполнения выводов и обучения в сложных моделях, могут быть выражены в виде графических операций, таких как передача сообщений, где базовые математические операции проводятся в неявном виде.

Хотя такие модели графов имеют узлы и ребра, очень похожие на диаграммы нейронных сетей, их интерпретация носит специфически вероятностный характер и несет в себе более богатую семантику. Во избежание путаницы в этой книге диаграммы нейронных сетей обозначаются синим цветом, а вероятностные модели графов – красным.

11.1.1. Ориентированные графы

Граф состоит из узлов (*nodes*), также называемых *вершинами* (*vertices*), соединенных связями (*links*), также называемыми *ребрами* (*edges*). В вероятностной модели графа каждый узел представляет собой случайную переменную, а связи выражают вероятностные отношения между этими переменными. Таким образом, граф является способом отображения, с помощью которого совместное распределение по всем случайным переменным может быть разложено на произведение факторов, зависящих только от подмножества переменных. В этой главе основное внимание будет уделено моделям графов, в которых связи графов имеют определенное направление, обозначенное стрелками. Такие модели известны как *направленные модели графов* (*directed graphical model*). Их также называют *байесовскими сетями* (*Bayesian networks*) или *сетями Байеса* (*Bayes nets*).

Другим важным классом моделей графов являются *случайные поля Маркова*, или *марковские случайные поля* (*Markov random fields*), также известные как *неориентированные модели графов* (*undirected graphical models*), в которых связи не содержат стрелок и не обладают значением направленности. На-

правленные графы полезны для выражения причинно-следственных связей между случайными величинами, в то время как неориентированные графы лучше подходят для выражения мягких ограничений между случайными величинами. И направленные, и неориентированные графы можно рассматривать как частные случаи представления, называемого *фактор-графом* (*factor graph*). В дальнейшем основное внимание будет уделяться направленным графическим моделям. Впрочем, следует отметить, что неориентированные графы, не имеющие вероятностной интерпретации, также будут встречаться в процессе обсуждения графовых нейронных сетей (см. главу 13), где узлы представляют собой детерминированные переменные, как в стандартных нейронных сетях.

11.1.2. Факторизация

Для обоснования использования направленных графов при описании распределений вероятностей вначале рассмотрим произвольное совместное распределение $p(a, b, c)$ по трем переменным a, b и c . Обратите внимание, что на этом этапе совершенно не требуется что-либо уточнять об этих переменных – например, являются ли они дискретными или непрерывными. Действительно, один из мощных аспектов графовых моделей заключается в том, что с помощью определенного графа можно сформулировать вероятностные утверждения для широкого класса распределений. Применяя правило произведения вероятностей (2.9), можно записать совместное распределение в виде

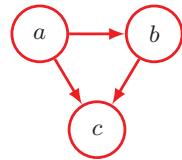
$$p(a, b, c) = p(c|a, b)p(a, b). \quad (11.2)$$

Второе применение правила произведения, на этот раз ко второму члену в правой части (11.2), дает

$$p(a, b, c) = p(c|a, b)p(b|a)p(a). \quad (11.3)$$

Следует отметить, что это разложение справедливо для любого выбора совместного распределения. Теперь представим правую часть (11.3) в виде простой модели графа следующим образом. Сначала для каждой из случайных величин a, b и c введем узел и свяжем каждый узел с соответствующим условным распределением в правой части (11.3). Затем для каждого условного распределения добавляем направленные связи (изображенные в виде стрелок) от узлов, соответствующих переменным, от которых зависит распределение. Так, для фактора $p(c|a, b)$ будут существовать связи от узлов a и b к узлу c , тогда как для фактора $p(a)$ входящих связей не будет. В результате получится граф, показанный на рис. 11.1. Если существует связь, идущая от узла a к узлу b , то говорят, что узел a является *родительским* (*parent*) по отношению к узлу b , а узел b является *дочерним* (*child*) по отношению к узлу a . Обратите внимание, что в дальнейшем не будет проводиться никакого формального различия между узлом и переменной, которой он соответствует, а для обозначения обоих будет использоваться один и тот же символ.

РИС. 11.1 Направленная модель графа, представляющая совместное распределение вероятностей по трем переменным a , b и c , соответствующая разложению в правой части (11.3)



Важно отметить, что левая часть (11.3) симметрична относительно трех переменных a , b и c , тогда как правая часть – нет. При выполнении разложения в (11.3) был неявно выбран определенный порядок, а именно a , b , c , и если бы был выбран другой порядок, то получилось бы другое разложение и, следовательно, другое представление графа.

Сейчас расширим пример рис. 11.1, рассмотрев совместное распределение по K переменным, заданное $p(x_1, \dots, x_K)$. Повторное применение правила произведения вероятностей позволяет записать это совместное распределение в виде произведения условных распределений, по одному для каждой из переменных:

$$p(x_1, \dots, x_K) = p(x_K | x_1, \dots, x_{K-1}) \dots p(x_2 | x_1)p(x_1). \quad (11.4)$$

Для заданного выбора K можно снова представить это в виде направленного графа с K вершинами, по одной для каждого условного распределения в правой части (11.4), при этом каждая вершина имеет входящие связи со всеми нижестоящими вершинами. Можно сказать, что этот граф является *полносвязным* (*fully connected*), поскольку между каждой парой узлов имеется связь.

До сих пор речь шла о совершенно общих совместных распределениях, поэтому их факторизация и соответствующее представление в виде полносвязных графов будут применимы к любому выбору распределения. Как вскоре будет показано, именно отсутствие связей в графе передает важную информацию о свойствах класса распределений, которые представляет этот график. Рассмотрим график, показанный на рис. 11.2. Обратите внимание, что это не полносвязный график, поскольку, например, в нем нет связей от x_1 к x_2 или от x_3 к x_7 . Возьмем этот график и извлечем соответствующее представление совместного распределения вероятностей, записанное в виде произведения множества условных распределений, по одному для каждого узла графа. Каждое такое условное распределение будет обусловлено только родителями соответствующего узла графа. Например, x_5 определяется по x_1 и x_3 . Таким образом, совместное распределение всех семи переменных имеет вид:

$$p(x_1)p(x_2)p(x_3)p(x_4 | x_1, x_2, x_3)p(x_5 | x_1, x_3)p(x_6 | x_4)p(x_7 | x_4, x_5). \quad (11.5)$$

Читателю стоит внимательно изучить соответствие между (11.5) и рис. 11.2.

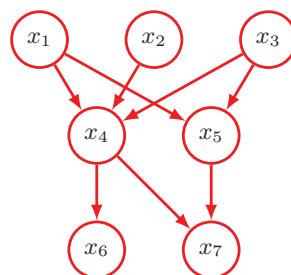
Теперь в общих чертах можно сформулировать связь между заданным направленным графиком и соответствующим распределением по переменным. Совместное распределение, определяемое графиком, дается произведением по всем узлам графа условного распределения для каждого узла, зависящего от

переменных, соответствующих родителям этого узла в графе. Таким образом, для графа с K узлами совместное распределение задается как

$$p(x_1, \dots, x_K) = \prod_{k=1}^K p(x_k | \text{pa}(k)), \quad (11.6)$$

где $\text{pa}(k)$ обозначает множество родителей x_k . Это ключевое уравнение выражает свойства *факторизации* (*factorization*) совместного распределения для модели направленного графа. Хотя каждый узел рассматривался здесь как соответствующий одной переменной, с узлами графа можно с равным успехом ассоциировать наборы переменных, а также переменные с векторными или тензорными значениями. Несложно доказать, что представление в правой части (11.6) всегда является правильно нормализованным, если нормализованы отдельные условные распределения (см. упражнение 11.1).

РИС. 11.2 Пример направленного графа, описывающего совместное распределение по переменным x_1, \dots, x_7 . Соответствующая декомпозиция совместного распределения дается в (11.5)



На рассматриваемые нами ориентированные графы накладывается важное ограничение: в них не должно быть *направленных циклов* (*directed cycles*). Другими словами, в графе нет замкнутых путей, по которым можно переходить от узла к узлу, следуя направлению стрелок, и в итоге вернуться в начальный узел. Такие графы также называются *направленными ациклическими графиками* (*directed acyclic graphs*, или *DAG*) (см. упражнение 11.2). Это эквивалентно утверждению, что существует такое упорядочивание узлов, при котором не существует связей, идущих от любого узла к любому узлу с меньшим номером.

11.1.3. Дискретные переменные

Ранее уже обсуждалась значимость распределений вероятностей, входящих в экспоненциальное семейство (см. раздел 3.4), где было отмечено, что это семейство включает в себя множество известных распределений в качестве частных случаев. Хотя такие распределения относительно просты, они образуют полезные строительные блоки для формирования более сложных распределений вероятностей, и рамки графовых моделей хорошо подходят для выражения взаимосвязи между этими строительными блоками. Существует два конкретных варианта широко используемых распределений компонен-

тов, соответствующих дискретным переменным и гауссовым переменным. Для начала рассмотрим дискретный случай.

Распределение вероятности $p(\mathbf{x}|\boldsymbol{\mu})$ для одной дискретной переменной \mathbf{x} , имеющей K возможных состояний (в представлении «1 из K »), определяется как

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^K \mu_k^{x_k} \quad (11.7)$$

и управляетя параметрами $\boldsymbol{\mu} = (\mu_1, \dots, \mu_K)^T$. Благодаря ограничению $\sum_k \mu_k = 1$ для определения распределения необходимо задать только $K - 1$ значений для μ_k .

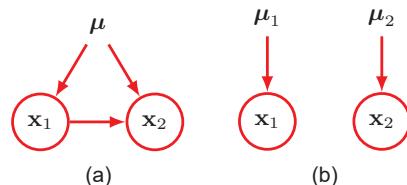
Пусть имеется две дискретные переменные \mathbf{x}_1 и \mathbf{x}_2 , каждая из которых имеет K состояний, и требуется смоделировать их совместное распределение. Обозначим вероятность наблюдения как $x_{1k} = 1$, так и $x_{2l} = 1$ параметром μ_{kl} , где x_{1k} обозначает k -ю компоненту \mathbf{x}_1 , и аналогично для x_{2l} . Совместное распределение можно записать в виде

$$p(\mathbf{x}_1, \mathbf{x}_2 | \boldsymbol{\mu}) = \prod_{k=1}^K \prod_{l=1}^K \mu_{kl}^{x_{1k} x_{2l}}.$$

Поскольку на параметры μ_{kl} накладывается ограничение $\sum_k \sum_l \mu_{kl} = 1$, это распределение управляетя $K^2 - 1$ параметрами. Отсюда очевидно, что общее число параметров, которые должны быть заданы для произвольного совместного распределения по M переменным, равно $K^M - 1$ и, следовательно, растет экспоненциально с увеличением числа M переменных.

С помощью правила произведения совместное распределение $p(\mathbf{x}_1, \mathbf{x}_2)$ можно представить в виде $p(\mathbf{x}_2 | \mathbf{x}_1)p(\mathbf{x}_1)$, что соответствует графу из двух узлов со связью, идущей от узла \mathbf{x}_1 к узлу \mathbf{x}_2 , как показано на рис. 11.3a. Маргинальное распределение $p(\mathbf{x}_1)$, как и прежде, управляетя $K - 1$ параметрами. Точно так же условное распределение $p(\mathbf{x}_2 | \mathbf{x}_1)$ требует задания $K - 1$ параметров для каждого из K возможных значений \mathbf{x}_1 . Таким образом, общее число параметров, которые должны быть указаны в совместном распределении, равно $(K - 1) + K(K - 1) = K^2 - 1$, как и ранее.

РИС. 11.3 (a) Этот полносвязный граф описывает общее распределение по двум дискретным переменным с K состояниями, имеющим в общей сложности $K^2 - 1$ параметров. (b) Если отбросить связь между узлами, число параметров сократится до $2(K - 1)$

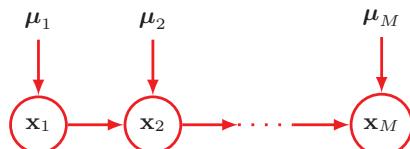


Теперь предположим, что переменные \mathbf{x}_1 и \mathbf{x}_2 независимы, что соответствует модели графа на рис. 11.3b. Тогда каждая переменная описывается отдельным дискретным распределением, и общее число параметров будет

равно $2(K - 1)$. Для распределения по M независимым дискретным переменным, каждая из которых имеет K состояний, общее число параметров составит $M(K - 1)$, и, следовательно, оно линейно растет с увеличением числа переменных. С перспективы графа число параметров уменьшилось за счет отказа от связей в графе, но при этом класс распределений стал более ограниченным.

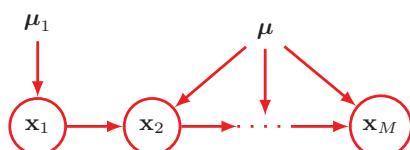
В более общем случае при наличии M дискретных переменных $\mathbf{x}_1, \dots, \mathbf{x}_M$ можно смоделировать совместное распределение с помощью направленного графа с одной переменной для каждого узла. Условное распределение в каждом узле задается набором неотрицательных параметров, подчиняющихся обычному ограничению нормализации. Если граф полносвязный, то получается полностью общее распределение с $K^M - 1$ параметрами, если же в графе нет связей, то совместное распределение сводится к произведению маргинальных распределений, а общее число параметров равно $M(K - 1)$. Графы с промежуточным уровнем связности позволяют получить более общее распределение, нежели полностью факторизованное, и при этом требуют меньшего числа параметров, чем общее совместное распределение. В качестве иллюстрации рассмотрим цепочку узлов, показанную на рис. 11.4. Маргинальное распределение $p(\mathbf{x}_i)$ требует $K - 1$ параметров, тогда как каждое из $M - 1$ условных распределений $p(\mathbf{x}_i | \mathbf{x}_{i-1})$ для $i = 2, \dots, M$ требует $K(K - 1)$ параметров. Это дает общее количество параметров $K - 1 + (M - 1)K(K - 1)$, которое характеризуется квадратичной зависимостью от K и линейно (а не экспоненциально) увеличивается с длиной M цепочки.

РИС. 11.4 Эта цепочка из M дискретных узлов, каждый из которых имеет K состояний, требует указания $K - 1 + (M - 1)K(K - 1)$ параметров, число которых линейно растет с длиной M цепочки. В отличие от этого полносвязный граф из M узлов имел бы $K^M - 1$ параметров, которые растут экспоненциально с M



Альтернативным способом сокращения числа независимых параметров в модели является *совместное использование (sharing)* параметров, также известное как *связывание (tying)* параметров. Так, в примере с цепочкой на рис. 11.4 все условные распределения $p(\mathbf{x}_i | \mathbf{x}_{i-1})$ для $i = 2, \dots, M$ управляются одним и тем же набором $K(K - 1)$ параметров, что приводит к модели, показанной на рис. 11.5. Вместе с $K - 1$ параметрами, управляющими распределением \mathbf{x}_1 , это дает в общей сложности $K^2 - 1$ параметров, которые необходимо указать для определения совместного распределения.

РИС. 11.5 Аналогично рис. 11.4, но с одним набором параметров μ , общим для всех условных распределений $p(\mathbf{x}_i | \mathbf{x}_{i-1})$

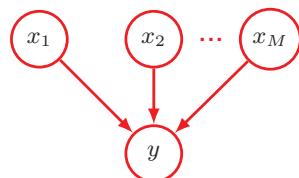


Другой способ борьбы с экспоненциальным ростом числа параметров в моделях дискретных переменных основан на использовании параметризованных представлений для условных распределений вместо полных таблиц значений условных вероятностей. Чтобы проиллюстрировать эту идею, рассмотрим граф на рис. 11.6, где все узлы представляют бинарные переменные. Каждая из родительских переменных x_i управляет одним параметром μ_i , представляющим вероятность $p(x_i = 1)$, что дает в общей сложности M параметров для родительских узлов. Однако условное распределение $p(y|x_1, \dots, x_M)$ потребует $2M$ параметров, представляющих вероятность $p(y = 1)$ для каждого из $2M$ возможных значений родительских переменных. Таким образом, в общем случае количество параметров, необходимых для задания этого условного распределения, будет расти экспоненциально с M . Для условного распределения можно получить более простую форму с помощью логистической сигмоидной функции (см. раздел 3.4), которая действует на линейную комбинацию родительских переменных, что дает

$$p(y = 1|x_1, \dots, x_M) = \sigma\left(w_0 + \sum_{i=1}^M w_i x_i\right) = \sigma(\mathbf{w}^\top \mathbf{x}), \quad (11.8)$$

где $\sigma(a) = (1+\exp(-a))^{-1}$ – это логистическая сигмоидная функция, $\mathbf{x} = (x_0, x_1, \dots, x_M)^\top$ – это $(M+1)$ -мерный вектор родительских состояний, расширенный дополнительной переменной x_0 , значение которой ограничено 1, а $\mathbf{w} = (w_0, w_1, \dots, w_M)^\top$ – это вектор из $M+1$ параметров. Это более ограниченная форма условного распределения, чем в общем случае, но теперь она управляема числом параметров, которое растет линейно с M . В этом смысле она аналогична выбору ограничительной формы ковариационной матрицы (например, диагональной матрицы) в многомерном гауссовом распределении.

РИС. 11.6 Граф из M родителей x_1, \dots, x_M и одного дочернего y , который иллюстрирует идею параметризованных условных распределений для дискретных переменных



11.1.4. Гауссовые переменные

Теперь мы перейдем к изучению моделей графов, в которых узлы представляют собой непрерывные переменные с гауссовыми распределениями. Каждое распределение зависит от состояния его родителей в графе. Эта зависимость может принимать различные формы, но здесь основное внимание будет уделено конкретному варианту, в котором среднее значение каждой гауссовой переменной является некоторой линейной функцией состояний родительских гауссовых переменных. Это приводит к классу моделей, называемых линейными гауссовыми моделями (*linear-Gaussian models, LGM*),

к которым относятся многие практические случаи применения, такие как вероятностный анализ главных компонентов (см. раздел 16.2), факторный анализ и линейные динамические системы (Roweis and Ghahramani, 1999).

Рассмотрим произвольный направленный ациклический граф по D переменным, в котором узел i соответствует одной непрерывной случайной величине x_i с гауссовым распределением. Среднее значение этого распределения принимается за линейную комбинацию состояний родительских узлов $\text{pa}(i)$ узла i :

$$p(x_i | \text{pa}(i)) = \mathcal{N}\left(x_i \middle| \sum_{j \in \text{pa}(i)} w_{ij} x_j + b_i, v_i\right), \quad (11.9)$$

где w_{ij} и b_i – это параметры, определяющие среднее значение, а v_i является дисперсией условного распределения для x_i . Тогда логарифм совместного распределения представляет собой логарифм произведения этих условных распределений по всем узлам графа и, следовательно, имеет вид:

$$\ln p(\mathbf{x}) = \sum_{i=1}^D \ln p(x_i | \text{pa}(i)) \quad (11.10)$$

$$= -\sum_{i=1}^D \frac{1}{2v_i} \left(x_i - \sum_{j \in \text{pa}(i)} w_{ij} x_j - b_i \right)^2 + \text{const}, \quad (11.11)$$

где $\mathbf{x} = (x_1, \dots, x_D)^T$, а const обозначает члены, не зависящие от \mathbf{x} . Как видно, это квадратичная функция компонентов \mathbf{x} , и, следовательно, совместное распределение $p(\mathbf{x})$ является многомерным гауссовым распределением.

Среднее значение и ковариацию этого совместного распределения можно определить следующим образом. Среднее значение каждой переменной (см. упражнение 11.6) задается рекуррентным соотношением:

$$\mathbb{E}[x_i] = \sum_{j \in \text{pa}(i)} w_{ij} \mathbb{E}[x_j] + b_i, \quad (11.12)$$

и поэтому компоненты $\mathbb{E}[\mathbf{x}] = (\mathbb{E}[x_1], \dots, \mathbb{E}[x_D])^T$ можно определить, начав с наименьшего пронумерованного узла и двигаясь рекурсивно по графу; при этом предполагается, что узлы пронумерованы таким образом, что каждый узел имеет больший номер, чем его родители. Аналогичным образом элементы ковариационной матрицы совместного распределения удовлетворяют рекуррентному соотношению (см. упражнение 11.7) вида

$$\text{cov}[x_i, x_j] = \sum_{k \in \text{pa}(i)} w_{jk} \text{cov}[x_i, x_k] + I_{ij} v_j, \quad (11.13)$$

и поэтому ковариацию можно аналогично оценить рекурсивно, начиная с самого нижнего узла.

Теперь рассмотрим два исключительных случая возможных структур графов. Для начала предположим, что в графе нет связей, поэтому он состоит

из D изолированных узлов. В этом случае нет параметров w_{ij} , а значит, есть только D параметров b_i и D параметров v_i . Из рекуррентных соотношений (11.12) и (11.13) следует, что среднее значение $p(\mathbf{x})$ задается $(b_1, \dots, b_D)^T$, а ковариационная матрица является диагональной в виде $\text{diag}(v_1, \dots, v_D)$. Совместное распределение имеет в общей сложности $2D$ параметров и представляет собой набор из D независимых одномерных гауссовых распределений.

Теперь рассмотрим полностью связный граф, в котором каждый узел имеет в качестве родителей все нижележащие узлы. В этом случае общее число независимых параметров $\{w_{ij}\}$ и $\{v_i\}$ в ковариационной матрице (см. упражнение 11.8) равно $D(D + 1)/2$, что соответствует общей симметричной ковариации.

Графы с некоторым промежуточным уровнем сложности соответствуют совместным гауссовым распределениям с частично ограниченными ковариационными матрицами. Рассмотрим, например, граф на рис. 11.7, в котором отсутствует связь между переменными x_1 и x_3 . Используя рекуррентные соотношения (11.12) и (11.13), можно убедиться, что среднее значение и ковариация совместного распределения (см. упражнение 11.9) определяются как

$$\boldsymbol{\mu} = (b_1, b_2 + w_{21}b_1, b_3 + w_{32}b_2 + w_{33}w_{21}b_1)^T, \quad (11.14)$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} v_1 & w_{21}v_1 & w_{32}w_{21}v_1 \\ w_{21}v_1 & v_2 + w_{21}^2v_1 & w_{32}(v_2 + w_{21}^2v_1) \\ w_{32}w_{21}v_1 & w_{32}(v_2 + w_{21}^2v_1) & v_3 + w_{32}^2(v_2 + w_{21}^2v_1) \end{pmatrix}. \quad (11.15)$$

Модель линейного гауссова графа можно также распространить на ситуацию, когда узлы графа представляют собой многомерные гауссовые переменные. В этом случае условное распределение для узла i можно записать в виде

$$p(\mathbf{x}_i | \text{pa}(i)) = \mathcal{N}\left(\mathbf{x}_i \mid \sum_{j \in \text{pa}(i)} \mathbf{W}_{ij} \mathbf{x}_j + \mathbf{b}_i, \boldsymbol{\Sigma}_i\right), \quad (11.16)$$

где в данном случае \mathbf{W}_{ij} – это матрица (которая является неквадратной, если \mathbf{x}_i и \mathbf{x}_j имеют разную размерность). И вновь нетрудно убедиться, что совместное распределение по всем переменным является гауссовым (см. упражнение 11.10).

РИС. 11.7 Направленный граф по трем гауссовым переменным с одной недостающей связью



11.1.5. Бинарный классификатор

Иллюстрацией возможности использования направленных графов для описания распределений вероятностей (см. раздел 2.6.2) может служить модель классификатора двух классов с гауссовым априорным распределением обучаемых параметров. Это можно записать в виде

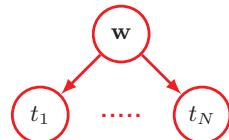
$$p(\mathbf{t}, \mathbf{w} | \mathbf{X}, \lambda) = p(\mathbf{w} | \lambda) \prod_{n=1}^N p(t_n | \mathbf{w}, \mathbf{x}_n), \quad (11.17)$$

где $\mathbf{t} = (t_1, \dots, t_N)^T$ – это вектор целевых значений, \mathbf{X} – это матрица данных со строками $\mathbf{x}_1^T, \dots, \mathbf{x}_N^T$, а распределение $p(\mathbf{t} | \mathbf{x}, \mathbf{w})$ задается в (11.1). Для вектора параметров \mathbf{w} также предполагается гауссова априорная вероятность, которая задается как

$$p(\mathbf{w} | \lambda) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \lambda \mathbf{I}). \quad (11.18)$$

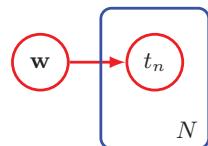
Стохастическими переменными в этой модели являются $\{t_1, \dots, t_N\}$ и \mathbf{w} . Кроме того, в этой модели есть дисперсия шума σ^2 и гиперпараметр λ , которые являются параметрами модели, а не стохастическими переменными. Если рассматривать только стохастические переменные, то распределение, заданное в (11.17), можно представить в виде модели графа на рис. 11.8.

РИС. 11.8 Направленная модель графа, отражающая модель бинарного классификатора с совместным распределением (11.17), где представлены только стохастические переменные $\{t_1, \dots, t_N\}$ и \mathbf{w}



При работе с более сложными моделями становится неудобно записывать множество узлов вида t_1, \dots, t_N в явном виде, как на рис. 11.8. Поэтому для более компактного выражения множества узлов используется графическая нотация. Сначала нужно нарисовать один репрезентативный узел t_n , а затем окружить его рамкой, называемой *табличкой* (*plate*), обозначенной N , чтобы показать, что существует N узлов такого типа. Переписав таким образом график на рис. 11.8, можно получить график, показанный на рис. 11.9.

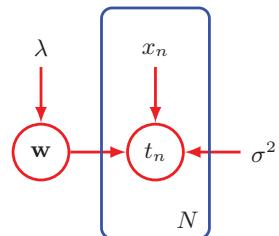
РИС. 11.9 Альтернативное, более компактное представление графа на рис. 11.8, в котором введена табличка (поле с меткой N), представляющая N узлов, из которых только один пример t_n показан в явном виде



11.1.6. Параметры и наблюдения

Иногда бывает полезно указать параметры модели в явном виде, а также ее стохастические переменные в их графовом представлении. Для этого принято обозначать случайные переменные открытыми кругами, а детерминированные параметры обозначаются плавающими переменными. Если рассмотреть график на рис. 11.9 и включить в него детерминированные параметры, то получится график, показанный на рис. 11.10.

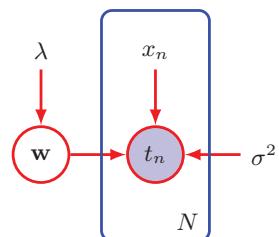
РИС. 11.10 Модель, представленная на рис. 11.9, но с детерминированными параметрами, которые явно обозначены плавающими переменными



При использовании модели графа для решения задач машинного обучения некоторые случайные переменные обычно задаются определенными наблюдаемыми значениями. Например, стохастические переменные $\{t_n\}$ в модели линейной регрессии будут заданы равными конкретным значениям, полученным в обучающем наборе. В модели графа такие наблюдаемые переменные обозначаются затенением соответствующих узлов. Таким образом, граф, соответствующий рис. 11.10, где переменные $\{t_n\}$ являются наблюдаемыми, показан на рис. 11.11.

Обратите внимание, что значение w не является наблюдаемым, поэтому w – это пример *латентной* (*latent*) переменной, также известной как *скрытая* (*hidden*) переменная. Такие переменные играют важную роль во многих моделях, рассматриваемых в этой книге. Таким образом, в направленной модели графа имеется три вида переменных. Во-первых, это ненаблюдаемые (также называемые латентными, или скрытыми) стохастические переменные, которые обозначаются открытыми красными кружками. Во-вторых, когда стохастические переменные являются наблюдаемыми, т. е. устанавливаются на определенные значения, они обозначаются красными кругами, заштрихованными синим цветом. Наконец, нестохастические параметры обозначаются плавающими переменными, как показано на рис. 11.11.

РИС. 11.11 Как и на рис. 11.10, но узлы $\{t_n\}$ заштрихованы, чтобы показать, что соответствующие случайные переменные были установлены на их наблюдаемые значения, заданные обучающим набором

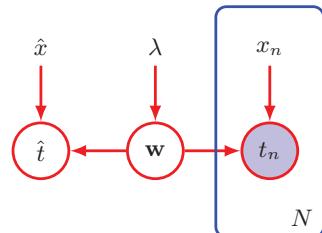


Обратите внимание, что такие параметры модели, как w , обычно не представляют непосредственного интереса, поскольку конечной целью является составление прогнозов для новых входных значений. Предположим, на вход подается новое значение \hat{x} и необходимо найти соответствующее распределение вероятностей для \hat{t} , которое построено на основе наблюдаемых данных. Совместное распределение всех случайных величин в этой модели, обусловленное детерминированными параметрами, задается как

$$p(\hat{t}, \mathbf{t}, \mathbf{w} | \hat{x}, \mathbf{X}, \lambda) = p(\mathbf{w} | \lambda) p(\hat{t} | \mathbf{w}, \hat{x}) \prod_{n=1}^N p(t_n | \mathbf{w}, \mathbf{x}_n) \quad (11.19)$$

и соответствует модели графа на рис. 11.12.

РИС. 11.12 Модель классификации, соответствующая рис. 11.11 и показывающая новое входное значение \hat{x} вместе с соответствующим прогнозом модели \hat{t}



Далее требуемое прогнозное распределение для \hat{t} получается из правила суммы вероятностей путем интегрирования по параметрам модели w . Это интегрирование по параметрам представляет собой полноценную обработку по методу Байеса, которая редко используется на практике, особенно с глубокими нейронными сетями. Вместо этого можно аппроксимировать этот интеграл, предварительно определив наиболее вероятное значение w_{MAP} , которое максимизирует апостериорное распределение, а затем с помощью этого единственного значения получить прогноз на основе $p(\hat{t} | w_{MAP}, \hat{x})$.

11.1.7. Теорема Байеса

Когда стохастические переменные в вероятностной модели устанавливаются равными наблюдаемым значениям, соответственно изменяются распределения других ненаблюдаемых стохастических переменных. Процесс вычисления этих обновленных распределений известен как *вывод (inference)*. Это можно проиллюстрировать, рассмотрев графическую интерпретацию теоремы Байеса. Предположим, что совместное распределение $p(x, y)$ по двум переменным x и y раскладывается на произведение коэффициентов в виде $p(x, y) = p(x)p(y|x)$. Это можно представить в виде направленного графа, показанного на рис. 11.13а. Теперь предположим, что наблюдается значение y , как указано заштрихованным узлом на рис. 11.13б. Можно рассматривать маргинальное распределение $p(x)$ как априорное по латентной переменной x , и тогда задача заключается в выводе соответствующего апостериорного распределения. Используя правила суммы и произведения вероятностей, можно оценить

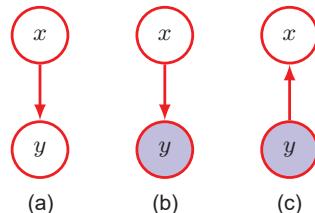
$$p(y) = \sum_{x'} p(y|x')p(x'), \quad (11.20)$$

что затем может быть использовано в теореме Байеса для вычисления

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \quad (11.21)$$

Таким образом, совместное распределение теперь выражается в виде $p(x|y)$ и $p(y)$. Наглядно совместное распределение $p(x,y)$ представлено графиком на рис. 11.13с, где направление стрелки изменено на противоположное. Это простейший пример задачи вывода для модели графа.

РИС. 11.13 Наглядное представление теоремы Байеса: (a) совместное распределение по двум переменным x и y , выраженное в факторизованной форме; (b) пример с y , установленным на наблюдаемое значение; и (c) результирующее апостериорное распределение по x , полученное по теореме Байеса



Для сложных графовых моделей, отражающих насыщенную вероятностную структуру, процесс вычисления апостериорных распределений после того, как некоторые из стохастических переменных будут наблюдаться, может быть сложным и непростым. Концептуально он сводится к систематическому применению правил суммы и произведения вероятностей, или, другими словами, теоремы Байеса. На практике же для эффективного выполнения этих вычислений очень полезно использовать структуру графов. Эти вычисления можно выразить в виде аккуратных вычислений на графе, подразумевающих отправку локальных сообщений между узлами. Такие методы дают точные ответы для графов с древовидной структурой и предоставляют приближенные итерационные алгоритмы для графов с циклами. Поскольку здесь эти методы не рассматриваются, всестороннее их обсуждение в контексте машинного обучения можно найти в (Bishop, 2006).

11.2. Условная независимость

Одной из важнейших концепций распределения вероятностей по нескольким переменным является *свойство условной независимости* (*conditional independence*) (Dawid, 1980). Рассмотрим три переменные a , b и c и предположим, что условное распределение a по b и c таково, что оно не зависит от значения b , так что

$$p(a|b,c) = p(a|c). \quad (11.22)$$

Можно сказать, что a условно не зависит от b с учетом c . Это можно выразить несколько иначе на примере совместного распределения a и b с учетом c , которое можно записать в виде

$$\begin{aligned} p(a,b|c) &= p(a|b,c)p(b|c) \\ &= p(a|c)p(b|c), \end{aligned} \quad (11.23)$$

где используется правило произведения вероятностей вместе с (11.22). Здесь можно увидеть, что при условии c совместное распределение a и b сводится к произведению маргинального распределения a и маргинального распределения b (опять же, оба при условии c). Это говорит о том, что переменные a и b статистически независимы с учетом c . Обратите внимание, что в соответствии с этим определением условной независимости выражение (11.22), или эквивалентно (11.23), будет справедливо для всех возможных значений c , а не только для некоторых их значений. Иногда для условной независимости будет использоваться сокращенная нотация (Dawid, 1979), где

$$a \perp\!\!\! \perp b \mid c \quad (11.24)$$

обозначает, что a условно не зависит от b при заданном c . Свойства условной независимости играют важную роль в вероятностных моделях машинного обучения, поскольку они упрощают как структуру модели, так и вычисления, необходимые для осуществления вывода и обучения в рамках этой модели.

Если задать выражение для совместного распределения по набору переменных в виде произведения условных распределений (т. е. математическое представление, лежащее в основе направленного графа), то с помощью многократного применения правил суммы и произведения вероятностей можно, в принципе, проверить, выполняется ли какое-либо потенциальное свойство условной независимости. На практике такой подход занял бы очень много времени. Важной и эффективной особенностью моделей на основе графов является то, что свойства условной независимости совместного распределения могут быть получены непосредственно из графа без необходимости выполнения каких-либо аналитических манипуляций. Общая схема для достижения этой цели называется *d-разделением* (*d-separation*), где «d» соответствует слову *directed* (направленный) (Pearl, 1988). Здесь будет приведена основная концепция d-разделения и дана общая формулировка критериев d-разделения. Формальное доказательство можно найти в работе (Lauritzen, 1996).

11.2.1. Три примера графов

Начнем обсуждение свойств условной независимости направленных графов с анализа трех простых примеров, в каждом из которых участвуют графы всего с тремя узлами. Вместе они будут служить примером и иллюстрацией ключевых понятий d-разделения. Первый из трех примеров показан на рис. 11.14, и совместное распределение, соответствующее этому графу, удобно вывести с помощью общего результата (11.6), который дает

$$p(a, b, c) = p(a \mid c)p(b \mid c)p(c). \quad (11.25)$$

Если ни одна из переменных не является наблюдаемой, то можно выяснить, являются ли a и b независимыми, используя маргинализацию обеих сторон (11.25) по отношению к c , чтобы получить

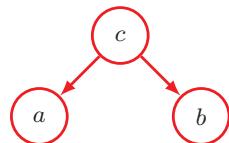
$$p(a, b) = \sum_c p(a|c)p(b|c)p(c). \quad (11.26)$$

В общем случае это не сводится к произведению $p(a)p(b)$, и поэтому

$$a \not\perp\!\!\!\perp b | \emptyset, \quad (11.27)$$

где \emptyset обозначает пустое множество, а символ $\not\perp\!\!\!\perp$ означает, что свойство условной независимости в общем случае не выполняется. Конечно, оно может выполняться для конкретного распределения в силу конкретных числовых значений, связанных с различными условными вероятностями, но в целом из структуры графа этого не следует.

РИС. 11.14 Первый из трех примеров графов по трем переменным a , b и c , используемых для изучения свойств условной независимости направленных моделей графов



Теперь предположим, что для переменной c нужно поставить условие, как показано на графике рис. 11.15. Из (11.25) удобно составить условное распределение a и b с учетом c в виде

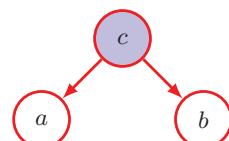
$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= p(a|c)p(b|c), \end{aligned}$$

и, таким образом, получается свойство условной независимости:

$$a \perp\!\!\!\perp b | c.$$

Можно дать простую графическую интерпретацию этого результата, рассмотрев путь от узла a к узлу b через c . В этом случае говорят, что узел c расположен «хвост к хвосту» (*tail-to-tail*) по отношению к этому пути, поскольку он соединен с хвостами двух стрелок, и наличие такого пути, соединяющего узлы a и b , делает эти узлы зависимыми. Но когда выставляется условие на узле c , как на рис. 11.15, этот обусловленный узел «блокирует» путь от a к b и приводит к тому, что a и b становятся (условно) независимыми.

РИС. 11.15 Как и на рис. 11.14, но здесь введено условие на значение переменной c



Аналогичным образом можно представить граф, показанный на рис. 11.16. Совместное распределение, соответствующее этому графу, вновь можно получить из общей формулы (11.6), что дает

$$p(a, b, c) = p(a)p(c|a)p(b|c). \quad (11.28)$$

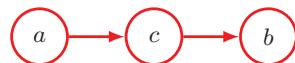
Для начала предположим, что ни одна из переменных не является наблюдаемой. Опять же, можно проверить, являются ли a и b независимыми, сделав маргинализацию по c , что дает

$$p(a, b) = p(a) \sum_c p(c|a)p(b|c) = p(a)p(b|a),$$

которое в общем случае не факторизуется в $p(a)p(b)$, и поэтому, как и раньше,

$$a \not\perp\!\!\!\perp b | \emptyset. \quad (11.29)$$

РИС. 11.16 Второй из трех примеров графов с тремя узлами, используемых для обоснования концепции условной независимости для моделей с направленными графами



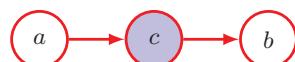
Теперь представим, что условие задано на узле c , как показано на рис. 11.17. Используя теорему Байеса вместе с (11.28), получаем:

$$\begin{aligned} p(a, b | c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c), \end{aligned}$$

и, таким образом, свойство условной независимости вновь соблюдается:

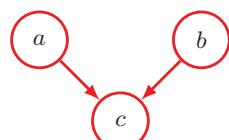
$$a \perp\!\!\!\perp b | c.$$

РИС. 11.17 Как и на рис. 11.16, но теперь условие определяется узлом c



Как и ранее, эти результаты можно интерпретировать графически. В этом случае говорят, что узел c находится в позиции пути «голова к хвосту» (*head-to-tail*) относительно пути от узла a к узлу b . Такой путь соединяет узлы a и b и делает их зависимыми. Если теперь наблюдается c , как на рис. 11.17, то это событие «блокирует» путь от a к b , и поэтому свойство условной независимости $a \perp\!\!\!\perp b | c$ соблюдается.

РИС. 11.18 Последний из трех примеров графов с тремя узлами, используемых для изучения свойств условной независимости в графических моделях. Этот график имеет совсем другие свойства, чем два предыдущих примера



Наконец, рассмотрим третий из наших примеров с тремя узлами, представленный графом на рис. 11.18. Как видно, он имеет более сложное строение, чем два предыдущих графа. Совместное распределение вновь можно записать с использованием общего результата (11.6), чтобы получить

$$p(a, b, c) = p(a)p(b)p(c|a, b). \quad (11.30)$$

Рассмотрим сначала случай, когда ни одна из переменных не является наблюдаемой. Маргинализируя обе стороны (11.30) по c , получим

$$p(a, b) = p(a)p(b),$$

и, таким образом, a и b независимы при отсутствии наблюдаемых переменных, в отличие от двух предыдущих примеров. Этот результат можно записать в виде

$$a \perp\!\!\!\perp b | \emptyset. \quad (11.31)$$

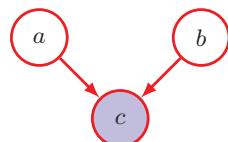
Теперь предположим, что нужно поставить условие на c , как показано на рис. 11.19. Тогда условное распределение a и b будет иметь вид:

$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(b)p(c|a, b)}{p(c)}, \end{aligned}$$

что в общем случае не сводится к произведению $p(a|c)p(b|c)$, и поэтому

$$a \not\perp\!\!\!\perp b | c.$$

РИС. 11.19 Как на рис. 11.18, но с условием на значение узла c . В этом графе действие условия приводит к зависимости между a и b



Получается, третий пример демонстрирует поведение, противоположное первым двум. Графически можно определить, что узел c является головным по отношению к пути из a к b , поскольку он соединяется с вершинами двух стрелок. Узел c иногда называют узлом-коллайдером (*collider node*). Когда узел c является ненаблюдаемым, он «блокирует» путь, и переменные a и b оказываются независимыми. Однако при условии, что c «разблокирует» путь, a и b становятся зависимыми.

С этим третьим примером связана еще одна тонкость, которую важно прояснить. Сначала введем еще немного терминологии. Говорят, что узел y является потомком (*descendant*) узла x , если существует путь от x к y , на каждом

шаге которого направления стрелок совпадают. Тогда можно доказать, что путь «голова к голове» (*head-to-head*) (см. упражнение 11.13) будет разблокирован, если будет наблюдаемым либо узел, либо любой из его потомков.

Получается, что узел «хвост к хвосту» или узел «голова к голове» оставляет путь свободным, если только он не наблюдается, и в этом случае он будет блокировать путь. Напротив, узел «голова к голове» блокирует путь, если он не наблюдается, но как только этот узел и/или хотя бы один из его потомков становятся наблюдаемыми, путь становится разблокированным.

11.2.2. Объяснения

Здесь имеет смысл потратить немного времени на то, чтобы разобраться в необычном поведении графа на рис. 11.19. Рассмотрим конкретный пример такого графа, соответствующий задаче с тремя бинарными случайными величинами, которые имеют отношение к топливной системе автомобиля, как показано на рис. 11.20. Здесь имеются следующие переменные: B – это состояние аккумулятора, который либо заряжен ($B = 1$), либо разряжен ($B = 0$), F – это состояние топливного бака, который либо полон топлива ($F = 1$), либо пуст ($F = 0$), и G – это состояние датчика уровня топлива, который показывает, что бензобак либо полон ($G = 1$), либо пуст ($G = 0$). Аккумулятор либо заряжен, либо разряжен, и независимо от этого топливный бак либо полон, либо пуст с априорными вероятностями:

$$p(B = 1) = 0,9,$$

$$p(F = 1) = 0,9.$$

Принимая во внимание состояние топливного бака и аккумулятора, указатель уровня топлива показывает, что он полон, с вероятностями, заданными

$$p(G = 1 | B = 1, F = 1) = 0,8,$$

$$p(G = 1 | B = 1, F = 0) = 0,2,$$

$$p(G = 1 | B = 0, F = 1) = 0,2,$$

$$p(G = 1 | B = 0, F = 0) = 0,1,$$

так что это довольно ненадежный датчик топлива! Все остальные вероятности определяются условием, чтобы вероятности в сумме равнялись единице, и, таким образом, получается полная спецификация вероятностной модели.

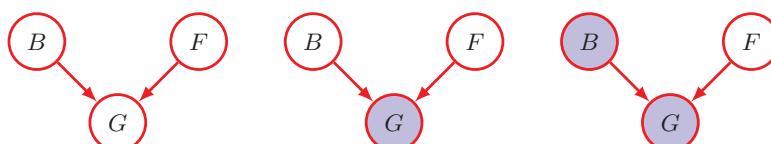


РИС. 11.20 Пример графа с тремя узлами, используемого для иллюстрации «объяснения»

Пока не получено никаких данных, априорная вероятность того, что топливный бак пуст, равна $p(F = 0) = 0,1$. Теперь предположим, что при наблюдении за указателем уровня топлива выясняется, что бензобак пуст, т. е. $G = 0$, что соответствует среднему графику на рис. 11.20. С помощью теоремы Байеса можно оценить апостериорную вероятность того, что топливный бак пуст. Сначала определим знаменатель теоремы Байеса:

$$p(G = 0) = \sum_{B \in \{0,1\}} \sum_{F \in \{0,1\}} p(G = 0|B, F)p(B)p(F) = 0,315 \quad (11.32)$$

и аналогичным образом оцениваем:

$$p(G = 0|F = 0) = \sum_{B \in \{0,1\}} p(G = 0|B, F = 0)p(B) = 0,81. \quad (11.33)$$

Используя эти результаты, получаем

$$p(F = 0|G = 0) = \frac{p(G = 0|F = 0)p(F = 0)}{p(G = 0)} \simeq 0,257, \quad (11.34)$$

и поэтому $p(F = 0|G = 0) > p(F = 0)$. Таким образом, наблюдение за тем, что датчик показывает пустой бак, повышает вероятность того, что бак действительно пуст, как и следовало ожидать на интуитивном уровне. Далее предположим, что при проверке состояния аккумулятора выяснилось, что он разряжен, т. е. $B = 0$. Теперь можно наблюдать состояния и топливного датчика, и аккумулятора, как показано на правом графике рис. 11.20. Апостериорная вероятность того, что топливный бак пуст, с учетом наблюдений за состоянием указателя уровня топлива и аккумулятора, определяется как

$$p(F = 0|G = 0, B = 0) = \frac{p(G = 0|B = 0, F = 0)p(F = 0)}{\sum_{F \in \{0,1\}} p(G = 0|B = 0, F)p(F)} \simeq 0,111, \quad (11.35)$$

где априорная вероятность $p(B = 0)$ между числителем и знаменателем исчезла. Таким образом, вероятность того, что бак пуст, в результате наблюдения за состоянием аккумулятора уменьшилась (с 0,257 до 0,111). Это согласуется с предположением, что факт разряда аккумулятора объясняет наблюдение за тем, что указатель уровня топлива свидетельствует о пустом баке. Теперь видно, что состояние топливного бака и состояние аккумулятора действительно стали зависеть друг от друга в результате наблюдения за показаниями датчика уровня топлива. Фактически так было бы и в том случае, если бы вместо непосредственного наблюдения за топливным датчиком велось бы наблюдение за состоянием какого-нибудь потомка G , например довольно сомнительного свидетеля (см. упражнение 11.14), который бы сообщил о том, что датчик показывал пустой бак. Обратите внимание, что вероятность $p(F = 0|G = 0, B = 0) \simeq 0,111$ больше, чем априорная вероятность $p(F = 0) = 0,1$, поскольку наблюдение за тем, что показания датчика уровня топлива равны нулю, все же дает некоторое свидетельство в пользу пустого бензобака.

11.2.3. D-разделение

Теперь дадим общую формулировку свойства d-разделения (Pearl, 1988) для направленных графов. Рассмотрим общий направленный граф, в котором A , B и C – это произвольные непересекающиеся множества узлов (объединение которых может быть меньше полного множества узлов в графе). Необходимо выяснить, вытекает ли конкретное условное утверждение независимости $A \perp\!\!\!\perp B | C$ из заданного направленного ациклического графа. Для этого рассмотрим все возможные пути из любого узла множества A в любой узел множества B . Любой такой путь считается *блокированным*, если он включает такой узел, и тогда:

- (a) стрелки на этом пути встречаются в этом узле либо «голова к хвосту», либо «хвост к хвосту», и этот узел находится в множестве C ,
- (b) либо стрелки встречаются в этом узле «голова к голове», и ни этот узел, ни любой из его потомков не находится в множестве C .

Если все пути перекрыты, то говорят, что A имеет d-разделение от B с помощью C , и совместное распределение по всем переменным в графе будет соответствовать $A \perp\!\!\!\perp B | C$.

D-разделение показано на рис. 11.21. В графе (a) путь из a к b не блокируется узлом f , потому что он является узлом «хвост к хвосту» для этого пути и не наблюдается, а также не блокируется узлом e , поскольку, хотя последний и является узлом «голова к голове», у него есть потомок c в обусловленном множестве. Таким образом, из этого графа *не следует* утверждение об условной независимости $a \perp\!\!\!\perp b | c$. В графе (b) путь от a к b блокирует узел f , поскольку это наблюдаемый узел «хвост к хвосту», и поэтому свойство условной независимости $a \perp\!\!\!\perp b | f$ будет выполняться любым распределением, факторизованным в соответствии с этим графом. Обратите внимание, что этот путь также блокируется узлом e , поскольку e – это узел «голова к голове», и ни он, ни его потомок не входят в множество условия. В d-разделении параметры, такие как λ на рис. 11.12, обозначенные плавающими переменными, ведут себя так же, как и наблюдаемые узлы. Однако с такими узлами не связано

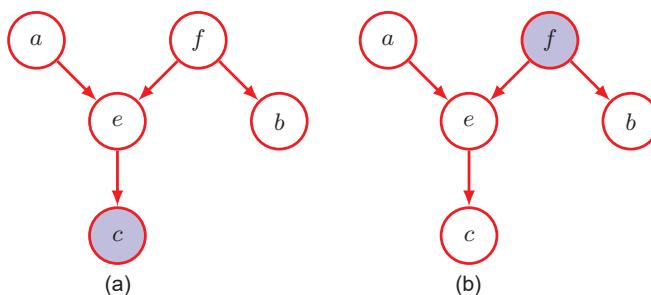


РИС. 11.21 Иллюстрация d-разделения

никаких маргинальных распределений, и, следовательно, узлы параметров никогда не имеют родителей, а значит, все пути через эти узлы всегда будут «хвост к хвосту» и, таким образом, будут заблокированы. Следовательно, они не играют никакой роли в d-разделении.

Другой пример условной независимости и d-разделения дают данные i.i.d. (независимые и идентично распределенные) (см. раздел 2.3.2). Рассмотрим модель бинарной классификации, показанную на рис. 11.12. Здесь стохастические узлы соответствуют $\{t_n\}$, w и \hat{t} . Как видно, узел для w – это «хвост к хвосту» по отношению к пути от \hat{t} к любому из узлов t_n , и поэтому выполняется следующее свойство условной независимости:

$$\hat{t} \perp\!\!\!\perp t_n | w. \quad (11.36)$$

Таким образом, в зависимости от параметров сети w прогнозируемое распределение для \hat{t} не зависит от обучающих данных $\{t_1, \dots, t_N\}$. Поэтому сначала можно использовать обучающие данные для определения апостериорного распределения (или некоторого приближения к апостериорному распределению) для коэффициентов w , а затем отбросить обучающие данные и использовать апостериорное распределение для w для прогнозирования \hat{t} по новым входным наблюдениям \hat{x} .

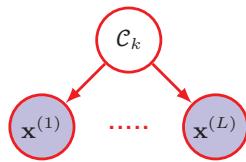
11.2.4. Наивный Байес

Похожая графическая структура возникает при классификации, называемой *моделью наивного Байеса* (*naive Bayes model*), где для упрощения структуры модели используются предположения об условной независимости. Пусть данные состоят из наблюдений вектора x , и необходимо отнести значения x к одному из K классов. Для каждого из классов можно определить условную плотность $p(x|\mathcal{C}_k)$, а также априорные вероятности классов $p(\mathcal{C}_k)$. Ключевое предположение модели наивного Байеса состоит в том, что при условии принадлежности к классу \mathcal{C}_k распределение входной переменной факторизуется в виде произведения двух или более плотностей. Предположим, что x разбивается на L элементов $x = (x^{(1)}, \dots, x^{(L)})$. В этом случае метод наивного Байеса принимает вид:

$$p(x|\mathcal{C}_k) = \prod_{l=1}^L p(x^{(l)}|\mathcal{C}_k), \quad (11.37)$$

и предполагается, что (11.37) справедливо для каждого из классов \mathcal{C}_k в отдельности. Графическое представление этой модели показано на рис. 11.22. Здесь видно, что наблюдение \mathcal{C}_k блокирует путь между $x^{(i)}$ и $x^{(j)}$ для $j \neq i$, поскольку такие пути пересекаются в узле \mathcal{C}_k , и поэтому $x^{(i)}$ и $x^{(j)}$ условно независимы с учетом \mathcal{C}_k . Однако если маргинализировать \mathcal{C}_k , то путь от $x^{(i)}$ к $x^{(j)}$ больше не блокируется, что говорит об отсутствии факторизации предельной плотности $p(x)$ относительно элементов $x^{(1)}, \dots, x^{(L)}$.

РИС. 11.22 Графическое представление модели наивного Байеса для классификации. Элементы наблюдаемого вектора $\mathbf{x} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)})$ с условием по метке класса C_k



При наличии маркированного обучающего набора, состоящего из наблюдений $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ вместе с метками классов, можно подогнать модель наивного Байеса к обучающим данным с помощью максимального правдоподобия, допуская, что данные получены из модели независимо друг от друга (см. упражнение 11.15). Решение можно получить путем подгонки модели для каждого класса отдельно, используя соответствующие помеченные данные, а затем установив априорные значения классов $p(C_k)$ равными доле точек обучающих данных в каждом классе. Вероятность того, что вектор \mathbf{x} принадлежит классу C_k , определяется по теореме Байеса в виде

$$p(C_k | \mathbf{x}) = \frac{p(\mathbf{x} | C_k)p(C_k)}{p(\mathbf{x})}, \quad (11.38)$$

где $p(\mathbf{x} | C_k)$ задается в (11.37), а для оценки $p(\mathbf{x})$ можно использовать

$$p(\mathbf{x}) = \sum_{k=1}^K p(\mathbf{x} | C_k)p(C_k). \quad (11.39)$$

Модель наивного Байеса проиллюстрирована для двумерного пространства данных на рис. 11.23, где $\mathbf{x} = (x_1, x_2)$. Здесь предполагается, что условные плотности $p(x_i | C_k)$ для каждого из двух классов являются выровненными по оси гауссовыми распределениями и, следовательно, каждый из них факторизуется относительно x_1 и x_2 так, что

$$p(\mathbf{x} | C_k) = p(x_1 | C_k)p(x_2 | C_k). \quad (11.40)$$

Однако предельная плотность $p(\mathbf{x})$, заданная выражением

$$p(\mathbf{x}) = \sum_{k=1}^K p(\mathbf{x} | C_k)p(C_k), \quad (11.41)$$

теперь является смесью гауссовых функций и не факторизуется относительно x_1 и x_2 . Применения модели наивного Байеса уже встречались ранее в контексте объединения данных из разных источников, таких как анализы крови и изображения кожи для медицинской диагностики (см. раздел 5.2.4).

Применение предположения наивного Байеса полезно, когда размерность D входного пространства велика, что усложняет оценку плотности в полном D -мерном пространстве. Оно также полезно, если входной вектор содержит как дискретные, так и непрерывные переменные, поскольку каждая из них может быть представлена отдельно с помощью соответствующих моделей (например, распределений Бернулли для бинарных наблюдений

или распределений Гаусса для вещественных переменных). Предположения об условной независимости в этой модели, безусловно, являются строгими, что может привести к довольно плохому представлению условных плотностей классов. Тем не менее, даже если это предположение не выполняется точно, модель все равно может давать хорошие результаты классификации на практике, поскольку границы принятия решений могут быть нечувствительны к некоторым деталям в условных плотностях классов, как показано на рис. 5.8.

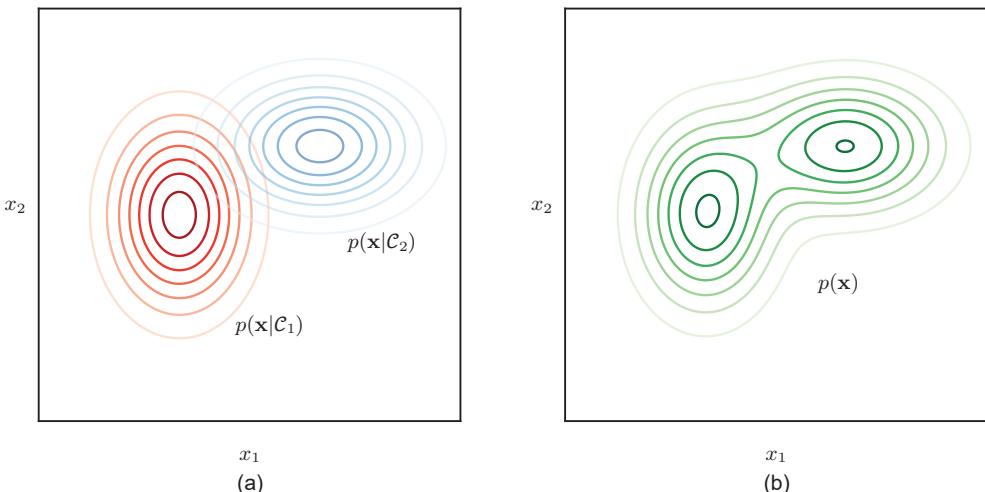


РИС. 11.23 Иллюстрация классификатора наивного Байеса для двумерного пространства данных, где показаны (a) условные распределения $p(\mathbf{x}|\mathcal{C}_k)$ для каждого из двух классов и (b) маргинальное распределение $p(\mathbf{x})$, в котором приняты равные априорные значения классов $p(\mathcal{C}_1) = p(\mathcal{C}_2) = 0,5$. Обратите внимание, что условные распределения факторизуются относительно x_1 и x_2 , а маргинальное распределение — нет

11.2.5. Генеративные модели

Многие приложения машинного обучения можно рассматривать как примеры обратных задач (*inverse problems*), при решении которых существует базовый, зачастую физический процесс, генерирующий данные, а цель состоит в обучении инвертированию этого процесса. Например, изображение объекта можно рассматривать как результат генеративного процесса, в котором тип объекта выбирается из некоторого распределения возможных классов объектов. Положение и ориентация объекта также выбираются из некоторых априорных распределений, а затем создается результирующее изображение. Учитывая большой набор данных изображений, маркированных типом, положением и масштабом содержащихся в них объектов, задача состоит в обучении модели машинного обучения, которая может принимать новые, немаркованные изображения и определять наличие объекта, а также его местоположение на изображении и размер. Таким образом, решение

для машинного обучения представляет собой процесс, обратный тому, который привел к появлению данных.

Один из подходов заключается в обучении глубокой нейронной сети, такой как сверточная сеть, получать на вход изображение и генерировать выходные данные, описывающие тип, положение и масштаб объекта. Такой подход направлен на решение обратной задачи напрямую и является примером *дискриминативной модели* (*discriminative model*). Он может обеспечить высокую точность при наличии большого количества примеров маркированных изображений. На практике немаркированных изображений зачастую довольно много, и большая часть усилий по получению обучающего набора уходит на проверку меток, которая может выполняться вручную. Простая дискриминативная модель не может напрямую использовать немаркированные изображения в процессе своего обучения.

Альтернативный подход заключается в моделировании генеративного процесса и последующей его инверсии вычислительным путем. В нашем примере с изображением, если предположить, что класс, положение и масштаб объекта выбираются независимо, можно представить генеративный процесс с помощью направленной модели в виде графа, как показано на рис. 11.24. Здесь идентичность объекта (дискретная переменная), а также положение и ориентация объекта (непрерывные переменные) имеют независимые априорные вероятности. Изображение (массив интенсивностей пикселей) имеет распределение вероятностей, зависящее от идентичности объекта, а также от его положения и ориентации. Обратите внимание, что направления стрелок соответствуют последовательности генеративных шагов, и, таким образом, модель представляет собой *причинно-следственный* (*causal*) процесс (Pearl, 1988), с помощью которого создаются наблюдаемые данные. Это пример *генеративной модели*, поскольку после обучения ее можно использовать для создания синтетических изображений, сначала выбрав значения класса, положения и масштаба объекта из обученных априорных распределений, а затем выполнив выборку изображения из обученного условного распределения. Позже будет рассказано об использовании диффузионных и других генеративных моделей для синтеза впечатляющих изображений высокого разрешения (см. главу 20) на основе текстового описания желаемого содержания и стиля изображения.

РИС. 11.24 Графическая модель, представляющая процесс создания изображений объектов

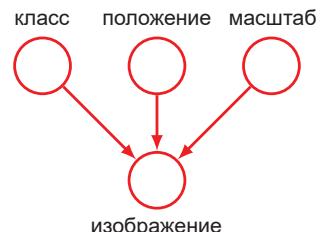


График на рис. 11.24 показывает, что при отсутствии изображения переменные класса, положения и масштаба независимы. Это следует из того, что

каждый путь между любыми двумя этими переменными является прямым по отношению к переменной изображения, которая не наблюдается. При наблюдении изображения эти пути становятся свободными, и переменные класса, положения и масштаба перестают быть независимыми. Интуитивно это вполне обоснованно, так как знание идентичности объекта на изображении дает очень важную информацию, помогающую определить его местоположение.

Однако скрытые переменные в вероятностной модели не обязательно должны иметь явную физическую интерпретацию, они могут быть введены просто для возможности построения более сложного совместного распределения из более простых компонентов. Например, такие модели, как нормализующие потоки, вариационные автокодировщики и диффузионные модели, используют глубокие нейронные сети для создания сложных распределений в пространстве данных путем преобразования скрытых переменных, имеющих простое гауссово распределение.

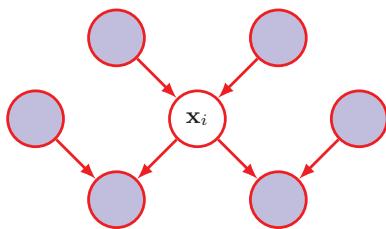
11.2.6. Покрытие Маркова

Свойство условной независимости, которое полезно использовать при анализе более сложных направленных графов, называется *покрытием Маркова* (*Markov blanket*, *марковское одеяло*) или *границей Маркова* (*Markov boundary*, *марковское ограждение*). Рассмотрим совместное распределение $p(\mathbf{x}_1, \dots, \mathbf{x}_D)$, представленное направленным графом с D узлами, и рассмотрим условное распределение конкретного узла с переменными \mathbf{x}_i , обусловленное всеми остальными переменными $\mathbf{x}_{j \neq i}$. Используя свойство факторизации (11.6), можно выразить это условное распределение в виде

$$\begin{aligned} p(\mathbf{x}_i | \mathbf{x}_{\{j \neq i\}}) &= \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_D)}{\int p(\mathbf{x}_1, \dots, \mathbf{x}_D) d\mathbf{x}_i} \\ &= \frac{\prod_k p(\mathbf{x}_k | \text{pa}(k))}{\int \prod_k p(\mathbf{x}_k | \text{pa}(k)) d\mathbf{x}_i}, \end{aligned}$$

в котором интеграл заменен суммированием по дискретным переменным. Теперь можно заметить, что любой фактор $p(\mathbf{x}_k | \text{pa}(k))$, не имеющий функциональной зависимости от \mathbf{x}_i , можно вынести за пределы интеграла по \mathbf{x}_i , следовательно, отменить числитель и знаменатель. Останутся только условные распределения $p(\mathbf{x}_i | \text{pa}(i))$ для самого узла \mathbf{x}_i , а также условные распределения для любых узлов \mathbf{x}_k таких, что узел \mathbf{x}_i находится в обусловленном множестве $p(\mathbf{x}_k | \text{pa}(k))$, другими словами, для которых \mathbf{x}_i является родителем \mathbf{x}_k . Условие $p(\mathbf{x}_i | \text{pa}(i))$ будет зависеть от родителей узла \mathbf{x}_i , тогда как условия $p(\mathbf{x}_k | \text{pa}(k))$ будут зависеть от дочерних узлов \mathbf{x}_i , а также от сородителей, т. е. переменных, соответствующих родителям узла \mathbf{x}_k , отличным от узла \mathbf{x}_i . Набор узлов, включающий родителей, потомков и сородителей, который и называют марковским покрытием, показан на рис. 11.25.

РИС. 11.25 Марковское покрытие узла x_i включает множество родителей, потомков и сородичей этого узла. Оно обладает тем свойством, что условное распределение x_i , обусловленное всеми остальными переменными в графе, зависит только от переменных в марковском покрытии



Марковское покрытие узла x_i можно представить в виде минимального набора узлов, который изолирует x_i от остальной части графа. Обратите внимание, что недостаточно включить только родителей и потомков узла x_i , поскольку объяснение этого означает, что наблюдения за дочерними узлами не будут блокировать пути к сородителям. Поэтому необходимо также наблюдать за узлами-сородичами.

11.2.7. Графы в качестве фильтров

Ранее уже было показано, что определенный направленный граф представляет собой конкретное разложение совместного распределения вероятностей в произведение условных вероятностей, а также выражает набор утверждений об условной независимости, полученных с помощью критерия d-разделения. Теорема о d-разделении на самом деле является выражением эквивалентности этих двух свойств. Чтобы прояснить это, можно представить направленный граф в качестве фильтра. Предположим, речь идет о конкретном совместном распределении вероятностей $p(x)$ по переменным x , соответствующим (ненаблюдаемым) узлам графа. Фильтр пропустит это распределение, если и только если оно может быть выражено в виде факторизации (11.6), обусловленной графиком. Если предоставить фильтру множество всех возможных распределений $p(x)$ на множестве переменных x , то подмножество распределений, пропускаемых фильтром, обозначают DF , что является сокращением от *directed factorization* (направленная факторизация). Это показано на рис. 11.26.

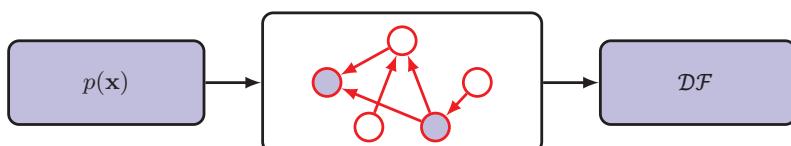


РИС. 11.26 Графовую модель (в данном случае направленный граф) можно рассматривать как фильтр, где распределение вероятностей $p(x)$ пропускается через фильтр тогда и только тогда, когда оно удовлетворяет свойству направленной факторизации (11.6)

В качестве альтернативы можно использовать граф для фильтрации распределений в зависимости от того, удовлетворяют ли они всем свойствам условной независимости, вытекающим из d-разделительных свойств графа.

Теорема о d -разделении утверждает, что через этот второй вид фильтра будет пропущен тот же набор распределений \mathcal{DF} .

Следует подчеркнуть, что свойства условной независимости, полученные с помощью d -разделения, применимы к любой вероятностной модели, описываемой соответствующим направленным графом. Это будет справедливо, например, независимо от того, являются ли переменные дискретными, непрерывными или их комбинацией. И вновь можно заметить, что определенный граф описывает целое семейство распределений вероятностей.

В одном крайнем случае можно получить полносвязный граф, который вообще не проявляет свойств условной независимости и может представлять любое возможное совместное распределение вероятностей по заданным переменным. Множество \mathcal{DF} будет содержать все возможные распределения $p(\mathbf{x})$. В другом крайнем случае получается полностью несвязный граф, т. е. граф вообще без связей. Это соответствует совместным распределениям, которые факторизуются в виде произведения маргинальных распределений по переменным, входящим в узлы графа. Обратите внимание, что для любого заданного графа набор распределений \mathcal{DF} будет включать любые распределения, обладающие дополнительными свойствами независимости, помимо тех, которые описываются самим графом. Например, полностью факторизованное распределение всегда будет проходить через фильтр, накладываемый любым графом на соответствующий набор переменных.

11.3. Модели последовательностей

Есть много важных областей применения машинного обучения, где данные представляют собой *последовательность* (*sequence*) значений. Например, текст состоит из последовательности слов, а белок представляет собой последовательность аминокислот. Многие последовательности упорядочены по времени, например аудиосигналы с микрофона или ежедневные измерения количества осадков в определенной местности. Иногда термины «время», «прошлое» и «будущее» используются для обозначения других типов последовательных данных, а не только временных последовательностей. К числу приложений, в которых используются последовательности, относятся такие, как распознавание речи, автоматический перевод с одного языка на другой, выявление генов в ДНК, синтез музыки, написание компьютерного кода, диалог с современной поисковой системой и многие другие.

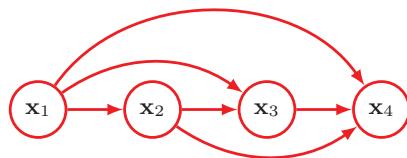
Обозначим последовательность данных через $\mathbf{x}_1, \dots, \mathbf{x}_N$, где каждый элемент \mathbf{x}_n последовательности представляет собой вектор значений. Обратите внимание, что можно работать с несколькими такими последовательностями, взятыми независимо из одного и того же распределения, и тогда общее распределение по всем последовательностям сводится к произведению распределений по каждой последовательности в отдельности. В дальнейшем речь пойдет о моделировании только одной из таких последовательностей.

В (11.4) уже было показано, что путем многократного применения правила произведения вероятностей общее распределение по N переменным можно записать в виде произведения условных распределений, при этом форма этого разбиения зависит от определенной упорядоченности переменных. Если выбрать расположение, соответствующее порядку переменных в последовательности, то для векторных переменных можно записать:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}). \quad (11.42)$$

Это соответствует направленному графу, в котором каждый узел получает ссылку от каждого предыдущего узла в последовательности, как показано с помощью четырех переменных на рис. 11.27. Это называется *моделью авторегрессии (autoregressive model)*.

РИС. 11.27 Иллюстрация общей авторегрессионной модели вида (11.42) с четырьмя узлами



Это представление характеризуется полной обобщенностью и, следовательно, не имеет никакой практической ценности с позиции моделирования, поскольку не содержит никаких предположений. Можно ограничить пространство моделей, введя свойства условной независимости путем удаления связей из графа или, что равнозначно, путем удаления переменных из обусловленного множества факторов в правой части (11.42).

Самым сильным предположением будет удаление всех обусловливающих переменных, что даст совместное распределение вида

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{n=1}^N p(\mathbf{x}_n), \quad (11.43)$$

в котором переменные рассматриваются как независимые и, следовательно, полностью игнорируется информация об упорядочении. Это соответствует вероятностной графовой модели без связей, как показано на рис. 11.28.



РИС. 11.28 Самый простой подход к моделированию последовательности наблюдений: считать их независимыми, что соответствует вероятностной графической модели без связей

Интересные модели, которые отражают свойства последовательности и при этом включают модельные допущения, находятся между этими двумя

крайностями. Одним из сильных допущений было бы предположение, что каждое условное распределение зависит только от непосредственно предшествующей переменной в последовательности, что дает совместное распределение вида

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^N p(\mathbf{x}_n | \mathbf{x}_{n-1}). \quad (11.44)$$

Обратите внимание, что первая переменная в последовательности рассматривается несколько по-другому, поскольку у нее нет обуславливающей переменной. Функциональная форма (11.44) известна как *модель Маркова* (*Markov model*), или *цепь Маркова* (*Markov chain*). Она представлена в виде графа, состоящего из простой цепочки узлов, как показано на рис. 11.29 (см. раздел 11.2.3). Если использовать d-разделение, то условное распределение для наблюдения \mathbf{x}_n с учетом всех наблюдений до момента времени n будет иметь вид:

$$p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}) = p(\mathbf{x}_n | \mathbf{x}_{n-1}), \quad (11.45)$$

что нетрудно проверить путем непосредственной оценки, начиная с (11.44) и используя правило произведения вероятностей (см. упражнение 11.16). Таким образом, если использовать такую модель для прогнозирования следующего наблюдения в последовательности, то распределение предсказаний будет зависеть только от значения непосредственно предшествующего наблюдения и не будет зависеть от всех предыдущих наблюдений.



РИС. 11.29 Цепь Маркова первого порядка, в которой распределение конкретного наблюдения \mathbf{x}_n зависит от значения предыдущего наблюдения \mathbf{x}_{n-1}

Говоря более конкретно, уравнение (11.44) известно в качестве модели Маркова первого порядка, поскольку в каждом условном распределении присутствует только одна переменная. Можно расширить эту модель, если допустить, что каждое условное распределение зависит от двух предшествующих переменных, что приводит к марковской модели второго порядка вида

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2 | \mathbf{x}_1) \prod_{n=3}^N p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{x}_{n-2}). \quad (11.46)$$

Обратите внимание, что первые две переменные рассматриваются иначе, поскольку у них меньше двух обуславливающих переменных. Эта модель показана в виде направленного графа на рис. 11.30.

Используя d-разделение (или непосредственную оценку по правилам теории вероятностей) (см. упражнение 11.17), можно увидеть, что в марковской

модели второго порядка условное распределение x_n с учетом всех предыдущих наблюдений x_1, \dots, x_{n-1} не зависит от наблюдений x_1, \dots, x_{n-3} . Точно так же можно рассмотреть расширения до марковской цепи M -го порядка, в которой условное распределение для конкретной переменной зависит от M предыдущих переменных. Однако за эту повышенную гибкость приходится расплачиваться тем, что число параметров в модели теперь значительно увеличивается. Предположим, что наблюдения – это дискретные переменные с K состояниями. Тогда условное распределение $p(x_n | x_{n-1})$ в цепи Маркова первого порядка будет задано набором из $K - 1$ параметров для каждого из K состояний x_{n-1} , что в сумме дает $K(K - 1)$ параметров. Теперь предположим, что модель расширяется до цепи Маркова M -го порядка, так что совместное распределение строится из условий $p(x_n | x_{n-M}, \dots, x_{n-1})$. Если переменные дискретны, а условные распределения представлены общими таблицами условных вероятностей, то такая модель будет иметь $K^{M-1}(K - 1)$ параметров. Получается, что число параметров растет экспоненциально с увеличением M , что в общем случае при больших значениях M делает этот метод непрактичным.

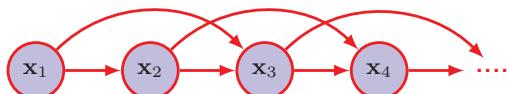


РИС. 11.30 Цепь Маркова второго порядка, где условное распределение конкретного наблюдения x_n зависит от значений двух предыдущих наблюдений x_{n-1} и x_{n-2}

11.3.1. Латентные переменные

Предположим, что требуется построить модель для последовательностей, которая не ограничивается каким-либо порядком в соответствии с предположением по модели Маркова и может быть задана с помощью ограниченного числа свободных параметров. Это можно сделать путем введения дополнительных латентных переменных, что позволит построить обширный класс моделей из простых компонентов. Для каждого наблюдения x_n вводится соответствующая латентная переменная z_n (которая может быть другого типа или размерности, чем наблюданная переменная). Теперь предположим, что именно латентные переменные образуют цепь Маркова, и в результате получается графовая структура, известная как *модель пространства состояний (state-space model)*, как показано на рис. 11.31. Она удовлетворяет ключевому свойству условной независимости, согласно которому z_{n-1} и z_{n+1} независимы от z_n , так что

$$z_{n+1} \perp\!\!\!\perp z_{n-1} | z_n. \quad (11.47)$$

Совместное распределение для этой модели задается формулой:

$$p(x_1, \dots, x_N, z_1, \dots, z_N) = p(z_1) \left[\prod_{n=2}^N p(z_n | z_{n-1}) \right] \prod_{n=1}^N p(x_n | z_n). \quad (11.48)$$

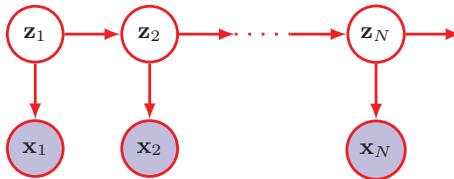


РИС. 11.31 Модель пространства состояний выражает совместное распределение вероятностей для последовательности наблюдаемых состояний x_1, \dots, x_n в виде цепи Маркова для скрытых состояний z_1, \dots, z_n в форме (11.48)

Используя критерий d -разделения, можно убедиться, что в модели пространства состояний всегда есть путь, соединяющий любые две наблюдаемые переменные x_n и x_m через латентные переменные, и что этот путь никогда не блокируется. Таким образом, прогнозируемое распределение $p(x_{n+1} | x_1, \dots, x_n)$ для наблюдения x_{n+1} с учетом всех предыдущих наблюдений не обладает свойствами условной независимости, и поэтому прогнозы для x_{n+1} зависят от всех предыдущих наблюдений. Следовательно, наблюдаемые переменные не удовлетворяют свойству Маркова ни при каком порядке.

Существуют две важные модели для последовательных данных, которые описываются таким графом. Если латентные переменные дискретны, то получится скрытая марковская модель (*hidden Markov model*) (Elliott, Aggoun and Moore, 1995). Обратите внимание, что наблюдаемые переменные в скрытой марковской модели могут быть дискретными или непрерывными, и для их моделирования может использоваться множество различных условных распределений. Если и скрытые, и наблюдаемые переменные являются гауссовыми (с линейной гауссовой зависимостью условных распределений от их родителей), то получится линейная динамическая система, также известная как фильтр Калмана (*Kalman filter*) (Zarchan and Musoff, 2005). Скрытые марковские модели и фильтры Калмана, а также алгоритмы их обучения подробно описаны в работе (Bishop, 2006). Такие модели можно сделать значительно более гибкими за счет замены простых дискретных таблиц вероятностей или линейных гауссовых распределений, используемых для определения $p(x_n | z_n)$, глубокими нейронными сетями.

Упражнения

- 11.1** (*) Путем маргинализации переменных по порядку докажите, что представление (11.6) для совместного распределения направленного графа является правильно нормализованным, если нормализовано каждое из условных распределений.
- 11.2** (*) Докажите, что свойство отсутствия циклов в направленном графе следует из утверждения, что существует такая упорядоченная нумерация узлов, при которой для каждого узла не существует связей, ведущих к узлу с меньшим номером.

- 11.3** (**) Рассмотрим три бинарные переменные $a, b, c \in \{0, 1\}$, имеющие совместное распределение, приведенное в табл. 11.1. Докажите с помощью прямой оценки, что это распределение обладает тем свойством, что a и b являются маргинально зависимыми, так что $p(a, b) \neq p(a)p(b)$, но становятся независимыми при условии c , так что $p(a, b|c) = p(a|c)p(b|c)$ как для $c = 0$, так и для $c = 1$.

a	b	c	$p(a, b, c)$
0	0	0	0,192
0	0	1	0,144
0	1	0	0,048
0	1	1	0,216
1	0	0	0,192
1	0	1	0,064
1	1	0	0,048
1	1	1	0,096

ТАБЛИЦА 11.1 Совместное распределение по трем бинарным переменным

- 11.4** (**) Оцените распределения $p(a)$, $p(b|c)$ и $p(c|a)$, соответствующие совместному распределению в табл. 11.1. Докажите путем непосредственной оценки, что $p(a, b, c) = p(a)p(c|a)p(b|c)$. Нарисуйте соответствующий направленный граф.
- 11.5** (*) Для модели на рис. 11.6, показано, что число параметров, необходимых для задания условного распределения $p(y|x_1, \dots, x_M)$, где $x_i \in \{0, 1\}$, можно сократить с 2^M до $M + 1$, используя представление логистической сигмоидной функции из (11.8). Альтернативное представление (Pearl, 1988) имеет вид:

$$p(y = 1|x_1, \dots, x_M) = 1 - (1 - \mu_0) \prod_{i=1}^M (1 - \mu_i)^{x_i}, \quad (11.49)$$

где параметры μ_i представляют собой вероятности $p(x_i = 1)$, а μ_0 – это дополнительный параметр, удовлетворяющий условию $0 \leq \mu_0 \leq 1$. Условное распределение (11.49) известно как «шумное ИЛИ» (noisy-OR). Докажите, что его можно интерпретировать как «мягкую» (вероятностную) форму логической функции ИЛИ (т. е. функции, которая дает $y = 1$ всякий раз, когда хотя бы один из $x_i = 1$). Проанализируйте интерпретацию μ_0 .

- 11.6** (**) Начиная с определения (11.9) для условных распределений, выведите рекуррентное соотношение (11.12) по среднему значению совместного распределения для линейной гауссовой модели.
- 11.7** (**) Начиная с определения (11.9) для условных распределений, выведите рекуррентное соотношение (11.13) для ковариационной матрицы совместного распределения для линейной гауссовой модели.

- 11.8** (**) Докажите, что число параметров в ковариационной матрице полностью связной линейной гауссовой модели графа по D переменным, определяемой по (11.9), равно $D(D + 1)/2$.
- 11.9** (**) Используя рекуррентные соотношения (11.12) и (11.13), докажите, что среднее значение и ковариация совместного распределения для графа, изображенного на рис. 11.7, заданы в (11.14) и (11.15) соответственно.
- 11.10** (*) Убедитесь, что совместное распределение по набору векторных переменных, заданных линейной гауссовой моделью, в которой каждому узлу соответствует распределение вида (11.16), само является гауссовым.
- 11.11** (*) Докажите, что из $a \perp\!\!\!\perp b, c \mid d$ следует $a \perp\!\!\!\perp b \mid d$.
- 11.12** (*) Используя критерий d -разделения, докажите, что условное распределение для узла x в направленном графе, обусловленное всеми узлами в марковском покрытии, не зависит от остальных переменных в графе.
- 11.13** (*) Рассмотрим направленный граф, показанный на рис. 11.32, в котором ни одна из переменных не является наблюдаемой. Докажите, что $a \perp\!\!\!\perp b \mid \emptyset$. Предположим, что теперь мы наблюдаем переменную d . Докажите, что в общем случае $a \not\perp\!\!\!\perp b \mid d$.
- 11.14** (**) Рассмотрим пример топливной системы автомобиля, показанный на рис. 11.20, и предположим, что вместо непосредственного наблюдения за состоянием указателя уровня топлива G его видит водитель D , который сообщает нам показания указателя. В этом сообщении говорится, что датчик показывает либо полный бак $D = 1$, либо пустой $D = 0$. Этот водитель немного недостоверен, что выражается в следующих вероятностях:

$$p(D = 1 \mid G = 1) = 0,9, \quad (11.50)$$

$$p(D = 0 \mid G = 0) = 0,9. \quad (11.51)$$

Предположим, что водитель сообщил о пустом бензобаке, другими словами, наблюдается $D = 0$. Оцените вероятность того, что бак пуст, учитывая только это наблюдение. Точно так же оцените соответствующую вероятность, учитывая также, что аккумулятор разряжен, и обратите внимание, что эта вторая вероятность меньше. Проанализируйте смысл этого результата и соотнесите его с рис. 11.32.

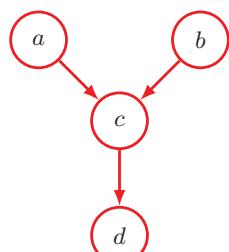


РИС. 11.32 Пример графической модели для изучения свойств условной независимости пути «голова к голове» $a-c-b$, когда наблюдается потомок c , а именно узел d

- 11.15** (**) Предположим, для обучения модели наивного Байеса с предположением (11.37) будет использоваться метод максимального правдоподобия. Предположим, что каждая из условных плотностей классов $p(\mathbf{x}^{(l)} | \mathcal{C}_k)$ управляет своими независимыми параметрами $\mathbf{w}^{(l)}$. Докажите, что решение с максимальным правдоподобием состоит в подгонке каждой из условных плотностей с помощью соответствующих векторов наблюдаемых данных $\mathbf{x}_1^{(l)}, \dots, \mathbf{x}_N^{(l)}$ путем максимизации правдоподобия относительно данных о метках соответствующего класса, а затем установки классовых априорных вероятностей $p(\mathcal{C}_k)$ на долю обучающих точек данных в каждом классе.
- 11.16** (**) Рассмотрим совместное распределение вероятностей (11.44), соответствующее направленному графу на рис. 11.29. Используя правила суммы и произведения вероятностей, докажите, что это совместное распределение удовлетворяет свойству условной независимости (11.45) для $n = 2, \dots, N$. Аналогично докажите, что марковская модель второго порядка, описываемая совместным распределением (11.46), удовлетворяет свойству условной независимости:
- $$p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}) = p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{x}_{n-2}) \quad (11.52)$$
- для $n = 3, \dots, N$.
- 11.17** (*) Используйте d-разделение, как обсуждалось в разделе 11.2, чтобы доказать, что марковская модель, показанная на рис. 11.29 и имеющая в общей сложности N узлов, удовлетворяет свойствам условной независимости (11.45) для $n = 2, \dots, N$. Точно так же докажите, что модель, описываемая графом на рис. 11.30, в котором всего N узлов, удовлетворяет свойствам условной независимости (11.52) для $n = 3, \dots, N$.
- 11.18** (*) Рассмотрим модель Маркова второго порядка, описываемую графом на рис. 11.30. Объединив соседние пары переменных, докажите, что он может быть выражен как марковский процесс первого порядка для новых переменных.
- 11.19** (*) Используйте d-разделение, докажите, что распределение $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ наблюдаемых данных для модели пространства состояний, представленной направленным графом на рис. 11.31, не удовлетворяет никаким свойствам условной независимости и, следовательно, не проявляет свойства Маркова ни при каком конечном порядке.

Глава 12

Трансформеры

Трансформеры, или *преобразователи* (*transformers*), представляют собой одну из наиболее важных разработок в области глубокого обучения. Они основаны на концепции обработки данных, называемой *вниманием* (*attention*), которая позволяет сети присваивать различные веса различным входным сигналам с весовыми коэффициентами, которые сами зависят от входных значений, тем самым улавливая мощные индуктивные смещения, связанные с последовательными и другими формами данных.

Эти модели получили название «трансформеры» по той причине, что они преобразуют набор векторов в определенном пространстве представлений в соответствующий набор векторов той же размерности в каком-либо новом пространстве. Цель преобразования состоит в том, чтобы новое пространство имело более содержательное внутреннее представление, которое лучше подходит для решения последующих задач. Входные данные для трансформера могут принимать форму неструктурированных наборов векторов, упорядоченных последовательностей или более общих представлений, что обеспечивает широкую применимость трансформеров.

Трансформеры были первоначально представлены в сфере обработки естественного языка (natural language processing, NLP), где «естественным» языком считается английский или мандаринский диалект китайского языка. Они значительно превзошли предыдущие передовые подходы, основанные на использовании рекуррентных нейронных сетей (recurrent neural networks, RNN). Впоследствии выяснилось, что трансформеры показывают отличные результаты и во многих других областях. Например, в задачах обработки изображений визуальные трансформеры часто превосходят CNN, а мульти-модальные трансформеры, объединяющие несколько типов данных, таких как текст, изображения, аудио и видео, относятся к числу наиболее мощных моделей глубокого обучения.

Одним из главных преимуществ трансформеров является высокая эффективность трансферного обучения: модель трансформера может быть обучена на большом объеме данных, а затем ее можно использовать для решения множества последующих задач с помощью той или иной формы тонкой настройки. Крупномасштабная модель, которая впоследствии может быть адаптирована для решения множества различных задач, называется *базовой моделью* (*foundation model*). Кроме того, трансформеры можно обучать само-

стоятельно, используя немаркированные данные, что особенно эффективно для языковых моделей, поскольку трансформеры способны обрабатывать огромные объемы текстов из интернета и других источников. Гипотеза масштабной инвариантности (*scaling hypothesis*) гласит, что увеличение масштаба модели, измеряемого количеством обучаемых параметров, и обучение на соразмерно большом наборе данных позволяет добиться значительного улучшения производительности даже без изменения архитектуры. Более того, трансформер особенно хорошо подходит для аппаратных средств массивно-параллельной обработки данных, таких как *графические процессоры* (GPU), что позволяет обучать чрезвычайно большие нейросетевые языковые модели, имеющие порядка триллиона (10^{12}) параметров, за разумное время. Такие модели обладают необычайно широкими возможностями и демонстрируют явные признаки независимых свойств (*emergent properties*), которые уже были названы ранними признаками общего искусственного интеллекта (Bubeck et al., 2023).

Архитектура трансформера может показаться новичку сложной или даже обескураживающей, поскольку она включает в себя множество взаимодействующих компонентов, при этом выбор вариантов построения может показаться произвольным. Поэтому в этой главе ставится задача дать исчерпывающее пошаговое введение во все ключевые идеи, лежащие в основе трансформеров, и предоставить четкие формулировки для обоснования выбора различных элементов. Сначала будет описана архитектура трансформеров, затем основное внимание будет уделено обработке естественного языка, после чего будут рассмотрены другие области применения.

12.1. Внимание

Фундаментальной концепцией, лежащей в основе трансформеров, является *внимание* (*attention*). Изначально эта концепция была разработана (см. раздел 12.2.5) в качестве усовершенствования RNN для машинного перевода (Bahdanau, Cho and Bengio, 2014). Однако позже в работе (Vaswani et al., 2017) было показано, что можно добиться значительно более высокой производительности, если отказаться от рекуррентной структуры и сосредоточиться исключительно на механизме внимания. Сегодня трансформеры с использованием принципа внимания полностью вытеснили RNN почти во всех приложениях.

В качестве примера использования внимания рассмотрим естественный язык, хотя эта технология имеет гораздо более широкое применение. Рассмотрим два следующих предложения:

I swam across the river to get to the other **bank** (я переплыл реку, чтобы попасть на другой берег).

I walked across the road to get cash from the **bank** (я перешел дорогу, чтобы получить наличные в банке).

Здесь слово «bank» имеет разные значения в двух предложениях. Однако это можно определить, только обратив внимание на контекст, создаваемый другими словами в последовательности. Кроме того, можно заметить, что некоторые слова более важны, чем другие, для определения интерпретации слова «bank». В первом предложении слова «swam» (плыл) и «river» (река) наиболее весомо свидетельствуют о том, что «bank» относится к берегу реки, в то время как во втором предложении слово «cash» (наличные) является весомым индикатором того, что «bank» относится к финансовому учреждению. Таким образом, для определения правильной интерпретации слова «bank» нейронной сети, обрабатывающей такое предложение, необходимо обратить внимание на конкретные слова из остальной части последовательности, т. е. опираться на них в большей степени. Эта концепция внимания проиллюстрирована на рис. 12.1.

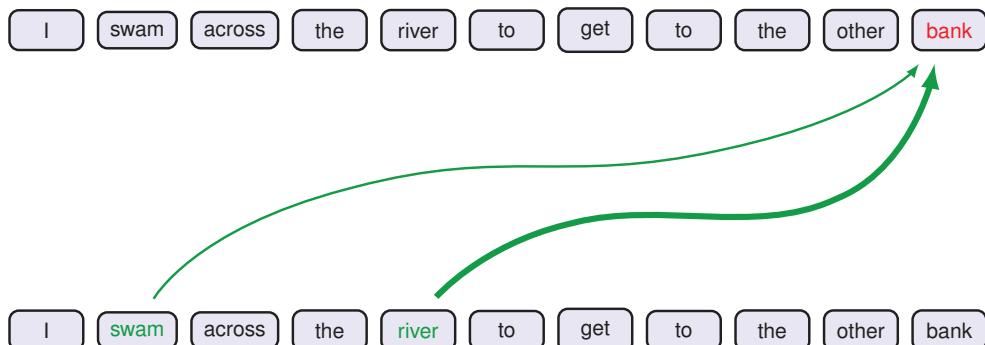


РИС. 12.1 Схематичная иллюстрация внимания, где на интерпретацию слова «bank» влияют слова «river» и «swam», а толщина каждой линии указывает степень ее влияния

Более того, теперь становится ясно, что конкретные места, которым следует уделять больше внимания, зависят от самой входной последовательности: в первом предложении важны второе и пятое слова, а во втором — восьмое. В стандартной нейронной сети различные входы будут влиять на выход в разной степени в зависимости от значений весов, на которые умножаются эти входы. Однако после обучения сети эти веса и связанные с ними входы остаются неизменными. В отличие от этого при работе с техникой внимания применяются весовые коэффициенты, значения которых зависят от конкретных входных данных. На рис. 12.2 показаны весовые коэффициенты внимания из участка сети трансформера, обученного на естественном языке.

Когда речь пойдет об обработке естественного языка, будет показано, как вложение слов может использоваться для преобразования слов в векторы во вложенном пространстве. Эти векторы могут быть использованы в качестве входных данных для последующей обработки нейронной сетью. Такие вложения отражают элементарные семантические свойства, например, путем соотнесения слов с похожими значениями с близкими местами во вложенном пространстве. Одной из характеристик таких вложений явля-

ется то, что данное слово всегда соотносится с одним и тем же вектором вложения. Трансформер можно рассматривать как более обширную форму вложения, где данный вектор отображается на место, зависящее от других векторов в последовательности. Таким образом, вектор, представляющий «bank» в примере выше, может быть сопоставлен с разными местами в новом пространстве вложения для двух разных предложений. Например, в первом предложении преобразованное представление может поместить «bank» рядом с «water» в пространстве вложения, тогда как во втором предложении преобразованное представление может поместить его рядом с «money».

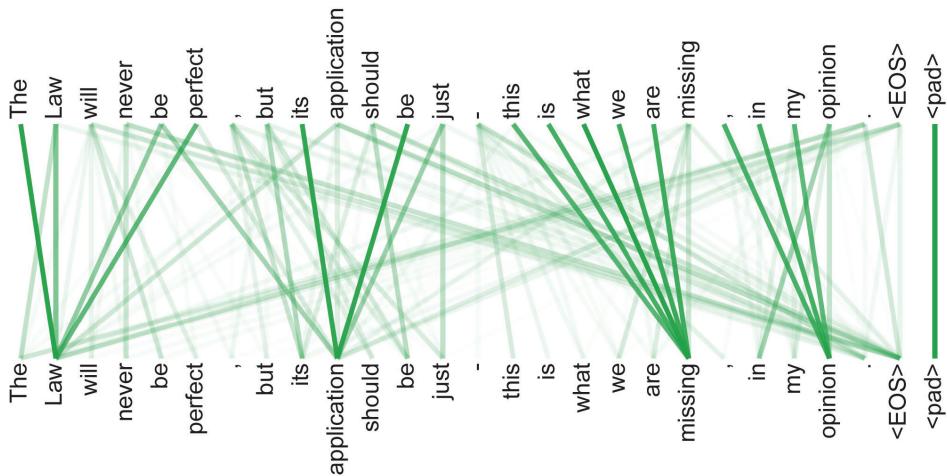


РИС. 12.2 Пример обучаемых весов внимания. [Из (Vaswani et al., 2017) с разрешения авторов]

В качестве примера внимания рассмотрим моделирование белков. Белок можно рассматривать как одномерную последовательность молекулярных элементов, называемых аминокислотами. Потенциально белок может состоять из сотен или тысяч таких элементов, каждый из которых задается одной из 22 возможных опций. В живой клетке белок собирается в трехмерную структуру, в которой аминокислоты, сильно удаленные друг от друга в одномерной последовательности, могут физически сближаться в трехмерном пространстве и, таким образом, взаимодействовать между собой (рис. 1.2). Модели трансформеров позволяют этим удаленным друг от друга аминокислотам «обращать внимание» друг на друга, тем самым значительно повышая точность моделирования их трехмерной структуры (Vig et al., 2020).

12.1.1. Обработка трансформеров

Входные данные для трансформера – это набор векторов $\{x_n\}$ размерности D , где $n = 1, \dots, N$. Эти векторы данных называют *лексемами (tokens)*, где лексема может, например, соответствовать слову в предложении, пятну в изображе-

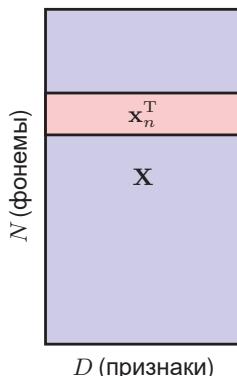
ний или аминокислот в белке. Элементы x_{ni} лексем называются *признаками* (*features*). Позже будет описано построение таких лексемных векторов для данных естественного языка и изображений. Эффективное свойство трансформеров заключается в том, что для работы с различными типами данных не нужно разрабатывать новую архитектуру нейронной сети, а можно просто объединить переменные данные в общий набор лексем.

Прежде чем четко понять принцип работы трансформера, необходимо точно определиться с обозначениями. Согласно стандартным правилам векторы данных объединяются в матрицу \mathbf{X} размером $N \times D$, в которой n -я строка представляет собой вектор лексем \mathbf{x}_n^T , а $n = 1, \dots, N$ обозначают строки, как показано на рис. 12.3. Обратите внимание, что эта матрица представляет один набор входных лексем, а для большинства приложений потребуется набор данных, содержащий множество наборов лексем, например независимые отрывки текста, где каждое слово представлено одной лексемой. Фундаментальный строительный блок трансформера – это функция, которая принимает на вход матрицу данных и создает на выходе преобразованную матрицу $\tilde{\mathbf{X}}$ той же размерности. Этую функцию можно записать в виде

$$\tilde{\mathbf{X}} = \text{TransformerLayer}[\mathbf{X}]. \quad (12.1)$$

Затем можно последовательно применять несколько трансформирующих слоев (TransformerLayer) для построения глубоких сетей, способных обучаться насыщенным внутренним представлениям. Каждый слой-трансформер содержит свои собственные веса и смещения, которые можно обучить с помощью градиентного спуска с использованием соответствующей функции стоимости (см. раздел 12.3), что будет подробно рассмотрено далее в этой главе.

РИС. 12.3 Структура матрицы данных \mathbf{X} размерности $N \times D$, в которой строка n представляет транспонированный вектор данных \mathbf{x}_n^T



Один слой трансформера включает в себя два этапа. Первый этап, реализующий механизм внимания, смешивает соответствующие признаки из разных векторов маркеров по столбцам матрицы данных, в то время как второй этап работает с каждой строкой независимо и преобразует при-

знаки внутри каждого вектора маркеров. Для начала рассмотрим работу механизма внимания.

12.1.2. Коэффициенты внимания

Предположим, что во вложенном пространстве имеется набор входных лексем $\mathbf{x}_1, \dots, \mathbf{x}_N$, и необходимо сопоставить его с другим набором $\mathbf{y}_1, \dots, \mathbf{y}_N$ с тем же количеством лексем, но в новом вложенном пространстве, которое отражает более насыщенную семантическую структуру. Рассмотрим конкретный выходной вектор \mathbf{y}_n . Значение \mathbf{y}_n будет зависеть не только от соответствующего входного вектора \mathbf{x}_n , но и от всех векторов $\mathbf{x}_1, \dots, \mathbf{x}_N$ в наборе. При использовании механизма внимания эта зависимость будет усиливаться для тех входов \mathbf{x}_m , которые особенно важны для определения модифицированного представления \mathbf{y}_n . Простой способ добиться этого – определить каждый выходной вектор \mathbf{y}_n как линейную комбинацию входных векторов $\mathbf{x}_1, \dots, \mathbf{x}_N$ с весовыми коэффициентами a_{nm} :

$$\mathbf{y}_n = \sum_{m=1}^N a_{nm} \mathbf{x}_m, \quad (12.2)$$

где a_{nm} называются *весами внимания* (*attention weight*). Для входных лексем, слабо влияющих на выход \mathbf{y}_n , коэффициенты должны быть близки к нулю, а для входных лексем, оказывающих наибольшее влияние на выход, должны быть наибольшими. Поэтому во избежание ситуаций, когда один коэффициент может стать большим и положительным, а другой коэффициент компенсирует это большим и отрицательным, необходимо ограничить коэффициенты неотрицательными значениями. Кроме того, необходимо учитывать, что если выход уделяет больше внимания определенному входу, то это произойдет за счет меньшего внимания к другим входам. Поэтому нужно ограничить сумму коэффициентов единицей. Таким образом, весовые коэффициенты должны удовлетворять следующим двум ограничениям:

$$a_{nm} \geq 0, \quad (12.3)$$

$$\sum_{m=1}^N a_{nm} = 1. \quad (12.4)$$

В совокупности это означает (см. упражнение 12.1), что каждый коэффициент лежит в интервале $0 \leq a_{nm} \leq 1$, а значит, коэффициенты определяют «разбиение единства». Для специального случая $a_{nn} = 1$ следует, что $a_{nm} = 0$ для $n \neq m$, и поэтому $\mathbf{y}_n = \mathbf{x}_n$, так что выходной вектор при преобразовании не изменяется. В более общем случае выходной \mathbf{y}_n представляет собой смесь входных векторов, где некоторым входам придается больший вес, чем другим.

Обратите внимание, что для каждого выходного вектора \mathbf{y}_n имеется свой набор коэффициентов, и ограничения (12.3) и (12.4) применяются отдельно для каждого значения n . Эти коэффициенты a_{nm} зависят от входных данных, и далее будет описан способ их вычисления.

12.1.3. Самовнимание

Следующий вопрос – определение коэффициентов a_{nm} . Прежде чем обсуждать этот вопрос подробно, полезно для начала ввести некоторую терминологию, взятую из области информационных поисковых систем. Рассмотрим проблему выбора фильма для просмотра в онлайн-сервисе потокового кино. Один из подходов заключается в том, чтобы связать каждый фильм со списком атрибутов, описывающих такие вещи, как жанр (комедия, боевик и т. д.), имена ведущих актеров, продолжительность фильма и т. д. Затем пользователь может искать в каталоге фильм в соответствии со своими предпочтениями. Процесс можно автоматизировать, закодировав атрибуты каждого фильма в векторе под названием *ключ* (*key*). Сам соответствующий файл фильма называют *значением* (*value*). Аналогичным образом пользователь может предоставить свой собственный вектор значений для желаемых атрибутов, который и называется *запросом* (*query*). Затем потоковый сервис может сравнить вектор запроса со всеми векторами ключей для поиска наилучшего соответствия и отправить соответствующий фильм пользователю в виде файла-значения. Можно считать, что пользователь «посещает» тот фильм, ключ которого наиболее точно соответствует его запросу. Это можно считать разновидностью *жесткого внимания* (*hard attention*), при котором возвращается единственный вектор значений. Для трансформера это свойство обобщается до *мягкого внимания* (*soft attention*), когда для измерения степени соответствия между запросами и ключами используются непрерывные переменные, а затем с их помощью взвешивается влияние векторов значений на выходы. Это также гарантирует, что функция трансформера является дифференцируемой и, следовательно, может быть обучена методом градиентного спуска.

Продолжая аналогию с информационным поиском, можно рассматривать каждый из входных векторов \mathbf{x}_n как вектор значений, который будет использоваться для создания выходных лексем. Кроме того, вектор \mathbf{x}_n можно использовать непосредственно как ключевой вектор для входной лексемы n . Это аналогично использованию самого фильма для краткого описания его характеристик. Наконец, можно использовать \mathbf{x}_m в качестве вектора запроса для выходного y_m , который затем можно сравнить с каждым из ключевых векторов. Чтобы понять, насколько лексема, представленная \mathbf{x}_n , должна соответствовать лексеме, представленной \mathbf{x}_m , нужно оценить степень сходства этих векторов. Одна из простых мер сходства – взять их точечное произведение $\mathbf{x}_n^T \mathbf{x}_m$. Чтобы наложить ограничения (12.3) и (12.4), можно определить весовые коэффициенты a_{nm} , используя функцию *softmax* (см. раздел 5.3) для преобразования точечных произведений:

$$a_{nm} = \frac{\exp(\mathbf{x}_n^T \mathbf{x}_m)}{\sum_{m'=1}^N \exp(\mathbf{x}_n^T \mathbf{x}_{m'}^T)}. \quad (12.5)$$

Обратите внимание, что в данном случае нет вероятностной интерпретации функции *softmax*, и она просто используется для соответствующей нормализации весов внимания.

Таким образом, каждый входной вектор \mathbf{x}_n преобразуется в соответствующий выходной вектор \mathbf{y}_n путем взятия линейной комбинации входных векторов вида (12.2), в котором вес a_{nm} , применяемый к входному вектору \mathbf{x}_m , задается функцией softmax (12.5), определяемой точечным произведением $\mathbf{x}_n^T \mathbf{x}_m$ между запросом \mathbf{x}_n для входа n и ключом \mathbf{x}_m , связанным с входом m . Обратите внимание, что если все входные векторы ортогональны, то каждый выходной вектор просто равен соответствующему входному вектору, так что $\mathbf{y}_m = \mathbf{x}_m$ для $m = 1, \dots, N$ (см. упражнение 12.3).

Можно записать (12.2) в матричной системе с помощью матрицы данных \mathbf{X} и аналогичной выходной матрицы \mathbf{Y} размером $N \times D$, строки которой заданы \mathbf{y}_m , так что

$$\mathbf{Y} = \text{Softmax}[\mathbf{X}\mathbf{X}^T\mathbf{X}], \quad (12.6)$$

где $\text{Softmax}[\mathbf{L}]$ – это оператор, который берет экспоненту каждого элемента матрицы \mathbf{L} и затем нормализует каждую строку по отдельности, чтобы сумма равнялась единице. В дальнейшем для ясности описания будут использоваться матричные обозначения.

Этот процесс называется *самовниманием (self-attention)*, поскольку для определения запросов, ключей и значений используется одна и та же последовательность. Варианты этого механизма внимания будут встречаться далее в этой главе. Кроме того, поскольку мера сходства между векторами запросов и ключей задается точечным произведением, этот процесс также известен как *самовнимание по точечному произведению (dot-product self-attention)*.

12.1.4. Сетевые параметры

В таком виде преобразование входных векторов $\{\mathbf{x}_n\}$ в выходные векторы $\{\mathbf{y}_n\}$ является фиксированным и не способно обучаться на данных, поскольку не имеет настраиваемых параметров. Более того, каждое из значений признаков в векторе лексем \mathbf{x}_n играет одинаковую роль в определении коэффициентов внимания, в то время как для определения сходства лексем желательно, чтобы сеть имела возможность уделять некоторым признакам больше внимания, чем другим. Обе проблемы можно решить путем определения модифицированных векторов признаков, заданных линейным преобразованием исходных векторов в виде

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{U}, \quad (12.7)$$

где \mathbf{U} – это матрица $D \times D$ обучаемых весовых параметров, аналогичная «слою» в стандартной нейронной сети. Это дает модифицированное преобразование вида

$$\mathbf{Y} = \text{Softmax}[\tilde{\mathbf{X}}\mathbf{U}\tilde{\mathbf{X}}^T\mathbf{U}]\tilde{\mathbf{X}}\mathbf{U}. \quad (12.8)$$

Несмотря на большую гибкость этого преобразования, оно обладает тем свойством, что матрица

$$\mathbf{X} \mathbf{U} \mathbf{U}^T \mathbf{X}^T \quad (12.9)$$

является симметричной, тогда как в механизме внимания должна присутствовать значительная асимметрия. Например, можно ожидать, что слово «chisel» (зубило) будет устойчиво ассоциироваться с «tool» (инструмент), поскольку каждое зубило – это инструмент, в то время как слово «tool» (инструмент) будет лишь слабо ассоциироваться с «chisel» (зубило), поскольку существует множество других видов инструментов, помимо зубил. Несмотря на то что в результате использования функции softmax полученная матрица весов внимания сама по себе не является симметричной, с ее помощью можно создать гораздо более гибкую модель, позволив запросам и ключам иметь независимые параметры. Кроме того, форма (12.8) использует одну и ту же матрицу параметров \mathbf{U} для определения векторов значений и коэффициентов внимания, что также представляется излишним ограничением.

Преодолеть эти ограничения можно с помощью определения отдельных матриц запросов, ключей и значений, каждая из которых имеет свои независимые линейные преобразования:

$$\mathbf{Q} = \mathbf{X} \mathbf{W}^{(q)}, \quad (12.10)$$

$$\mathbf{K} = \mathbf{X} \mathbf{W}^{(k)}, \quad (12.11)$$

$$\mathbf{V} = \mathbf{X} \mathbf{W}^{(v)}, \quad (12.12)$$

где весовые матрицы $\mathbf{W}^{(q)}$, $\mathbf{W}^{(k)}$ и $\mathbf{W}^{(v)}$ представляют собой параметры, которые будут обучаться в процессе обучения архитектуры конечного трансформера. Здесь матрица $\mathbf{W}^{(k)}$ имеет размерность $D \times D_k$, где D_k – это длина ключевого вектора. Матрица $\mathbf{W}^{(q)}$ должна иметь ту же размерность $D \times D_k$, что и $\mathbf{W}^{(k)}$, чтобы была возможность сформировать точечные произведения между векторами запроса и ключа. Как правило, выбирается $D_k = D$. Аналогичным образом $\mathbf{W}^{(v)}$ – это матрица размером $D \times D_v$, где D_v определяет размерность выходных векторов. Если задать $D_v = D$, чтобы выходное представление имело ту же размерность, что и входное, это облегчит включение остаточных связей, которые будут рассмотрены позже (см. раздел 12.1.7). Кроме того, несколько слоев трансформера могут быть наложены друг на друга, если каждый слой имеет ту же размерность. Затем (12.6) можно обобщить, чтобы получить

$$\mathbf{Y} = \text{Softmax}[\mathbf{Q} \mathbf{K}^T] \mathbf{V}, \quad (12.13)$$

где $\mathbf{Q} \mathbf{K}^T$ имеет размерность $N \times N$, а матрица \mathbf{Y} имеет размерность $N \times D_v$. Вычисление матрицы $\mathbf{Q} \mathbf{K}^T$ показано на рис. 12.4. Здесь входной сигнал \mathbf{X} отдельно преобразуется с помощью (12.10) и (12.11) для получения матрицы запросов \mathbf{Q} и матрицы ключей \mathbf{K} соответственно, и затем они перемножаются.

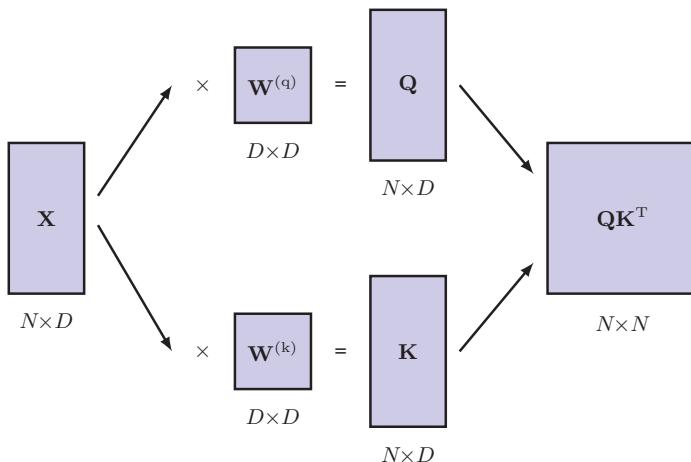


РИС. 12.4 Иллюстрация оценки матрицы QK^T , определяющей коэффициенты внимания в трансформере

Оценка матрицы Y представлена на рис. 12.5. Запись в выделенной позиции выходной матрицы Y получается из точечного произведения выделенных строк и столбца матриц Softmax QK^T и V соответственно.

$$\begin{matrix} Y \\ \vdots \end{matrix}_{N \times D_v} = \text{Softmax} \left\{ \begin{matrix} \vdots \\ QK^T \\ \vdots \end{matrix}_{N \times N} \right\} \times \begin{matrix} \vdots \\ V \\ \vdots \end{matrix}_{N \times D_v}$$

РИС. 12.5 Иллюстрация оценки выхода слоя внимания с учетом матриц запросов, ключей и значений Q , K и V соответственно

На практике параметры смещения также могут быть включены в эти линейные преобразования. Однако параметры смещения можно включить в весовые матрицы, как это делается в стандартных нейронных сетях (см. раздел 6.2.1), путем дополнения матрицы данных X дополнительным столбцом единиц и дополнения весовых матриц дополнительной строкой параметров для представления смещений. В дальнейшем во избежание путаницы в обозначениях параметры смещения будут рассматриваться как неявные.

По сравнению с обычной нейронной сетью сигнальные пути имеют мультиплексивные зависимости между значениями активации. Если в обычных сетях активации умножаются на фиксированные веса, то здесь активации умножаются на коэффициенты внимания, зависящие от данных. Это означает, например, что если один из коэффициентов внимания для конкретного выбора входного вектора близок к нулю, то результирующий путь сигнала бу-

дет игнорировать соответствующий входящий сигнал, что, соответственно, не окажет никакого влияния на выходы сети. В отличие от этого, если стандартная нейронная сеть учится игнорировать определенный входной сигнал или переменную скрытого блока, она делает это для всех входных векторов.

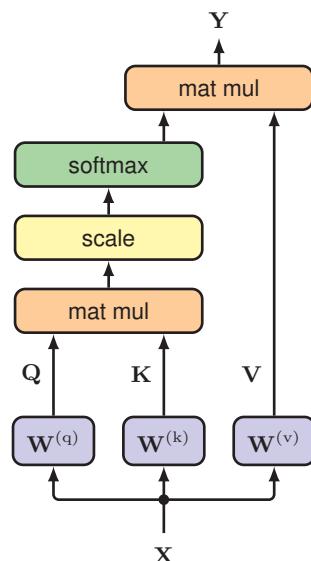
12.1.5. Масштабируемое самовнимание

Есть одно последнее уточнение, которое стоит отметить в работе слоя самовнимания. Вспомним, что градиенты функции softmax становятся экспоненциально малыми для входов большой величины, как это происходит с функциями активации \tanh или логистической сигмоидной функцией. Чтобы этого не произошло, перед применением функции softmax необходимо изменить масштаб произведения векторов запроса и ключа. Для определения подходящего масштаба учтите, что, если бы элементы векторов запроса и ключа были независимыми случайными числами с нулевым средним значением и единичной дисперсией (см. упражнение 12.4), то дисперсия точечного произведения была бы равна D_k . Поэтому при нормализации аргумента softmax используется стандартное отклонение, равное квадратному корню из D_k , так что выход слоя внимания принимает вид:

$$\mathbf{Y} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \equiv \text{Softmax}\left[\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}}\right]\mathbf{V}. \quad (12.14)$$

Это называется *масштабированным точечным самовниманием (scaled dot-product self-attention)* и представляет собой конечную форму слоя нашей нейронной сети самовнимания. Структура этого слоя кратко представлена на рис. 12.6 и в алгоритме 12.1.

РИС. 12.6 Информационный поток в масштабированном точечном слое самовнимания нейронной сети. Здесь «mat mul» означает умножение матрицы, а «scale» – это нормализация аргумента для softmax с помощью $\sqrt{D_k}$. Эта структура представляет собой единую «голову» внимания



АЛГОРИТМ 12.1 *Масштабированное точечное самовнимание*

Input: множество лексем $\mathbf{X} \in \mathbb{R}^{N \times D} : \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
 Матрицы весов $\{\mathbf{W}^{(q)}, \mathbf{W}^{(k)}\} \in \mathbb{R}^{D \times D_k}$ и $\mathbf{W}^{(v)} \in \mathbb{R}^{D \times D_v}$

Output: $\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{N \times D_v} : \{\mathbf{y}_1, \dots, \mathbf{y}_N\}$

$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$ // вычисление запросов $\mathbf{Q} \in \mathbb{R}^{N \times D_k}$

$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$ // вычисление ключей $\mathbf{K} \in \mathbb{R}^{N \times D_k}$

$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$ // вычисление значений $\mathbf{V} \in \mathbb{R}^{N \times D_v}$

return $\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \equiv \text{Softmax} \left[\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}} \right] \mathbf{V}$

12.1.6. Многоголовое внимание

Описанный ранее слой внимания позволяет выходным векторам принимать участие в зависящих от данных паттернах входных векторов и называется *головой внимания* (*attention head*). Однако может существовать несколько моделей внимания, которые актуальны в одно и то же время. Например, в естественном языке одни паттерны могут быть связаны с временем, а другие – со словарным запасом. Использование одной головы внимания может привести к усреднению этих эффектов. Вместо этого можно параллельно использовать несколько голов внимания. Они состоят из идентично структурированных копий одной головы с независимыми обучаемыми параметрами, которые управляют вычислением матриц запросов, ключей и значений. Это аналогично использованию нескольких различных фильтров в каждом слое сверточной сети.

Предположим, имеется H голов, индексируемых $h = 1, \dots, H$ в виде

$$\mathbf{H}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h), \quad (12.15)$$

где $\text{Attention}(\cdot, \cdot, \cdot)$ задается в (12.14), и для каждой головы определены отдельные матрицы запросов, ключей и значений с помощью

$$\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^{(q)}, \quad (12.16)$$

$$\mathbf{K}_h = \mathbf{X}\mathbf{W}_h^{(k)}, \quad (12.17)$$

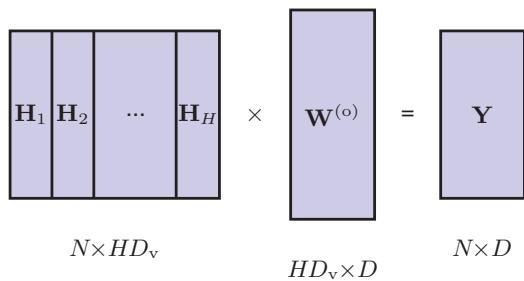
$$\mathbf{V}_h = \mathbf{X}\mathbf{W}_h^{(v)}. \quad (12.18)$$

Сначала головы объединяются в одну матрицу, а затем результат линейно преобразуется с помощью матрицы $\mathbf{W}^{(o)}$ для получения комбинированного вывода в виде

$$\mathbf{Y}(\mathbf{X}) = \text{Concat} [\mathbf{H}_1, \dots, \mathbf{H}_H] \mathbf{W}^{(o)}. \quad (12.19)$$

Это показано на рис. 12.7.

РИС. 12.7 Сетевая архитектура для многоголового внимания. Каждая голова состоит из структуры, показанной на рис. 12.6, и имеет свои параметры ключа, запроса и значения. Выходы голов конкatenируются, а затем линейно проецируются обратно в размерность входных данных



Каждая матрица H_h имеет размерность $N \times D_v$, поэтому конкатенированная матрица имеет размерность $N \times HD_v$. Она преобразуется линейной матрицей $W^{(o)}$ размерности $HD_v \times D$ для получения конечной выходной матрицы Y размерности $N \times D$, которая совпадает с исходной входной матрицей X . Элементы матрицы $W^{(o)}$ обучаются на этапе обучения вместе с матрицами запросов, ключей и значений. Обычно D_v выбирается равной D/H , чтобы результирующая конкатенированная матрица имела размерность $N \times D$. Многоголовое внимание кратко описано в алгоритме 12.2, а поток информации в многоголовом слое внимания показан на рис. 12.8.

АЛГОРИТМ 12.2 Многоголовое внимание

Input: множество лексем $X \in \mathbb{R}^{N \times D} : \{x_1, \dots, x_N\}$
 Матрицы веса запросов $\{W_1^{(q)}, \dots, W_H^{(q)}\} \in \mathbb{R}^{D \times D}$
 Матрицы веса ключей $\{W_1^{(k)}, \dots, W_H^{(k)}\} \in \mathbb{R}^{D \times D}$
 Матрицы веса значений $\{W_1^{(v)}, \dots, W_H^{(v)}\} \in \mathbb{R}^{D \times D_v}$
 Выходная матрица весов $W^{(o)} \in \mathbb{R}^{HD_v \times D}$

Output: $Y \in \mathbb{R}^{N \times D} : \{y_1, \dots, y_N\}$

// вычисление самовнимания для каждой головы (алгоритм 12.1)

for $h = 1, \dots, H$ **do**

$$\left| \begin{array}{l} Q_h = XW_h^{(q)}, K_h = XW_h^{(k)}, V_h = XW_h^{(v)} \\ H_h = \text{Attention}(Q_h, K_h, V_h) // H_h \in \mathbb{R}^{N \times D_v} \end{array} \right.$$

end for

$H = \text{Concat}[H_1, \dots, H_N]$ // конкатенация голов

return $Y(X) = HW^{(o)}$

Обратите внимание, что приведенная выше формулировка многоголового внимания, которая соответствует используемой в исследовательской литературе, включает некоторую избыточность при последовательном перемножении матрицы $W^{(v)}$ для каждой головы и выходной матрицы $W^{(o)}$. Устранение этой избыточности позволяет записать многоголовый слой самовнимания в виде суммы вкладов от каждой из голов по отдельности (см. упражнение 12.5).

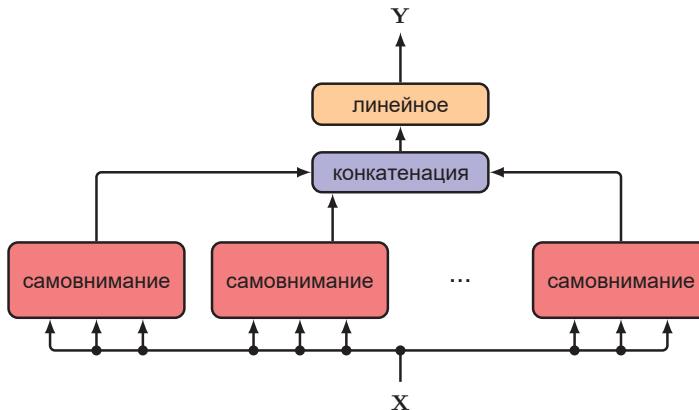


РИС. 12.8 Поток информации в многоголовом слое внимания. Соответствующее вычисление, выполняемое алгоритмом 12.2, показано на рис. 12.7

12.1.7. Слои трансформера

Многоголовое самовнимание является основным архитектурным элементом сети трансформеров. Как известно, нейронные сети получают огромное преимущество за счет глубины, поэтому желательно располагать несколько слоев самовнушения друг над другом. Для повышения эффективности обучения (см. раздел 9.5) можно ввести остаточные связи, обходящие многоголовую структуру. Для этого необходимо, чтобы размерность выхода была такой же, как размерность входа (см. раздел 7.4.3), а именно $N \times D$. Затем следует нормализация слоев (Ba, Kiros and Hinton, 2016), которая повышает эффективность обучения. Полученное преобразование можно записать в виде

$$Z = \text{LayerNorm}[Y(X) + X], \quad (12.20)$$

где Y определяется по (12.19). Иногда нормализация слоя заменяется квазинормой, при которой слой нормализации применяется до многоголового самовнимания, а не после, поскольку это может привести к более эффективной оптимизации, и в этом случае получается:

$$Z = Y(X') + X, \text{ где } X' = \text{LayerNorm}[X]. \quad (12.21)$$

В каждом случае Z вновь имеет ту же размерность $N \times D$, что и входная матрица X .

Выше уже говорилось, что механизм внимания создает линейные комбинации векторов значений, которые затем комбинируются линейно для получения выходных векторов. Кроме того, значения являются линейными функциями входных векторов, и поэтому выходы слоя внимания ограничены линейными комбинациями входов. Нелинейность вносится через веса внимания, и поэтому выходы будут нелинейно зависеть от входов через функцию softmax, но выходные векторы все равно ограничены подпространством, охватываемым входными векторами, и это ограничивает возможности слоя внимания в плане их выразительности. Гибкость трансформера можно по-

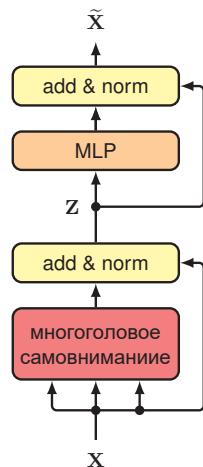
высить посредством последующей обработки выходных данных каждого слоя с помощью стандартной нелинейной нейронной сети с D входами и D выходами, обозначаемой $\text{MLP}[\cdot]$ в значении «многослойный перцептрон» (multilayer perceptron). Например, это может быть двухслойная полно связанные сеть со скрытыми элементами ReLU. Это необходимо сделать так, чтобы сохранить способность трансформера обрабатывать последовательности переменной длины. Для этого одна и та же общая сеть применяется к каждому из выходных векторов, соответствующих строкам Z . И вновь этот слой нейронной сети может быть улучшен за счет использования остаточной связи. Кроме того, он также включает нормализацию слоев, чтобы конечный выходной сигнал слоя трансформера имел вид:

$$\tilde{X} = \text{LayerNorm}[\text{MLP}[Z] + Z]. \quad (12.22)$$

Это приводит к общей архитектуре слоя трансформера, показанной на рис. 12.9 и обобщенной в алгоритме 12.3. Опять же, вместо этого можно использовать квазинорму (pre-norm), и тогда конечный результат будет выглядеть как

$$\tilde{X} = \text{MLP}(Z') + Z, \text{ где } Z' = \text{LayerNorm}[Z]. \quad (12.23)$$

РИС. 12.9 Один слой архитектуры трансформера, реализующего преобразование (12.1). Здесь «add and norm» означает остаточное соединение с последующей нормализацией слоев



АЛГОРИТМ 12.3. Уровень трансформера

Input: множество лексем $X \in \mathbb{R}^{N \times D} : \{x_1, \dots, x_N\}$

Параметры многоголового слоя самовнимания

Параметры сети прямой передачи

Output: $\tilde{X} \in \mathbb{R}^{N \times D} : \{\tilde{x}_1, \dots, \tilde{x}_N\}$

$Z = \text{LayerNorm}[Y(X) + X] // Y(X)$ из алгоритма 12.2

$\tilde{X} = \text{LayerNorm}[\text{MLP}[Z] + Z] // \text{общая нейронная сеть}$

return \tilde{X}

В типичном трансформере имеется несколько таких слоев, расположенных друг над другом. Слои, как правило, имеют идентичную структуру, хотя весовые коэффициенты и смещения между разными слоями не распределяются.

12.1.8. Вычислительная сложность

Слой внимания, о котором шла речь ранее, берет набор из N векторов длиной D каждый и сопоставляет их с другим набором из N векторов той же размерности. Получается, что входы и выходы имеют общую размерность $N D$. В случае использования стандартной полносвязной нейронной сети для отображения входных значений в выходные значения она имела бы $\mathcal{O}(N^2 D^2)$ независимых параметров. Точно так же вычислительные затраты на оценку одного прямого прохода через такую сеть также составили бы $\mathcal{O}(N^2 D^2)$.

В слое внимания матрицы $\mathbf{W}^{(q)}$, $\mathbf{W}^{(k)}$ и $\mathbf{W}^{(v)}$ являются общими для всех входных лексем, и поэтому количество независимых параметров составляет $\mathcal{O}(D^2)$ при условии, что $D_k \simeq D_v \simeq D$. Поскольку имеется N входных лексем, количество вычислительных шагов при оценке точечных произведений в слое самовнимания составляет $\mathcal{O}(N^2 D)$. Слой самовнимания можно представить в виде разреженной матрицы, в которой параметры распределены между ее определенными блоками (см. упражнение 12.6). Последующий слой нейронной сети, имеющий D входов и D выходов, характеризуется значением вычислительных издержек, равным $\mathcal{O}(D^2)$. Поскольку он разделяется между лексемами, его сложность линейна по отношению к N , и поэтому в целом затраты на этот слой равны $\mathcal{O}(ND^2)$. В зависимости от относительных размеров N и D в структуре вычислительных затрат может преобладать либо слой трансформера, либо слой MLP. По сравнению с полносвязной сетью слой трансформера является более эффективным с вычислительной точки зрения. В настоящее время разработано множество вариантов архитектуры трансформеров (Lin et al., 2021; Phuong and Hutter, 2022), включая модификации, направленные на повышение эффективности (Tay et al., 2020).

12.1.9. Позиционное кодирование

В архитектуре трансформера матрицы $\mathbf{W}_h^{(q)}$, $\mathbf{W}_h^{(k)}$ и $\mathbf{W}_h^{(v)}$ являются общими для всех входных лексем, как и последующая нейронная сеть. Как следствие, трансформер обладает тем свойством, что перестановка порядка входных лексем, т. е. строк \mathbf{X} (см. упражнение 12.7), приводит к такой же перестановке строк выходной матрицы $\tilde{\mathbf{X}}$. Другими словами, трансформер **эквиавариантен** относительно входных перестановок (см. раздел 10.2). Разделение параметров в архитектуре сети облегчает массовую параллельную обработку данных с помощью трансформера, а также позволяет сети обучаться зависимостям дальнего действия так же эффективно, как и зависимостям ближнего типа. Однако отсутствие зависимости от порядка следования лексем становится серьезным ограничением в случае обработки последовательных данных, таких как слова в естественном языке, поскольку представления, которым обучался трансформер, не зависят от порядка следования входных

лексем. Два предложения «*The food was bad, not good at all*» (Еда была плохая, совсем не хорошая) и «*The food was good, not bad at all*» (Еда была хорошая, совсем не плохая) содержат одни и те же лексемы, но имеют совершенно разные значения из-за разного порядка следования лексем. Совершенно очевидно, что порядок лексем имеет решающее значение для большинства задач последовательной обработки, включая обработку естественного языка, и поэтому необходимо найти способ введения в сеть информации о порядке следования лексем.

Чтобы сохранить мощные свойства тщательно сконструированных слоев внимания, необходимо кодировать порядок лексем в самих данных, а не в архитектуре сети. Поэтому для каждой входной позиции n нужно построить вектор позиционного кодирования \mathbf{r}_n , а затем объединить его с соответствующим входным вложением лексем \mathbf{x}_n . Одним из очевидных способов объединения этих векторов была бы их конкатенация, но это приведет к увеличению размерности входного пространства и, следовательно, всех последующих пространств внимания, что значительно увеличит вычислительные затраты. Вместо этого можно просто добавить векторы позиций к векторам маркеров, чтобы получить

$$\hat{\mathbf{x}}_n = \mathbf{x}_n + \mathbf{r}_n. \quad (12.24)$$

Для этого необходимо, чтобы векторы позиционного кодирования имели ту же размерность, что и векторы вложения лексем.

Сначала может показаться, что добавление информации о положении к вектору маркера приведет к искажению входных векторов и значительно усложнит задачу сети. Однако понимание причин, по которым это может дать хорошие результаты, приходит из того, что два случайно выбранных некоррелированных вектора имеют тенденцию быть почти ортогональными в пространствах высокой размерности (см. упражнение 12.8). Это свидетельствует о том, что сеть способна обрабатывать информацию об идентичности маркера и информацию о положении практически по отдельности. Отметим также, что вследствие наличия остаточных связей между всеми слоями информация о положении не теряется при переходе от одного слоя трансформера к другому. Более того, ввиду линейности слоев обработки в трансформере конкатенированное представление имеет те же свойства, что и аддитивное (см. упражнение 12.9).

Следующая задача сводится к построению векторов встраивания $\{\mathbf{r}_n\}$. Простой подход заключается в том, чтобы связать с каждой позицией целое число 1, 2, 3, Однако в этом случае проблема заключается в том, что величина значения неограниченно растет, и поэтому вектор встраивания может оказаться заметно искаженным. Кроме того, этот метод может оказаться непригодным для новых входных последовательностей, которые длиннее использовавшихся при обучении, поскольку в них значения кодирования будут лежать вне диапазона использовавшихся при обучении. В качестве альтернативы можно присвоить число в диапазоне (0, 1) каждой лексеме в последовательности, что сохранит представление ограниченным. Однако

такое представление не является уникальным для данной позиции, поскольку оно зависит от общей длины последовательности.

Идеальное позиционное кодирование должно обеспечивать уникальное представление для каждой позиции, быть ограниченным, обобщаться на более длинные последовательности и иметь последовательный способ выражения количества шагов между любыми двумя входными векторами независимо от их абсолютного положения, поскольку относительное положение лексем часто важнее абсолютного.

Существует множество методов позиционного кодирования (Dufter, Schmitt and Schütze, 2021). Здесь представлена техника, основанная на синусоидальных функциях, представленная в работе (Vaswani et al., 2017). Для определенной позиции i соответствующий вектор позиционного кодирования имеет компоненты r_{ni} , заданные как

$$r_{ni} = \begin{cases} \sin\left(\frac{n}{L^{i/D}}\right), & \text{если } i \text{ четное,} \\ \cos\left(\frac{n}{L^{(i-1)/D}}\right), & \text{если } i \text{ нечетное.} \end{cases} \quad (12.25)$$

Как видно, элементы вектора вложения r_n задаются серией синусоидальных и косинусоидальных функций с постоянно увеличивающейся длиной волны, как показано на рис. 12.10а. ((а) График, на котором горизонтальная ось показывает различные компоненты вектора встраивания r , а вертикаль-

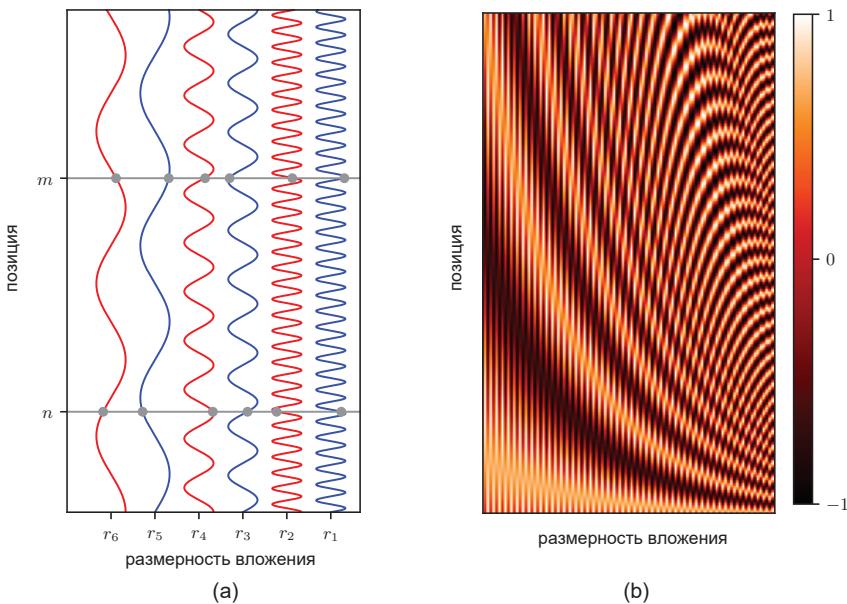


РИС. 12.10 Иллюстрации функций, определенных в (12.25) и используемых для построения векторов позиционного кодирования

ная ось – это положение в последовательности. Значения элементов вектора для двух позиций n и m показаны пересечениями кривых синуса и косинуса с горизонтальными серыми линиями. (b) Термовая карта векторов позиционного кодирования, определенных в (12.25) для размерности $D = 100$ с $L = 30$ для первых $N = 200$ позиций.)

Это кодирование обладает тем свойством, что все элементы вектора \mathbf{r}_n лежат в диапазоне $(-1, 1)$. Это напоминает способ представления двоичных чисел, при котором бит самого низкого порядка чередуется с высокой частотой, а последующие биты чередуются с неуклонно уменьшающейся частотой:

1:	0	0	0	1
2:	0	0	1	0
3:	0	0	1	1
4:	0	1	0	0
5:	0	1	0	1
6:	0	1	1	0
7:	0	1	1	1
8:	1	0	0	0
9:	1	0	0	1

Однако для кодирования, заданного в (12.25), элементы вектора имеют вид непрерывных переменных, а не двоичных. График векторов позиционного кодирования показан на рис. 12.10b.

Одно из полезных свойств синусоидального представления в (12.25) заключается в том, что для любого фиксированного смещения k кодирование в положении $n + k$ может быть представлено как линейная комбинация кодирования в положении n (см. упражнение 12.10), в которой коэффициенты не зависят от абсолютного положения, а только от значения k . Поэтому необходимо, чтобы сеть могла научиться воспринимать относительные положения. Обратите внимание, что это свойство предполагает, что для кодирования используются как синусоидальные, так и косинусоидальные функции.

Другой популярный подход к позиционному представлению предполагает использование обучаемого позиционного кодирования. Для этого используется вектор весов для каждой позиции лексемы, и этот вектор может быть обучен совместно с остальными параметрами модели в процессе обучения, что позволяет избежать ручной работы при создании представлений. Поскольку в позициях лексем параметры не разделяются, лексемы больше не являются инвариантными при перестановке, что и является основной целью позиционного кодирования. Однако этот подход не отвечает критериям, о которых говорилось ранее по поводу обобщения на более длинные входные последовательности, поскольку кодирование будет необучаемым для позиционных кодировок, не встречающихся во время обучения. Поэтому такой подход, как правило, оптимальен в тех случаях, когда длина входных данных относительно постоянна как во время обучения, так и во время вывода.

12.2. Естественный язык

Теперь после изучения архитектуры трансформера можно приступить к изучению способов его использования для обработки языковых данных, состоящих из слов, предложений и абзацев. Хотя трансформеры изначально разрабатывались именно для работы с этими формами, они оказались чрезвычайно универсальным классом моделей и уже стали самыми популярными для большинства типов входных данных. Позже в этой главе будет рассмотрена тема их использования в других областях (см. раздел 12.4).

Многие языки, включая английский, состоят из ряда слов, разделенных пробелами, а также знаков препинания и поэтому представляют собой пример последовательных данных (см. раздел 11.3). Здесь основное внимание будет уделено словам, а к пунктуации вернемся позже (см. раздел 12.2.2).

Первая задача заключается в преобразовании слов в числовое представление, пригодное для использования в качестве входных данных для глубокой нейронной сети. Один из простых подходов состоит в составлении фиксированного словаря слов с последующим введением векторов длины, равной размеру словаря, а также «горячего» представления для каждого слова, при котором k -е слово в словаре кодируется вектором, имеющим 1 в позиции k и 0 во всех остальных позициях. Например, если «aardwolf» (земляной волк) находится на третьей позиции нашего словаря, то его векторное представление будет иметь вид $(0, 0, 1, 0, \dots, 0)$.

Очевидная проблема с одноточечным представлением заключается в том, что реальный словарь может содержать несколько сотен тысяч записей, что приводит к появлению векторов очень высокой размерности. Кроме того, такое представление не отражает сходства или взаимосвязей, которые могут существовать между словами. Обе проблемы можно решить путем отображения слов в более низкоразмерное пространство с помощью процесса, называемого *векторным представлением слов* (*word embedding*), при котором каждое слово представляется в виде плотного вектора в пространстве, имеющем, как правило, несколько сотен измерений.

12.2.1. Векторное представление слов

Векторное представление слов может быть определено матрицей \mathbf{E} размера $D \times K$, где D – это размерность пространства встраивания, а K – это размерность словаря. Для каждого одноточечного закодированного входного вектора \mathbf{x}_n можно вычислить соответствующее векторное представление с использованием

$$\mathbf{v}_n = \mathbf{E}\mathbf{x}_n. \quad (12.26)$$

Поскольку \mathbf{x}_n имеет одноточечное кодирование, вектор \mathbf{v}_n просто задается соответствующим столбцом матрицы \mathbf{E} .

Обучение матрицы \mathbf{E} можно провести с использованием корпуса (corpus, большого массива данных) текстов, и для этого существует множество раз-

личных подходов. Здесь рассмотрим популярную методику под названием *word2vec* (Mikolov et al., 2013), которая может быть представлена в виде простой двухслойной нейронной сети. Для обучения составляется набор, в котором каждая выборка получается путем просмотра «окна» из M соседних слов текста, где типичным значением может быть $M = 5$. Выборки считаются независимыми, а функция ошибки определяется как сумма функций ошибки для каждой выборки. Существует два варианта этого подхода. В *непрерывном мульти множестве слов (continuous bag of words)* целевой переменной для обучения сети является среднее слово, а остальные контекстные слова служат входом, так что сеть обучается «заполнению пустого места». Похожий подход, называемый *skip-grams* (Буквально «пропуск последовательности». – Прим. перев.), заключается в том, что входы и выходы меняются местами, так что центральное слово представляется в качестве входа, а целевыми значениями являются контекстные слова. Эти модели показаны на рис. 12.11.

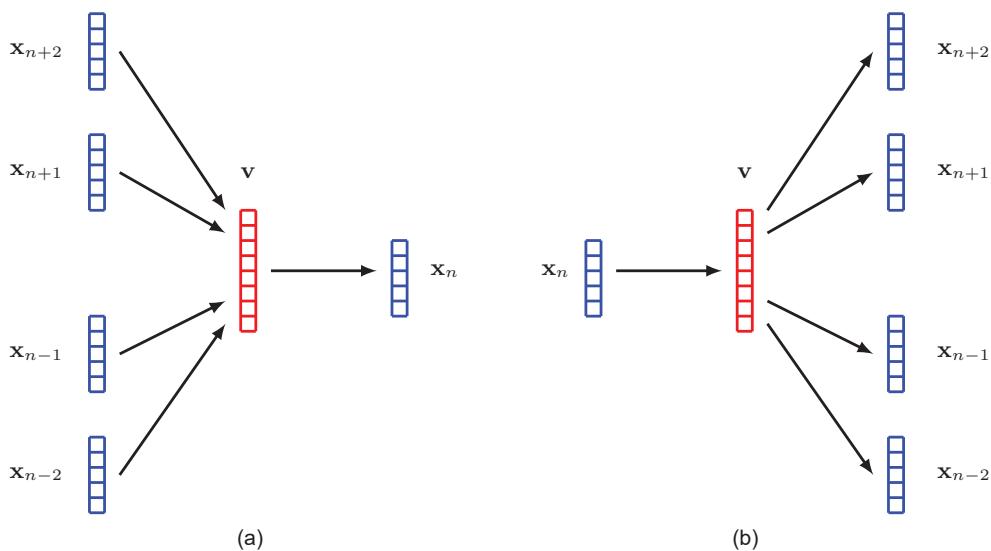


РИС. 12.11 Двухслойные нейронные сети для обучения векторному представлению слов, где (a) подход непрерывного мульти множества слов, (b) подход skip-grams

Такую процедуру обучения можно рассматривать как разновидность самоподдерживающегося обучения, поскольку данные состоят всего лишь из большого массива немаркированного текста, из которого случайным образом выбирается множество небольших окон последовательностей слов. Метки получаются из самого текста путем «маскировки» тех слов, значения которых сеть пытается предсказать.

После обучения модели матрица встраивания E задается транспонированием матрицы весов второго слоя для метода непрерывного мульти множества слов и матрицы весов первого слоя для метода skip-grams. Семантически связанные слова отображаются на соседние позиции в пространстве

встраивания. Это вполне ожидаемо, поскольку связанные слова с большей вероятностью будут встречаться с аналогичными контекстными словами по сравнению с несвязанными словами. Например, слова «город» и «столица» могут встречаться с большей частотой в качестве контекста для таких целевых слов, как «Париж» или «Лондон», и с меньшей частотой в качестве контекста для слов «апельсин» или «полином». Сеть может легче предсказать вероятность появления пропущенных слов, если «Париж» и «Лондон» будут отображены на соседние векторы встраивания.

Как выяснилось, обученное пространство встраивания зачастую имеет даже более богатую семантическую структуру, чем простая близость родственных слов, и это позволяет использовать простую векторную арифметику. Например, концепция «Париж для Франции – это как Рим для Италии» может быть выражена с помощью операций над векторами встраивания. Если обозначить вектор встраивания для слова «word» (слово) с помощью $v(\text{word})$, то получится, что

$$v(\text{Paris}) - v(\text{France}) + v(\text{Italy}) \simeq v(\text{Rome}). \quad (12.27)$$

Изначально встраивание слов разрабатывалось как самостоятельный инструмент для обработки естественного языка. Сегодня они чаще всего используются в качестве предварительной обработки для глубоких нейронных сетей. В этом отношении их можно рассматривать в качестве первого слоя глубокой нейронной сети. Они могут быть фиксированными, используя некоторую стандартную предварительно обученную матрицу встраивания, или же они могут рассматриваться как адаптивный слой, который обучается как часть общей системы сквозного обучения. В последнем случае слой встраивания может быть инициализирован либо с помощью случайных значений весов, либо с помощью стандартной матрицы встраивания.

12.2.2. Лексическая обработка

Одна из проблем использования фиксированного словаря слов заключается в том, что он не подходит для слов, которых в нем нет или которые написаны неправильно. Он также не учитывает знаки препинания или другие последовательности символов, например компьютерный код. Альтернативным подходом, решающим эти проблемы, является принцип работы на уровне символов, а не слов, так что в словарь входят прописные и строчные буквы, цифры, знаки препинания и символы пробелов, такие как пробелы и табуляции. Однако недостатком такого подхода является отказ от семантически важной структуры слов языка, и в дальнейшем нейронной сети придется учиться собирать слова из элементарных символов. Кроме того, это потребует гораздо большего количества последовательных шагов для конкретного текста, что увеличит вычислительные затраты на его обработку.

Преимущества представлений на уровне символов и слов можно объединить посредством этапа предварительной обработки, который преобразует строку слов и знаков препинания в строку лексем (*tokens*), обычно представ-

ляющих собой небольшие группы символов и включающих обычные слова полностью, а также фрагменты длинных слов и отдельные символы, которые могут быть собраны в менее распространенные слова (Schuster and Nakajima, 2012). Подобная лексическая обработка (*tokenization*) также позволяет системе обрабатывать другие виды последовательностей, такие как компьютерный код и даже такие формы, как изображения (см. раздел 12.4.1). Это также означает, что разные варианты одного и того же слова будут иметь взаимосвязанные представления. Например, слова «cook», «cooks», «cooked», «cooking» и «cooker» взаимосвязаны и имеют общий элемент «cook», который сам по себе может быть представлен как одна из лексем.

Существует множество различных методов лексической обработки. Например, техника, называемая *кодировкой пар байтов* (*byte pair encoding*) и используемая для сжатия данных, может быть адаптирована к лексической обработке текста путем слияния символов вместо байтов (Sennrich, Haddow and Birch, 2015). Процесс начинается с отдельных символов и итерационно объединяет их в более длинные строки. Список лексем сначала инициализируется списком отдельных символов. Затем в тексте отыскиваются наиболее часто встречающиеся смежные пары лексем, которые заменяются новой лексемой. Чтобы не допустить слияния слов, новая лексема не формируется из двух лексем, если вторая начинается с пробела. Процесс повторяется итерационно, как показано на рис. 12.12. В этом примере наиболее часто встречающейся парой символов является «ре», которая встречается четыре раза, поэтому они образуют новую лексему, которая заменяет все вхождения «ре». Обратите внимание, что «Ре» не включается в эту пару, поскольку прописная «Р» и строчная «р» – это разные символы. Далее добавляется пара «ск», поскольку она встречается три раза. Затем следуют такие лексемы, как «рі», «ед» и «пер», которые встречаются дважды, и т. д.

Peter Piper picked a peck of pickled peppers
Peter Piper picked a peck of pickled peppers

РИС. 12.12 Иллюстрация процесса токенизации естественного языка по аналогии с кодированием пар байтов

Изначально количество лексем равно количеству символов, что относительно немного. По мере формирования лексем их общее количество увеличивается, и если это продолжается достаточно долго, то в конечном итоге

лексемы будут соответствовать набору слов в тексте. Общее количество лексем обычно устанавливается предварительно и представляет собой компромисс между представлениями на уровне символов и на уровне слов. При достижении этого количества лексем алгоритм останавливается.

В практических приложениях глубокого обучения естественного языка входной текст обычно сначала преобразуется в представление на уровне лексем. Однако в оставшейся части этой главы будут использоваться представления на уровне слов, поскольку так легче проиллюстрировать и объяснить ключевые понятия.

12.2.3. Мультимножество слов

Теперь перейдем к задаче моделирования совместного распределения $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ упорядоченной последовательности векторов, таких как слова (или лексемы) в естественном языке. Самый простой подход предполагает, что слова берутся независимо из одного и того же распределения, и, следовательно, совместное распределение полностью факторизуется в виде

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{n=1}^N p(\mathbf{x}_n). \quad (12.28)$$

Это можно выразить в виде вероятностной модели графа, в которой узлы изолированы и не имеют соединительных связей (рис. 11.28).

Распределение $p(\mathbf{x})$ является общим для всех переменных и может быть представлено без потери обобщения в виде простой таблицы с перечислением вероятностей каждого из возможных состояний \mathbf{x} (соответствующих словарю слов или лексем). Решение с максимальным правдоподобием для этой модели можно получить простой настройкой каждой из этих вероятностей (см. упражнение 12.11) на количество повторений этого слова в обучающем наборе. Такая форма известна как *модель мультимножества слов*, или просто «*модель мешка слов*» (*bag-of-words model*), поскольку в ней полностью игнорируется порядок следования слов.

На основе этой модели можно построить простой текстовый классификатор. Он может использоваться, например, для анализа настроения, когда отрывок текста, представляющий собой отзыв о ресторане, необходимо классифицировать как положительный или отрицательный. В *классификаторе наивного Байеса* (*naive Bayes classifier*) подразумевается, что слова являются независимыми в пределах каждого класса \mathcal{C}_k , но с различным распределением для каждого класса, так что

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathcal{C}_k) = \prod_{n=1}^N p(\mathbf{x}_n | \mathcal{C}_k). \quad (12.29)$$

С учетом априорных вероятностей классов $p(\mathcal{C}_k)$ апостериорные вероятности классов для новой последовательности определяются следующим образом:

$$p(\mathcal{C}_k | \mathbf{x}_1, \dots, \mathbf{x}_N) \propto p(\mathcal{C}_k) \prod_{n=1}^N p(\mathbf{x}_n | \mathcal{C}_k). \quad (12.30)$$

Как условные плотности $p(\mathbf{x} | \mathcal{C}_k)$, так и априорные вероятности $p(\mathcal{C}_k)$ могут быть оценены с помощью значений частот из обучающего набора данных. Для получения новой последовательности записи таблицы перемножаются, чтобы получить желаемые апостериорные вероятности. Обратите внимание, что если в тестовом наборе встречается слово, которого не было в обучающем наборе, то соответствующая оценка вероятности будет равна нулю, поэтому эти оценки обычно «сглаживаются» после обучения путем переназначения небольшого уровня вероятности равномерно по всем записям во избежание нулевых значений.

12.2.4. Модели авторегрессии

Одно из очевидных ограничений модели «мешок слов» заключается в полном игнорировании порядка слов. Для решения этой проблемы можно применить авторегрессионный подход. Для этого без потери общности можно разложить распределение по последовательности слов на произведение условных распределений в виде

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}). \quad (12.31)$$

Это можно представить в виде вероятностной модели графа, в которой каждый узел в последовательности получает ссылку от каждого предыдущего узла (рис. 11.27). Каждый член в правой части (12.31) можно представить в виде таблицы, записи которой еще раз оцениваются с помощью простых подсчетов значений частоты из обучающего набора. Однако размер этих таблиц растет экспоненциально (см. упражнение 12.12) с увеличением длины последовательности, поэтому такой подход будет непомерно затратным.

Эту модель можно значительно упростить, если предположить, что каждое из условных распределений в правой части (12.31) не зависит от всех предыдущих наблюдений, кроме L самых последних слов. Например, если $L = 2$, то совместное распределение для последовательности из N наблюдений в рамках этой модели будет иметь вид:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2 | \mathbf{x}_1) \prod_{n=3}^N p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{x}_{n-2}). \quad (12.32)$$

В соответствующей модели графа каждый узел имеет связи с двумя предыдущими узлами (рис. 11.30). Здесь подразумевается, что условные распределения $p(\mathbf{x}_n | \mathbf{x}_{n-1})$ являются общими для всех переменных. И снова каждое из распределений в правой части (12.32) может быть представлено в виде таблиц, значения которых оцениваются по статистике тройки последовательных слов, взятых из обучающего корпуса.

Случай с $L = 1$ известен как *биграммная модель* (*bi-gram model*), поскольку она зависит от пар соседних слов. Аналогично случай при $L = 2$, где включены тройки соседних слов, называется *триграммной моделью* (*tri-gram model*), а в общем случае эти модели называются *n-граммными* (*n-gram models*).

Все рассмотренные ранее в этом разделе модели могут быть реализованы в генеративном режиме для синтеза нового текста. Например, если в последовательности слов даны первое и второе слова, то для создания третьего слова можно использовать выборку из статистики триграмм $p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{x}_{n-2})$, для четвертого – второе и третье слова и т. д. Однако полученный текст будет бессвязным, поскольку каждое слово прогнозируется только на основе двух предыдущих слов. Высококачественные текстовые модели должны учитывать дальние зависимости в языке. С другой стороны, нельзя просто взять и увеличить значение L , поскольку размер таблиц вероятностей растет экспоненциально по L , так что выход за рамки триграммных моделей оказывается непомерно затратным. Однако авторегрессионное представление будет играть центральную роль при изучении современных языковых моделей, основанных не на таблицах вероятностей, а на глубоких нейронных сетях, настроенных в качестве трансформеров.

Одним из способов обеспечить более дальние зависимости, избежав при этом экспоненциального роста числа параметров *n-граммной* модели, является использование *скрытой марковской модели* (см. раздел 11.3.1), структура графа которой показана на рис. 11.31. В этом случае число обучаемых параметров зависит от размерности скрытых переменных, а распределение по данному наблюдению \mathbf{x}_n в принципе зависит от всех предыдущих наблюдений. Однако влияние более отдаленных наблюдений все еще очень ограничено, поскольку их действие должно проходить через цепочку латентных состояний, которые сами обновляются более поздними наблюдениями.

12.2.5. Рекуррентные нейронные сети

Методы, подобные *n-граммным*, очень плохо масштабируются при увеличении длины последовательности, поскольку хранят слишком общие таблицы условных распределений. Гораздо лучшего масштабирования можно добиться при использовании параметризованных моделей на основе нейронных сетей. Например, для обработки последовательностей слов в естественном языке можно использовать стандартную нейронную сеть с прямой передачей. Возникает проблема: сеть имеет фиксированное количество входов и выходов, в то время как в обучающем и тестовом множествах необходимо иметь возможность обрабатывать последовательности разной длины. Кроме того, если слово или группа слов в определенном месте последовательности представляет собой какой-то концепт, то это же самое слово или группа слов в другом месте, скорее всего, будет представлять тот же самый концепт в этом новом месте. Это напоминает свойство эквивариантности, о котором шла речь при обработке данных изображений (см. главу 10). Если построить сетевую архитектуру, способную разделять параметры по всей последова-

тельности, то можно не только отразить это свойство эквивариантности, но и значительно сократить количество свободных параметров в модели, а также работать с последовательностями разной длины.

Для решения этой проблемы можно взять за основу скрытую модель Маркова и ввести в явном виде скрытую переменную z_n , связанную с каждым шагом n в последовательности. Нейронная сеть принимает на вход актуальное слово x_n и актуальное скрытое состояние z_{n-1} , а на выходе выдает слово y_n , а также следующее состояние z_n скрытой переменной. Затем можно объединить копии этой сети в цепочку, при этом значения весов будут общими для всех копий. Полученная архитектура называется *рекуррентной нейронной сетью* (*recurrent neural network, RNN*), а ее пример показан на рис. 12.13. Здесь начальное значение скрытого состояния может быть инициализировано, например, некоторым значением по умолчанию, таким как $z_0 = (0, 0, \dots, 0)^T$.

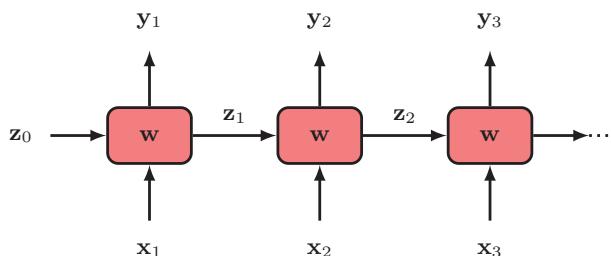


РИС. 12.13 Стандартная RNN с параметрами w . Она принимает на вход последовательность x_1, \dots, x_N в качестве входных данных и генерирует последовательность y_1, \dots, y_N в качестве выхода. Каждая из ячеек соответствует многослойной сети с нелинейными скрытыми блоками

В качестве примера практического применения RNN рассмотрим конкретную задачу перевода предложений с английского языка на голландский. Предложения могут иметь разную длину, и каждое выходное предложение может отличаться от соответствующего входного предложения. Кроме того, сеть может потребоваться увидеть все входное предложение, прежде чем она сможет начать генерировать выходное предложение. Эту проблему можно решить с помощью RNN, подавая на вход полное английское предложение, а затем специальную входную лексему, которую обозначим $\langle \text{start} \rangle$, для инициализации начала перевода. В процессе обучения сеть обучается ассоциировать $\langle \text{start} \rangle$ с началом выходного предложения. Кроме того, каждое последовательно генерированное слово подается на вход на следующем временном этапе, как показано на рис. 12.14. Сеть можно обучить формированию определенной лексемы $\langle \text{stop} \rangle$ для обозначения завершения перевода. Первые несколько стадий сети используются для усвоения входной последовательности, при этом связанные выходные векторы просто игнорируются. Эту часть сети можно рассматривать в качестве «кодировщика», где все входное предложение было сжато до состояния z^* скрытой переменной. На

остальных стадиях сети работает «декодер», который генерирует переведенное предложение на выходе по одному слову за раз. Обратите внимание, что каждое выходное слово подается на вход следующей стадии сети, поэтому такой подход имеет авторегрессионную структуру, аналогичную (12.31).

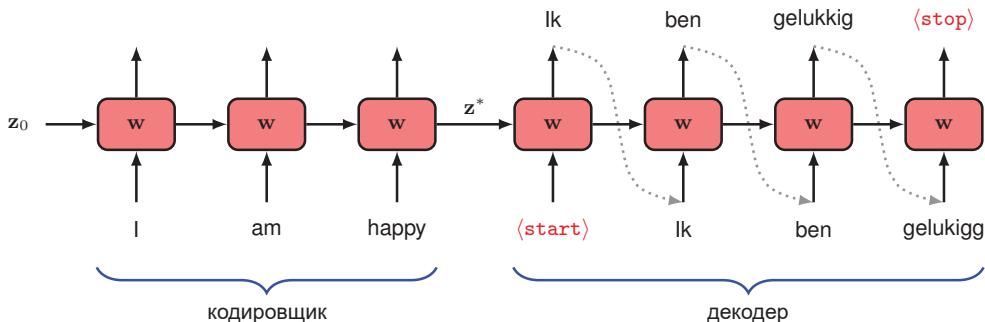


РИС. 12.14 Пример рекуррентной нейронной сети, используемой для перевода языка

12.2.6. Обратное распространение во времени

Обучение RNN может осуществляться методом стохастического градиентного спуска с использованием градиентов, вычисляемых методом обратного распространения и оцениваемых с помощью автоматического дифференцирования, как и в случае обычных нейронных сетей. Функция ошибки складывается из суммы по всем выходным блокам ошибки для каждого блока, в котором каждый выходной блок имеет функцию активации softmax (см. раздел 5.4.4) вместе с соответствующей функцией ошибки перекрестной энтропии. При прямом распространении значения активации распространяются от первого входа к выходным узлам в последовательности, а сигналы ошибки затем распространяются обратно по тем же путям. Этот процесс называется *обратным распространением во времени* (*backpropagation through time*) и принципиально очень прост. Однако на практике для очень длинных последовательностей обучение может быть затруднено из-за проблемы исчезающих или взрывающихся градиентов (см. раздел 7.4.2), встречающихся в архитектурах очень глубоких сетей.

Еще одна проблема стандартных RNN заключается в недостаточном взаимодействии с дальними зависимостями. Это особенно проблематично для естественного языка, где подобные зависимости встречаются повсеместно. В длинном отрывке текста может появиться концепт, который играет важную роль в прогнозировании слов, встречающихся в тексте намного позже. В архитектуре, показанной на рис. 12.14, вся концепция предложения на английском языке должна быть отражена в одном скрытом векторе z^* фиксированной длины, и это становится все более проблематичным при увеличении длины последовательности. Это явление известно в качестве *проблемы узкого места* (*bottleneck problem*), поскольку последовательность произвольной дли-

ны должна быть обобщена в одном скрытом векторе активаций, и сеть сможет приступить к генерации выходного перевода только после завершения полной обработки входной последовательности.

Один из способов решения проблем исчезающих и взрывающихся градиентов, а также ограниченности дальнодействующих зависимостей заключается в изменении архитектуры нейронной сети для обеспечения дополнительных путей передачи сигнала, которые обходят многие шаги обработки на каждой стадии сети и, следовательно, позволяют запоминать информацию за большее количество временных шагов. Среди наиболее известных примеров – модели с *долговременной и кратковременной памятью* (*long short-term memory*, LSTM) (Hochreiter and Schmidhuber, 1997) и модели с *управляемым рекуррентным элементом* (*gated recurrent unit*, GRU) (Cho et al., 2014). Несмотря на то что они улучшают производительность по сравнению со стандартными RNN, их возможности по моделированию дальних зависимостей по-прежнему ограничены. Кроме того, дополнительная сложность каждой ячейки означает, что LSTM обучаются еще медленнее, чем стандартные RNN. К тому же все рекуррентные сети имеют сигнальные пути, которые линейно растут с количеством шагов в последовательности. При этом они не поддерживают параллельные вычисления в рамках одного обучающего примера из-за последовательного характера обработки. В частности, это означает, что RNN не могут эффективно использовать современное аппаратное обеспечение с высокой степенью параллелизма на базе графических процессоров. Эти проблемы решаются путем замены RNN на трансформеры.

12.3. Языковые модели трансформеров

Слой обработки трансформера является чрезвычайно гибким модулем для построения мощных нейросетевых моделей с обширными возможностями применения. В этом разделе представлены возможности применения трансформеров для обработки естественного языка. Благодаря этим технологиям были разработаны большие нейронные сети, известные как *большие языковые модели* (*large language models*, LLM), которые уже доказали свои исключительные возможности (Zhao et al., 2023).

Трансформеры можно использовать для решения самых разнообразных задач обработки языка, и в зависимости от формы входных и выходных данных их можно разделить на три категории. В такой задаче, как анализ настроения, на вход подается последовательность слов, а на выходе получается одна переменная, отражающая настроение текста, например радостное или грустное. Здесь трансформер выступает в роли кодировщика последовательности. В других задачах на вход может подаваться один вектор, а на выходе генерироваться последовательность слов, например когда нужно получить текстовую надпись к входному изображению. В таких

случаях трансформер работает в качестве декодера, генерируя на выходе последовательность. Наконец, в задачах обработки последовательности слов и вход, и выход представляют собой последовательность слов, например при переводе с одного языка на другой. В этом случае трансформеры используются как в роли кодировщиков, так и в роли декодеров. Далее каждый из этих классов языковых моделей будет рассмотрен на наглядных примерах модельных архитектур.

12.3.1. Декодирующие трансформеры

Для начала рассмотрим модели трансформеров только для декодирования. Они могут быть использованы в качестве *генеративных моделей*, создающих выходные последовательности лексем. В качестве наглядного примера остановимся на классе моделей под названием *GPT*, что расшифровывается как *предварительно обученный генеративный трансформер (generative pretrained transformer)* (Radford et al., 2019; Brown et al., 2020; OpenAI, 2023). Идея заключается в использовании архитектуры трансформера для построения авторегрессионной модели в форме, определенной выражением (12.31), в которой условные распределения $p(x_n | x_1, \dots, x_{n-1})$ определяются с помощью нейронной сети трансформера, обученной на данных.

Модель принимает на вход последовательность из первых $n - 1$ лексем, а ее соответствующий выход представляет условное распределение для лексемы n . Если взять выборку из этого распределения, то последовательность будет расширена до n лексем, и эта новая последовательность может быть вновь обработана моделью с целью получения распределения для лексемы $n + 1$ и т. д. Этот процесс можно повторять для генерации последовательностей до максимальной длины, определяемой количеством входов трансформера. В разделе 12.3.2 будут кратко рассмотрены стратегии выборки из условных распределений, а сейчас основное внимание будет уделено созданию и обучению сети.

Архитектура модели GPT состоит из стека слоев трансформера, которые принимают на вход последовательность x_1, \dots, x_N лексем, каждая из которых имеет размерность D , в качестве входных данных, и производят последовательность $\tilde{x}_1, \dots, \tilde{x}_N$ лексем, опять же размерности D , в качестве выхода. Каждый выход должен представлять собой распределение вероятностей по словарю лексем на данном временном шаге, при этом словарь имеет размерность K , тогда как лексемы имеют размерность D . Поэтому для каждой выходной лексемы выполняется линейное преобразование с помощью матрицы $W^{(p)}$ размерности $D \times K$, за которой следует функция активации Softmax в виде

$$\mathbf{Y} = \text{Softmax}(\tilde{\mathbf{X}}\mathbf{W}^{(p)}), \quad (12.33)$$

где \mathbf{Y} – это матрица, n -я строка которой y_n^T , а $\tilde{\mathbf{X}}$ – это матрица, n -я строка которой \tilde{x}_n^T (см. раздел 5.4.4). Каждый выходной элемент softmax имеет соответствующую функцию ошибки перекрестной энтропии. Архитектура модели показана на рис. 12.15.

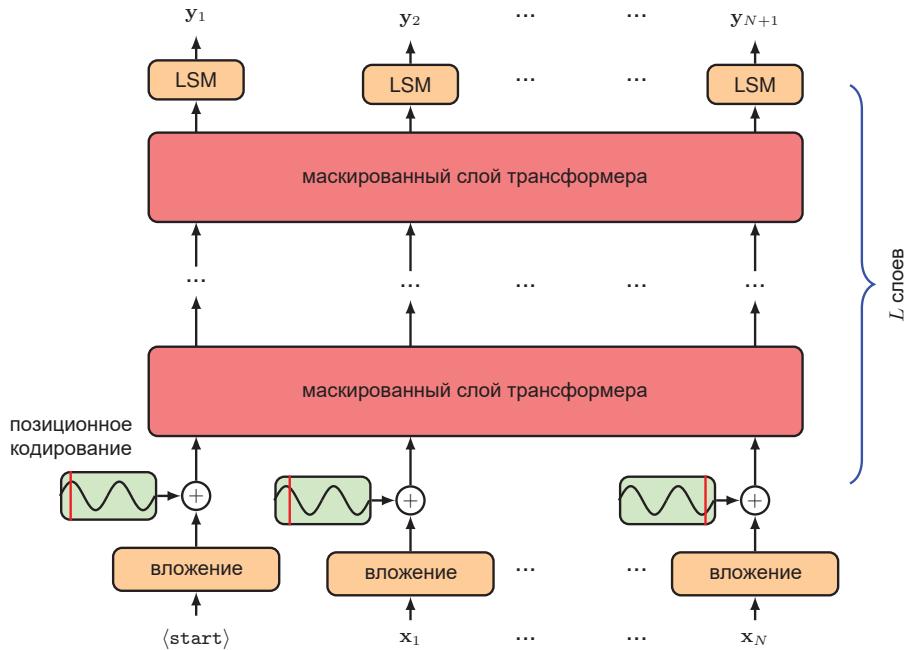


РИС. 12.15 Архитектура сети трансформеров декодера GPT. Здесь «LSM» означает *linear-softmax* (линейное преобразование, обучаемые параметры которого распределены по позициям лексем, с последующей функцией активации softmax)

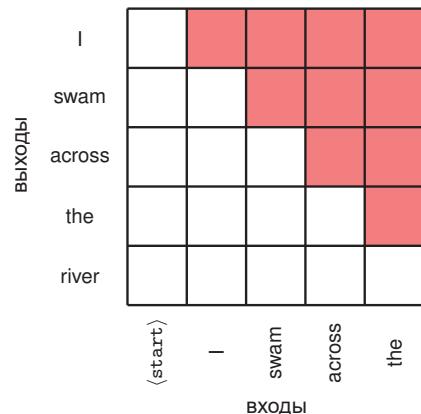
Модель можно обучить на большом корпусе немаркированного естественного языка с использованием метода самоконтроля. Каждая обучающая выборка состоит из последовательности лексем x_1, \dots, x_n , которые образуют вход сети, а также связанного с ними целевого значения x_{n+1} , состоящего из следующей лексемы в последовательности. Последовательности считаются независимыми и одинаково распределенными, поэтому функция ошибки, используемая для обучения, представляет собой сумму значений ошибки перекрестной энтропии, суммированных по обучающему множеству, которое сгруппировано в соответствующие мини-батчи. На первый взгляд, каждую такую обучающую выборку можно было бы обрабатывать отдельно методом прямого прохода через модель. Однако можно добиться гораздо большей эффективности за счет одновременной обработки всей последовательности, так что каждая лексема выступает одновременно и в качестве целевого значения для последовательности предыдущих лексем, и в качестве входного значения для последующих лексем. Для примера рассмотрим последовательность слов:

I swam across the river to get to the other bank (Я переплыл реку, чтобы перебраться на другой берег).

Можно использовать «I swam across» в качестве входной последовательности с ассоциированной целью «the» или использовать «I swam across the» в качестве входной последовательности с ассоциированной целью «river» и т. д.

Однако для параллельной обработки этих данных необходимо гарантировать, что сеть не сможет «обмануть», заглянув вперед в эту последовательность, иначе она просто научится копировать следующий входной сигнал непосредственно на выход. Если это произойдет, то сеть не сможет генерировать новые последовательности, поскольку последующий маркер по определению недоступен в момент тестирования. Чтобы решить эту проблему, необходимо сделать два действия. Во-первых, переместим входную последовательность вправо на один шаг, чтобы входу x_n соответствовал выход y_{n+1} с целью x_{n+1} , а в первую позицию входной последовательности вставим дополнительную специальную лексему, обозначаемую $\langle \text{start} \rangle$. Во-вторых, обратите внимание, что маркеры в трансформере обрабатываются независимо друг от друга, за исключением случаев, когда они используются для вычисления весов внимания, и тогда они попарно взаимодействуют через точечное произведение. Поэтому в каждый из слоев внимания необходимо ввести *маскированное внимание* (*masked attention*), иногда называемое *причинно-следственным вниманием* (*causal attention*), при котором все веса внимания, соответствующие лексеме, относящейся к любой последующей лексеме в последовательности, устанавливаются на ноль. Для этого нужно просто установить в ноль все соответствующие элементы матрицы внимания $\text{Attention}(Q, K, V)$, определяемой в (12.14), а затем нормализовать оставшиеся элементы так, чтобы каждая строка снова равнялась единице. На практике это достигается установкой соответствующих значений предварительной активации в $-\infty$, так что softmax оценивает ноль для соответствующих выходов и также обеспечивает нормализацию для не-нулевых выходов. Структура матрицы маскированного внимания показана на рис. 12.16. Здесь веса внимания, соответствующие красным элементам, установлены в ноль. Таким образом, при прогнозировании лексемы «across» выход может зависеть только от входных лексем $\langle \text{start} \rangle$, I и swam .

РИС. 12.16 Матрица масок для маскированного самовнимания



На практике важно максимально эффективно использовать массовый параллелизм графических процессоров, поэтому несколько последовательностей могут быть сложены во входной тензор для параллельной обработки в одном

пакете. Однако для этого необходимо, чтобы последовательности были одинаковой длины, в то время как текстовые последовательности, как правило, имеют разную длину. Эту проблему можно решить путем введения специального маркера, который обозначается $\langle \text{pad} \rangle$ и используется для заполнения неиспользуемых позиций для приведения всех последовательностей к одинаковой длине, чтобы их можно было объединить в один тензор. Затем в весах внимания используется дополнительная маска, которая гарантирует, что выходные векторы не будут учитывать входы, занятые маркером $\langle \text{pad} \rangle$. Обратите внимание, что форма этой маски зависит от конкретной входной последовательности.

Выход обученной модели – это вероятностное распределение в пространстве лексем, которое задается функцией активации softmax на выходе и представляет собой вероятность появления следующей лексемы с учетом существующей последовательности лексем. Как только следующее слово выбрано, последовательность лексем с включенной в нее новой лексемой может быть снова пропущена через модель для генерации следующей лексемы в последовательности, и этот процесс может повторяться бесконечно или до тех пор, пока не будет сгенерирована лексема конца последовательности. Это может показаться довольно неэффективным, поскольку для каждого нового генерированного маркера данные должны проходить через всю модель. Однако обратите внимание, что благодаря маскировке внимания встраивание, обученное для конкретной лексемы, зависит только от самой лексемы и от предыдущих лексем и, следовательно, не меняется при генерации новой, более поздней лексемы. Следовательно, при обработке новой лексемы большая часть вычислений может быть использована повторно.

12.3.2. Стратегии выборки

Ранее уже было показано, что на выходе декодирующего трансформера получается вероятностное распределение значений для следующей лексемы в последовательности, из которого необходимо выбрать конкретное значение для этой лексемы для продолжения последовательности. Существует несколько вариантов выбора значения лексемы на основе вычисленных вероятностей (Holtzman et al., 2019). Один из наиболее очевидных подходов, называемый *жадным поиском* (*greedy search*), сводится к простому выбору лексемы с наибольшей вероятностью. Это приводит к тому, что модель становится детерминированной, т. е. заданная входная последовательность всегда генерирует одну и ту же выходную последовательность. Обратите внимание, что просто выбрать лексему с наибольшей вероятностью на каждом этапе – это не то же самое, что выбрать последовательность лексем с наибольшей вероятностью (см. упражнение 12.15). Для поиска наиболее вероятной последовательности необходимо максимизировать совместное распределение по всем лексемам, которое задается как

$$p(\mathbf{y}_1, \dots, \mathbf{y}_N) = \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{y}_1, \dots, \mathbf{y}_{n-1}). \quad (12.34)$$

Если в последовательности N шагов, а количество значений лексем в словаре равно K , то общее количество последовательностей равно $\mathcal{O}(K^N)$, и оно растет экспоненциально с увеличением длины последовательности, следовательно, поиск единственной последовательности с наибольшей вероятностью оказывается невыполнимым. Для сравнения: жадный поиск предполагает затраты $\mathcal{O}(KN)$, которые линейно зависят от длины последовательности.

Одна из методик, позволяющая генерировать последовательности с более высокой вероятностью, чем при жадном поиске, называется *лучевым поиском* (*beam search*). Вместо выбора одного наиболее вероятного значения лексемы на каждом этапе необходимо хранить набор из B гипотез, где B называется *шириной луча* (*beam width*), каждая из которых состоит из последовательности значений лексем до шага n . Затем все эти последовательности пропускаются через сеть, и для каждой последовательности определяются B наиболее вероятных значений лексем, в результате чего создается B^2 возможных гипотез для расширенной последовательности. Затем этот список обрезается путем выбора наиболее вероятных B гипотез в соответствии с общей вероятностью расширенной последовательности. Таким образом, алгоритм лучевого поиска сохраняет B альтернативных последовательностей и отслеживает их вероятности, выбирая в итоге наиболее вероятную последовательность из всех рассмотренных. Поскольку вероятность последовательности получается путем перемножения вероятностей на каждом этапе последовательности, и поскольку эти вероятности всегда меньше или равны единице, длинная последовательность, как правило, имеет меньшую вероятность, чем короткая, что смещает результаты в сторону коротких последовательностей. По этой причине вероятности последовательностей перед сравнением обычно нормализуются по соответствующим длинам последовательностей. Лучевой поиск требует затрат $\mathcal{O}(BKN)$, которые также линейны относительно длины последовательности. Однако стоимость генерации последовательности увеличивается в B раз, и поэтому для очень больших языковых моделей, где затраты на вывод могут оказаться значительными, это обстоятельство значительно снижает привлекательность лучевого поиска.

Одна из проблем использования таких подходов, как жадный поиск и лучевой поиск, заключается в ограниченном разнообразии потенциальных результатов, что может даже привести к зацикливанию процесса генерации, когда одна и та же последовательность слов повторяется снова и снова. Как видно на рис. 12.17, текст, созданный человеком, может иметь меньшую вероятность и, следовательно, быть более уникальным по отношению к созданной модели, чем текст, созданный автоматически.

Вместо попыток найти последовательность с наибольшей вероятностью можно просто формировать последовательные лексемы, делая выборку из распределения softmax на каждом этапе. Однако это может привести к созданию бессмысленных последовательностей. Как правило, причиной этого является очень большой размер словаря лексем, в котором существует длинный хвост из множества состояний лексем, каждое из которых имеет очень малую вероятность, но в совокупности составляет значительную долю от общей суммы вероятностей. Это приводит к возникновению проблемы

значительной вероятности того, что система сделает неправильный выбор следующей лексемы.

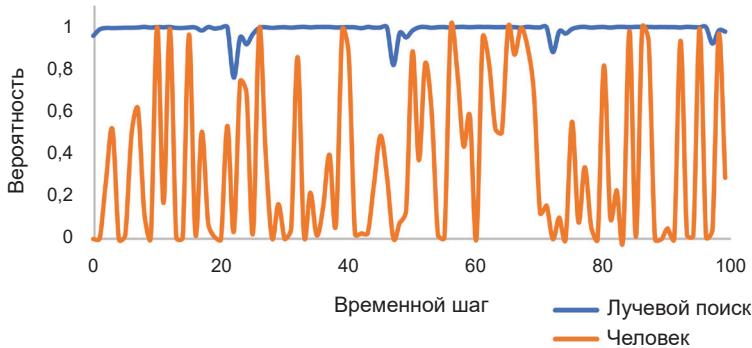


РИС. 12.17 Сравнение вероятностей лексем из лучевого поиска и человеческого текста для заданной обученной языковой модели трансформера и заданной начальной входной последовательности: последовательность, созданная человеком, имеет гораздо более низкие вероятности лексем. [Из (Holtzman et al., 2019) с разрешения авторов]

В качестве баланса между этими крайностями можно учитывать только состояния с наибольшими K вероятностями при определенном выборе K , а затем делать выборку из них в соответствии с их перенормированными вероятностями. Вариант этого подхода, называемый *выборкой верхнего p* (*top- p sampling*) или *ядерной выборкой* (*nucleus sampling*), вычисляет кумулятивные вероятности верхних выходов до достижения порога, а затем делает выборку из этого ограниченного набора состояний лексем.

Более «мягкая» версия под названием *выборка top- K* (*top- K sampling*) заключается в том, что в определение функции softmax вводится параметр T , называемый *температурой* (Hinton, Vinyals and Dean, 2015), так что сначала вычисляется

$$y_i = \frac{\exp(a_i/T)}{\sum_j \exp(a_j/T)}, \quad (12.35)$$

и затем производится выборка следующей лексемы из этого модифицированного распределения. При $T = 0$ массив вероятностей сосредоточен на наиболее вероятном состоянии, а все остальные состояния имеют нулевую вероятность, и, следовательно, это превращается в метод жадной выборки. При $T=1$ происходит восстановление немодифицированного распределения softmax, а при $T \rightarrow \infty$ распределение становится равномерным по всем состояниям. При выборе значения в диапазоне $0 < T < 1$ вероятность сосредотачивается в направлении больших значений.

Одна из проблем, связанных с генерацией последовательности, заключается в том, что на этапе обучения модель тренируется на сгенерированной человеком входной последовательности, в то время как при генеративной работе входная последовательность генерируется самой моделью. Это озна-

чает, что модель может отклоняться от распределения последовательностей, наблюдавшегося в процессе обучения.

12.3.3. Кодирующие трансформеры

Далее речь пойдет о кодирующих языковых моделях трансформеров – моделях, которые принимают на вход последовательности и выдают на выходе векторы фиксированной длины, например метки классов. Примером такой модели является *BERT*, что означает *bidirectional encoder representations from transformers* (дву направленный кодировщик представлений на базе трансформеров) (Devlin et al., 2018). Основная идея заключается в предварительном обучении языковой модели на большом корпусе текстов с последующей точной настройкой модели при помощи *трансферного обучения* для широкого спектра последующих задач, каждая из которых требует меньшего набора обучающих данных, специфичных для конкретного приложения. Архитектура кодирующего трансформера показана на рис. 12.18. Такой подход является простым применением рассмотренных ранее слоев трансформера (см. раздел 12.1.7). Здесь флаги с меткой «LSM» обозначают линейное преобразование, обучаемые параметры которого распределены по позициям лексем с последующей функцией активации softmax. Основные различия, по сравнению с моделью декодера, заключаются в том, что входная последовательность не сдвинута вправо, а маскирующая матрица с функцией «упреждающего просмотра» (look ahead) отсутствует, поэтому в каждом слое самовнимания каждая выходная лексема может соответствовать любой из входных лексем.

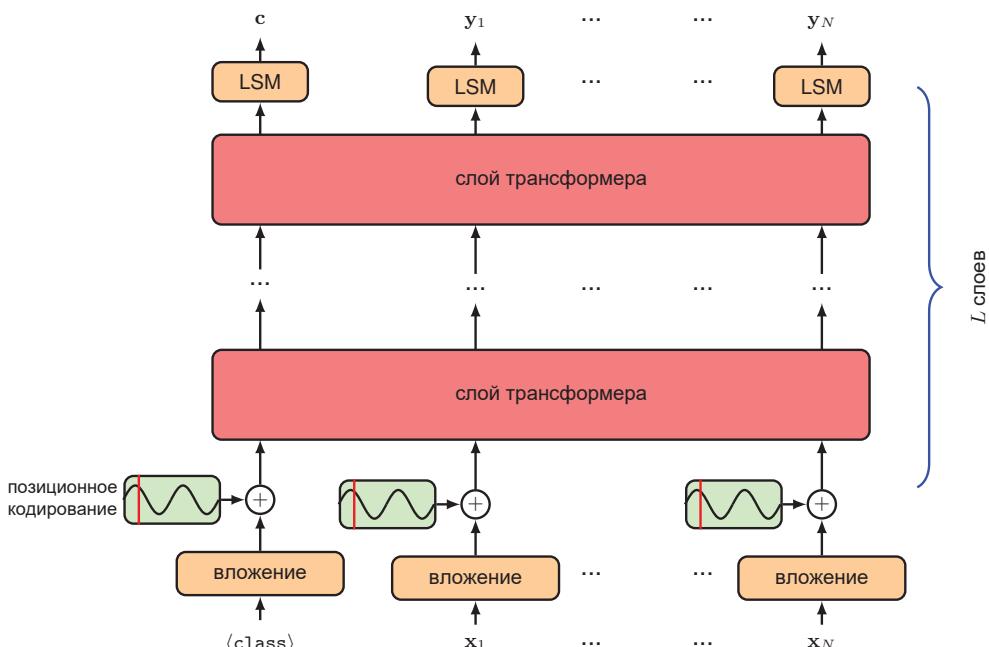


РИС. 12.18 Архитектура модели кодирующего трансформера

Первая лексема каждой входной строки задается специальной лексемой *(class)*, и в процессе предварительного обучения соответствующий выход модели игнорируется. Его роль станет очевидной, когда речь пойдет о тонкой настройке. Модель предварительно обучается посредством подачи на вход последовательностей лексем. Случайно выбранное подмножество лексем, например 15 %, заменяется специальной маскирующей лексемой *(mask)*. Модель обучается предсказывать недостающие лексемы в соответствующих выходных узлах. Это похоже на маскировку (см. раздел 12.2.1), используемую в word2vec для изучения вложений слов. Например, входная последовательность может быть такой:

I *(mask)* across the river to get to the *(mask)* bank (Я *(mask)* через реку, чтобы добраться до *(mask)* берега),

и сеть должна предсказать «swam» (переплыл) в выходном узле 2 и «other» (другого) в выходном узле 10. В этом случае только два выхода вносят вклад в функцию ошибки, а остальные выходы игнорируются.

Термин «двунаправленный» (*bidirectional*) подразумевает, что сеть распознает слова как до, так и после маскируемого слова и может использовать оба эти источника информации для предсказания. Как следствие, в отличие от моделей декодера, нет необходимости сдвигать входы вправо на одно место и маскировать выходы каждого слоя, чтобы они не видели входные лексемы, встречающиеся позже в последовательности. По сравнению с моделью декодера кодировщик менее эффективен, поскольку в качестве обучающих меток используется лишь часть лексем последовательности. Кроме того, модель кодировщика не способна генерировать последовательности.

Процедура замены случайно выбранных лексем на *(mask)* подразумевает несоответствие обучающего набора по сравнению с последующими наборами для тонкой настройки, поскольку последние не будут содержать лексем *(mask)*. Чтобы смягчить возможные проблемы, в работе (Devlin et al., 2018) предложили несколько изменить процедуру, так что из 15 % случайно выбранных лексем 80 % заменяются на *(mask)*, 10 % заменяются на слово, случайно выбранное из словаря, а в 10 % случаев исходные слова сохраняются на входе, но на выходе их все равно необходимо предсказать правильно.

После обучения модели кодировщика можно произвести ее тонкую настройку для решения различных задач. Для этого создается новый выходной слой, форма которого зависит от решаемой задачи. Для задачи классификации текста используется только первая позиция выхода, которая соответствует лексеме *(class)*, всегда появляющейся в первой позиции входной последовательности. Если этот выход имеет размерность D , то к первому выходному узлу добавляется матрица параметров размерности $D \times K$, где K – это количество классов. Эта матрица, в свою очередь, передается K -мерной функции softmax или вектору размерности $D \times 1$ с последующей логистической сигмоидной функцией для $K = 2$. Линейное преобразование выходного сигнала может быть заменено более сложной дифференцируемой моделью, например MLP. Если целью является классификация каждой лексемы входной строки, например ее соотнесение с какой-либо категорией (например,

человек, место, цвет и т. д.), то первый выход игнорируется, а последующие выходы имеют общий линейный слой плюс softmax. Во время тонкой настройки все параметры модели, включая новую матрицу выходов, обучаются методом стохастического градиентного спуска с использованием логарифмической вероятности правильной метки. В качестве альтернативы выход предварительно обученной модели может быть использован в сложной генеративной модели глубокого обучения для таких приложений, как синтез текста с изображением (см. главу 20).

12.3.4. Трансформеры последовательности в последовательность

Для полноты картины вкратце остановимся на третьей категории моделей-трансформеров, объединяющих процедуры кодирования и декодирования, о которых рассказано в оригинальной статье (Vaswani et al., 2017), посвященной трансформерам. Рассмотрим задачу перевода предложения на английском языке в предложение на голландском языке. Можно использовать модель декодера для генерации последовательности лексем (см. раздел 12.3.1), соответствующей голландскому результату, лексема за лексемой, как обсуждалось ранее. Основное отличие заключается в том, что этот выход должен быть определен на основе всей входной последовательности, соответствующей предложению на английском языке. Кодирующий трансформер может применяться для преобразования входной последовательности лексем в соответствующее внутреннее представление, которое обозначим Z . Чтобы включить Z в генеративный процесс для выходной последовательности, воспользуемся модифицированной формой механизма внимания, называемой *перекрестным вниманием* (*cross attention*). Это то же самое, что и самовнимание, за исключением того, что, хотя векторы запросов поступают из генерируемой последовательности (в данном случае выходной последовательности Dutch (голландский)), ключевые векторы и векторы значений поступают из последовательности, представленной Z , как показано на рис. 12.19. Здесь Z обозначает выход секции кодирования. Z определяет ключевые векторы и векторы значений для слоя перекрестного внимания, в то время как векторы запросов определяются в секции декодирования. Возвращаясь к аналогии с сервисом потокового видео, можно сказать, что пользователь посылает свой вектор запроса другой компании, предоставляющей потоковое видео, а та сравнивает его со своим собственным набором ключевых векторов для нахождения наилучшего соответствия, после чего возвращает соответствующий вектор значения в виде фильма.

При объединении модулей кодирования и декодирования получается архитектура модели, показанная на рис. 12.20. Чтобы не загромождать схему, входные лексемы показаны одним блоком, то же самое относится и к выходным лексемам. Векторы позиционного кодирования добавляются к входным лексемам как в секции кодирования, так и в секции декодирования. Каждый слой в кодирующем модуле соответствует структуре, показанной на рис. 12.9, а каждый слой перекрестного внимания имеет вид, показанный на рис. 12.19. Модель можно обучить на парных входных и выходных предложениях.

РИС. 12.19 Схематическое изображение одного слоя перекрестного внимания, который используется в секции декодера трансформера последовательности в последовательность

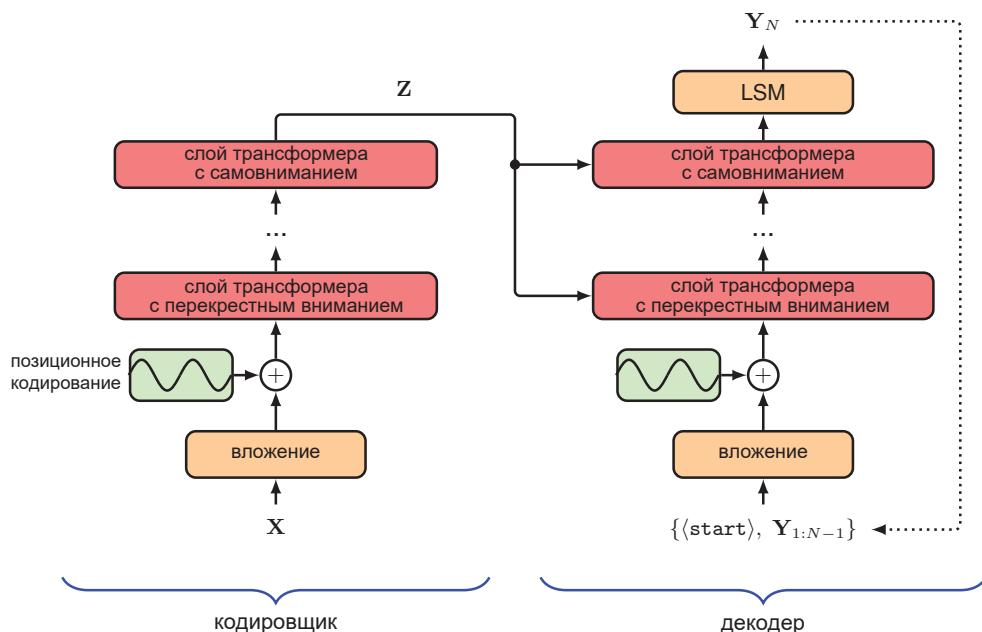
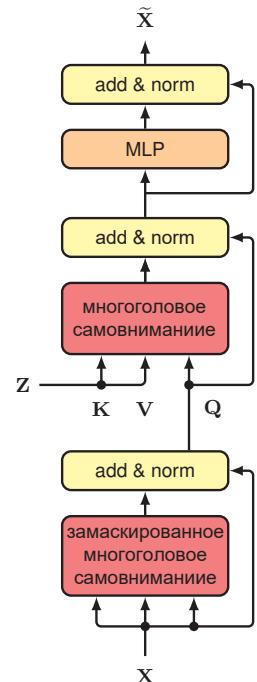


РИС. 12.20 Схематическое изображение трансформера последовательности в последовательность

12.3.5. Большие языковые модели

Важнейшим событием последнего времени в области машинного обучения стало создание очень больших нейронных сетей на основе трансформеров для обработки естественного языка, которые получили название *больших языковых моделей* (*large language models, LLM*). Под «большими» здесь подразумевается количество весовых параметров и параметров смещения в сети, которые на момент написания статьи могли достигать примерно одного триллиона (10^{12}). Обучение таких моделей обходится недешево, а стимулом для их создания выступают их экстраординарные возможности.

Помимо доступа к большим массивам данных, обучение все более масштабных моделей становится все более удобным благодаря появлению массивно-параллельного обучающего оборудования на базе графических процессоров (GPU) и подобных им процессоров, объединенных в масштабные кластеры, с быстрыми межблочными соединениями и большим объемом собственной памяти. Архитектура трансформеров сыграла ключевую роль в разработке подобных моделей, поскольку она позволяет использовать такое оборудование максимально эффективно. Зачастую увеличение размера обучающего набора данных наряду с соизмеримым увеличением количества параметров модели приводит к повышению производительности, которое опережает усовершенствование архитектуры или другие способы включения дополнительных знаний предметной области (Sutton, 2019; Kaplan et al., 2020). Например, впечатляющий рост производительности моделей семейства GPT (Radford et al., 2019; Brown et al., 2020; OpenAI, 2023) в последующих поколениях обусловлен преимущественно увеличением масштаба. Подобное повышение производительности привело к появлению новой разновидности закона Мура, согласно которому количество вычислительных операций, необходимых для обучения современной модели машинного обучения, растет экспоненциально примерно с 2012 года, а период удвоения производительности составляет порядка 3,4 месяца (рис. 1.16).

Первые языковые модели обучались методом контролируемого обучения. Например, для создания системы перевода обучающий набор включал в себя согласованные пары предложений на двух языках. Однако основное ограничение контролируемого обучения связано с тем, что данные для получения маркированных примеров, как правило, должны быть подготовлены человеком, а это сильно ограничивает количество доступных данных и для достижения разумной производительности требует активного использования индуктивных смещений, таких как создание признаков и соблюдение архитектурных ограничений.

Поэтому большие языковые модели обучаются методом самоконтроля на очень больших наборах данных, содержащих текст, а также последовательности других лексем иного формата, например компьютерный код (см. раздел 12.3.1). Ранее уже рассматривалось обучение декодирующего трансформера на последовательностях лексем, где каждая лексема выступает в роли маркированного целевого примера, а предыдущая последователь-

ность – в качестве входных данных для обучения условному распределению вероятности. Такая «самомаркировка» (self-labelling) существенно расширяет диапазон доступных обучающих данных и, следовательно, позволяет использовать глубокие нейронные сети с большим числом параметров.

Использование самоконтролируемого обучения привело к смене парадигмы, когда сначала происходит предварительное обучение большой модели на немаркированных данных, а затем производится ее тонкая настройка с помощью контролируемого обучения на основе гораздо меньшего набора маркированных данных. По сути, это форма трансферного обучения, и одна и та же предварительно обученная модель может быть использована для нескольких приложений «нисходящего» уровня. Модель с широкими возможностями, которая впоследствии может быть доработана для решения конкретных задач, называется *базовой моделью* (*foundation model*) (Bommasani et al., 2021).

Точная настройка может быть выполнена путем добавления дополнительных слоев к выходам сети или путем обновления параметров в нескольких последних слоях с последующим использованием маркированных данных для обучения этих последних слоев. Веса и смещения в основной модели на этапе тонкой настройки можно оставить без изменений или провести небольшую адаптацию. Как правило, расходы на тонкую настройку минимальны по сравнению с затратами на предварительное обучение.

Один из очень эффективных подходов к тонкой настройке называется *адаптацией низкого ранга* (*low-rank adaptation, LoRA*) (Hu et al., 2021). Этот подход основывается на выводах специалистов о том, что обученная модель с избыточными параметрами имеет низкую собственную размерность по отношению к методу точной настройки, т. е. изменения параметров модели при точной настройке относятся к многообразию, размерность которого намного меньше общего числа обучаемых параметров в модели (Aghajanyan, Zettlemoyer and Gupta, 2020). Решение LoRA использует эту возможность, замораживая веса исходной модели и добавляя дополнительные обучаемые весовые матрицы в каждый слой трансформера в виде произведений низкого ранга. Обычно изменяются только веса слоев внимания, в то время как веса MLP-слоев остаются фиксированными. Рассмотрим матрицу весов \mathbf{W}_0 размерностью $D \times D$, которая может представлять собой матрицу запросов, ключей или значений, и в которой матрицы от нескольких голов внимания рассматриваются как одна матрица. Затем вводится параллельный набор весов, определяемый произведением двух матриц \mathbf{A} и \mathbf{B} размером $D \times R$ и $R \times D$ соответственно, как схематично показано на рис. 12.21. Здесь дополнительные веса, заданные матрицами \mathbf{A} и \mathbf{B} , адаптируются в процессе точной настройки, а их произведение \mathbf{AB} добавляется к исходной матрице для последующего вывода. Этот слой генерирует выходной сигнал в виде $\mathbf{XW}_0 + \mathbf{XAB}$. Число параметров в дополнительной матрице весов \mathbf{AB} равно $2RD$ по сравнению с D^2 параметрами в исходной весовой матрице \mathbf{W}_0 , и поэтому если $R \ll D$, то число параметров, которые необходимо адаптировать при тонкой настройке, значительно меньше, чем в исходном трансформере. На практике это позво-

ляет сократить количество подлежащих обучению параметров в 10 000 раз. После завершения точной настройки дополнительные веса можно добавить к исходным матрицам весов и получить новую матрицу весов:

$$\hat{\mathbf{W}} = \mathbf{W}_0 + \mathbf{AB}. \quad (12.36)$$

По этой причине в процессе вывода не возникает дополнительных вычислительных затрат по сравнению с запуском исходной модели, поскольку обновленная модель имеет тот же размер, что и исходная.

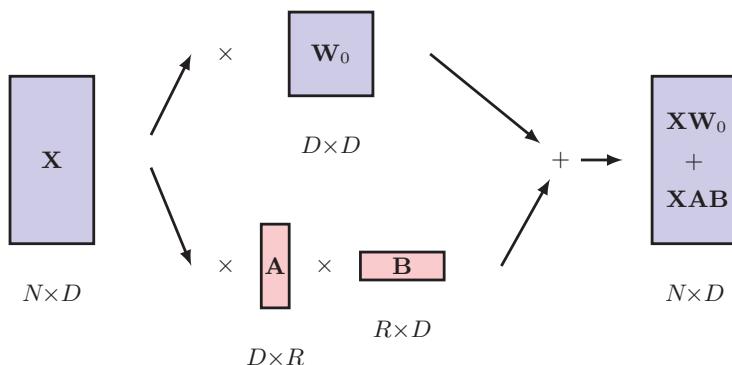


РИС. 12.21 Схематическая иллюстрация адаптации низкого ранга, показывающая матрицу весов \mathbf{W}_0 из одного из слоев внимания в предварительно обученном трансформере

По мере увеличения масштабов и роста производительности языковых моделей нужда в точной настройке уменьшалась, и в настоящее время генеративные языковые модели способны решать широкий спектр задач посредством простого взаимодействия с текстом. Например, если задать текстовую строку

English: the cat sat on the mat. French: (Английский: кошка села на коврик. Французский:)

в качестве входной последовательности, то авторегрессионная языковая модель может продолжать генерировать последующие лексемы до тех пор, пока не будет сгенерирована лексема `(stop)`, где вновь сгенерированные лексемы представляют собой перевод на французский язык. Обратите внимание, что модель не была специально обучена переводу, а научилась этому в результате обучения на обширном корпусе данных, включающем несколько языков.

Пользователь может взаимодействовать с такими моделями посредством диалога на естественном языке, что делает их очень доступными для широкой аудитории. Для улучшения пользовательских впечатлений и качества генерируемых результатов были разработаны методы точной настройки больших языковых моделей путем оценки генерируемых результатов человеком с использованием таких методов, как *обучение с подкреплением и обратной связью от человека* (*reinforcement learning through human feedback, RLHF*) (Chris-

tiano et al., 2017). Подобные методы помогли создать большие языковые модели с впечатляюще простыми в использовании разговорными интерфейсами, в частности систему от OpenAI под названием ChatGPT.

Последовательность вводимых пользователем лексем называется *запросом (prompt)*. К примеру, запрос может состоять из начальных слов рассказа, который модель должна завершить. Или это может быть вопрос, на который модель должна дать ответ. Используя различные запросы, одна и та же обученная нейронная сеть может решать широкий спектр задач, например генерировать компьютерный код на основе простого текстового запроса или писать рифмованные стихи по заданию. Производительность модели теперь зависит от формы запроса, что привело к появлению новой области, называемой *инженерией запросов (prompt engineering)* (Liu et al., 2021), целью которой является разработка оптимальной формы запроса, приводящей к высококачественному результату для последующей задачи. Поведение модели также может быть изменено путем адаптации пользовательского запроса перед подачей его в языковую модель путем предварительного добавления к пользовательскому запросу дополнительной последовательности лексем, называемой *префиксным запросом (prefix prompt)* для изменения формы вывода. Например, префиксный запрос может состоять из инструкций, выраженных на стандартном английском языке и указывающих сети не включать в вывод оскорбительные выражения.

В результате модель способна решать новые задачи за счет нескольких примеров в подсказке без необходимости адаптации параметров модели. Это пример так называемого *обучения на ограниченных примерах (few-shot learning)*.

Современные модели, такие как GPT-4, стали настолько мощными, что проявляют выдающиеся свойства, которые уже называют первыми признаками искусственного общего интеллекта (Bubeck et al., 2023). Они становятся стимулом для развития новой волны технологических разработок. Более того, возможности этих моделей постоянно совершенствуются впечатляющими темпами.

12.4. Мультиомодальные трансформеры

Несмотря на то что изначально трансформеры были разработаны в качестве альтернативы рекуррентным сетям для обработки последовательных языковых данных, они получили широкое распространение практически во всех областях глубокого обучения. Они доказали, что являются моделями общего назначения, поскольку делают очень мало предположений о входных данных, в отличие, например, от сверточных сетей (см. главу 10), которые делают сильные предположения об эквивариациях и локальности. Благодаря своей универсальности трансформеры уже стали самыми современными решениями для работы с различными категориями данных, включая тек-

стовые данные, изображения, видео, облака точек и аудиоданные, при этом они используются как для дискриминационных, так и для генеративных целей в каждой из этих областей. Основная архитектура слоя трансформации остается относительно постоянной как с течением времени, так и в разных приложениях. Поэтому ключевые инновации, позволившие использовать трансформеры в других областях, помимо естественного языка, в значительной степени были направлены на представление и кодирование входов и выходов.

Одно из главных преимуществ использования единой архитектуры, способной обрабатывать множество различных типов данных, заключается в возможности относительно простого выполнения мультимодальных вычислений. В данном контексте под мультимодальными понимаются задачи с сочетанием двух или более различных типов данных – либо на входе, либо на выходе, либо в обоих случаях. Например, можно сгенерировать изображение из текстовой подсказки или разработать робота, способного использовать информацию от нескольких датчиков, таких как камеры, радары и микрофоны. Важно отметить, что если можно выполнить лексическую обработку входных данных и декодировать выходные лексические данные, то, скорее всего, можно использовать трансформер.

12.4.1. Визуальные трансформеры

Трансформеры с большим успехом применяются в области компьютерного зрения и демонстрируют самые высокие результаты при решении многих задач. Наиболее распространенным выбором для дискриминационных задач является стандартный кодирующий трансформер, и в области компьютерного зрения такой принцип известен как *визуальный трансформер* (*vision transformer*, ViT) (Dosovitskiy et al., 2020). В случае использования трансформера необходимо определиться со способом преобразования входного изображения в лексемы, и самый простой вариант подразумевает применение каждого пикселя в качестве лексемы на основе линейной проекции. Однако объем памяти для стандартной реализации трансформера растет квадратично с ростом числа входных лексем, поэтому такой подход в целом непрактичен. Вместо этого наиболее распространенным подходом к лексической обработке является разбиение изображения на множество фрагментов одинакового размера. Предположим, что изображения имеют размерность $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, где H и W – это высота и ширина изображения в пикселях, а C – это количество каналов (обычно $C = 3$ для цветов R, G и B). Каждое изображение разбивается на непересекающиеся фрагменты размером $P \times P$ (где $P = 16$ – это наиболее распространенный случай) и затем «уплотняется» в одномерный вектор с представлением $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 C)}$, где $N = HW/P^2$ – это общее количество фрагментов для одного изображения. Архитектура ViT представлена на рис. 12.22. Здесь обучаемая лексема $\langle \text{class} \rangle$ включена в качестве дополнительного входа, а связанный с ней выход преобразуется линейным слоем с активацией softmax, обозначаемой LSM, для получения конечного вектора классов на выходе \mathbf{c} .

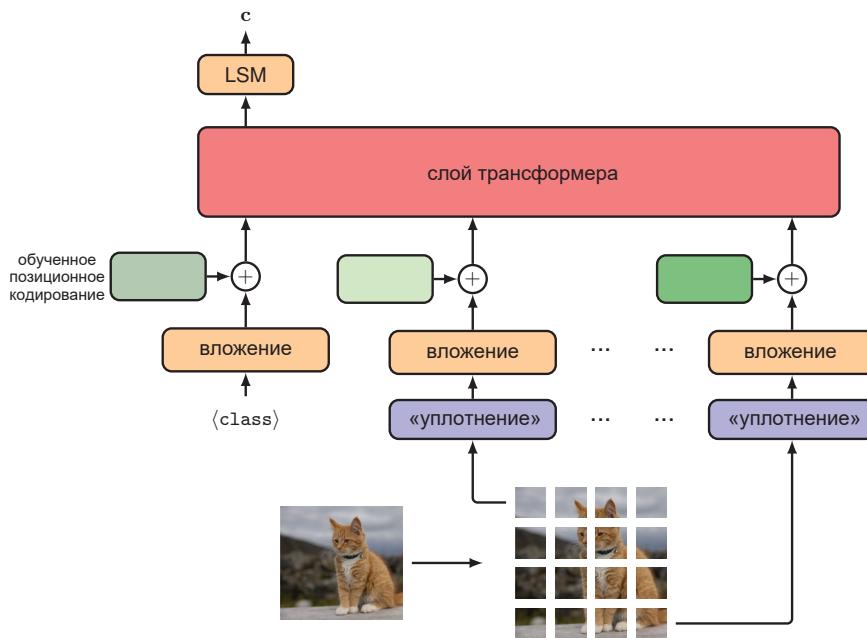


РИС. 12.22 Иллюстрация архитектуры визуального трансформера для задачи классификации

Другой подход к лексической обработке заключается в прохождении изображения через небольшую сверточную нейронную сеть (CNN) (см. главу 10). Это позволяет уменьшить выборку изображения для получения управляемого количества лексем, каждая из которых представлена одним из выходов сети. Например, типичная архитектура кодировщика ResNet18 уменьшает дискретизацию изображения в 8 раз по высоте и ширине, что дает в 64 раза меньше лексем, чем пикселей.

Также необходимо найти способ кодирования позиционной информации в лексемах. Можно построить явные позиционные вложения, которые кодируют двумерную позиционную информацию фрагментов изображения, но на практике это обычно не способствует повышению производительности, поэтому чаще всего применяют только обученные позиционные вложения. В отличие от трансформеров для естественного языка визуальные трансформеры обычно принимают на вход фиксированное количество лексем, что позволяет избежать проблемы неспособности обученных позиционных кодировок к обобщению по входным данным другого размера.

В сравнении с CNN визуальный трансформер имеет совершенно иную архитектурную структуру. Хотя в модели CNN заложены сильные индуктивные смещения, единственное двумерное индуктивное смещение в визуальном трансформере связано с фрагментами, используемыми для лексической разметки входных данных. Поэтому трансформеру, как правило, требуется больше обучающих данных, чем сопоставимой CNN, поскольку ему приходится

изучать геометрические свойства изображений с нуля. Однако из-за полного отсутствия строгих представлений о структуре входных данных трансформеры зачастую способны обеспечить более высокую точность сходимости. Это еще раз подтверждает наличие компромисса между индуктивным смещением и масштабом обучающих данных (Sutton, 2019).

12.4.2. Генеративные визуальные трансформеры

В области обработки языкового материала наиболее впечатляющие результаты были получены при использовании трансформеров в качестве авторегрессионной генеративной модели для синтеза текста. Поэтому естественно задаться вопросом, можно ли использовать трансформеры для синтеза реалистичных изображений. Поскольку естественный язык по своей сути является последовательным, он вполне вписывается в рамки авторегрессии, в то время как изображения не характеризуются естественным упорядочиванием пикселей, поэтому не совсем очевидно, что их авторегрессионное декодирование будет полезным. Однако любое распределение может быть разложено в произведение условий при условии предварительного определения некоторой упорядоченности переменных (см. раздел 11.1.2). Получается, что совместное распределение по упорядоченным переменным $\mathbf{x}_1, \dots, \mathbf{x}_N$ можно записать в виде

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1}). \quad (12.37)$$

Эта факторизация является совершенно общей и не накладывает никаких ограничений на форму отдельных условных распределений $p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$.

РИС. 12.23 Иллюстрация растровой развертки, определяющей конкретное линейное упорядочивание пикселей в двухмерном изображении

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	\mathbf{x}_4
\mathbf{x}_5	\mathbf{x}_6	\mathbf{x}_7	\mathbf{x}_8
\mathbf{x}_9	\mathbf{x}_{10}	\mathbf{x}_{11}	\mathbf{x}_{12}
\mathbf{x}_{13}	\mathbf{x}_{14}	\mathbf{x}_{15}	\mathbf{x}_{16}

Для обозначения n -го пикселя в виде трехмерного вектора значений RGB на изображении можно выбрать \mathbf{x}_n . Теперь необходимо определиться с порядком расположения пикселей, и один из широко используемых вариантов называется *растровой разверткой* (*raster scan*), как показано на рис. 12.23. Схематичная иллюстрация создания изображения с помощью авторегрессионной модели на основе растрового упорядочивания показана на рис. 12.24. Первый пиксель берется из маргинального распределения $p(\mathbf{x}_{11})$, второй – из

условного распределения $p(\mathbf{x}_{12} | \mathbf{x}_{11})$ и т. д. в порядке растрового сканирования до получения полного изображения.

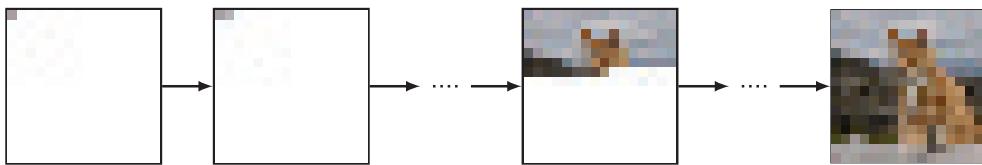


РИС. 12.24 Иллюстрация процесса выборки изображения с использованием авторегрессионной модели

Обратите внимание, что использование авторегрессионных генеративных моделей для обработки изображений появилось раньше трансформеров. Например, в PixelCNN (Oord et al., 2016) и PixelRNN (Oord, Kalchbrenner and Kavukcuoglu, 2016) используются маскированные слои свертки с сохранением условной независимости, определяемой для каждого пикселя соответствующим членом в правой части выражения 12.37.

В задачах дискриминации хорошо работают представления изображения с использованием непрерывных величин. Однако намного лучшие результаты при генерации изображений дает использование дискретных представлений. Непрерывные условные распределения, обучаемые методом максимального правдоподобия, такие как гауссовые распределения, для которых функция отрицательного логарифмического правдоподобия (см. раздел 4.2) является функцией суммы квадратов ошибок, имеют тенденцию к усреднению обучающих данных, что приводит к размытию изображений. И наоборот, дискретные распределения легко справляются с мультимодальностью. Например, одно из условных распределений $p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$ в (12.37) можно обучить тому, что пиксель может быть либо черным, либо белым, в то время как регрессионная модель сможет вычислить его как серый.

Однако работа с дискретными пространствами также сопряжена с определенными трудностями. Значения R, G и B пикселей изображения обычно представлены с точностью не менее 8 бит, так что каждый пиксель имеет $2^{24} \approx 16$ млн возможных значений. Обучение условного распределения softmax в таком пространстве высокой размерности является нереалистичной задачей.

Для решения проблемы высокой размерности используется техника *векторного квантования* (*vector quantization*) (см. раздел 15.1.1), которую можно рассматривать в качестве одной из форм сжатия данных. Предположим, имеется набор векторов данных $\mathbf{x}_1, \dots, \mathbf{x}_N$, каждый размерности D , которые могут, например, представлять пиксели изображения, и затем вводится набор из K векторов кодовой книги $\mathcal{C} = \mathbf{c}_1, \dots, \mathbf{c}_K$ также размерности D , где в большинстве случаев $K \ll D$. Теперь каждый вектор данных аппроксимируем его ближайшим вектором кодовой книги в соответствии с определенной метрикой сходства, обычно евклидовым расстоянием, так что

$$\mathbf{x}_n \rightarrow \underset{\mathbf{c}_k \in \mathcal{C}}{\operatorname{argmin}} \|\mathbf{x}_n - \mathbf{c}_k\|^2. \quad (12.38)$$

Поскольку существует K векторов кодовой книги, каждый \mathbf{x}_n можно представить одномоментно закодированным K -мерным вектором, а поскольку можно выбрать значение K , появляется возможность контролировать компромисс между более точным представлением данных, используя большее значение K , и большим сжатием, используя меньшее значение K .

Поэтому можно взять пиксели исходного изображения и отобразить их в низкоразмерное пространство кодовой книги. Затем можно обучить авторегрессионный трансформер генерировать последовательность векторов кодовых книг, и эту последовательность можно отобразить обратно в пространство исходного изображения, заменив каждый индекс кодовой книги k соответствующим D -мерным вектором кодовой книги \mathbf{c}_k .

Авторегрессионные трансформеры впервые были использованы для обработки изображений в ImageGPT (Chen, Radford, et al., 2020). Здесь каждый пиксель рассматривается как один из дискретного набора трехмерных векторов цветовой кодовой книги, каждый из которых соответствует кластеру в кластеризации цветового пространства по алгоритму *K-средних* (*K-means*, см. раздел 15.1). Таким образом, одноточечное кодирование дает дискретные лексемы, аналогичные лексемам языка, и позволяет обучать трансформер так же, как и языковые модели, с целью классификации по следующей лексеме. Это мощная задача для обучения представлений с целью последующей точной настройки (см. раздел 6.3.3), так же как и при моделировании языка.

Однако непосредственное использование отдельных пикселей в качестве лексем может привести к большим вычислительным нагрузкам, поскольку для каждого пикселя требуется прямой проход, а это значит, что и обучение, и вывод плохо масштабируются с увеличением разрешения изображения. Кроме того, использование отдельных пикселей в качестве входных данных означает необходимость использования изображений с низким разрешением для получения разумной длины контекста при последующем декодировании пикселей в растровой развертке. Как уже было показано на примере модели ViT, предпочтительнее использовать в качестве лексем не пиксели, а фрагменты изображения, так как это может привести к значительному уменьшению количества лексем и, следовательно, облегчает работу с изображениями более высокого разрешения. Как и прежде, необходимо работать с дискретным пространством значений лексем из-за потенциальной мульти modalности условных распределений. И здесь вновь возникает проблема размерности, которая для фрагментов гораздо серьезнее, чем для отдельных пикселей, поскольку размерность растет экспоненциально по отношению к количеству пикселей во фрагменте. Например, даже при наличии всего двух возможных лексем пикселей, представляющих черный и белый цвета, и фрагментов размером 16×16 получится словарь лексем фрагментов размером $2^{256} \approx 10^{77}$.

Для решения проблемы размерности вновь обратимся к векторному квантованию. Векторы кодовой книги могут быть выучены из набора изображе-

ний с помощью простых алгоритмов кластеризации, таких как K -среднее, или с помощью более сложных методов, таких как полностью сверточные сети (Oord, Vinyals and Kavukcuoglu, 2017; Esser, Rombach and Ommer, 2020) или даже визуальные трансформеры (Yu et al., 2021). Одна из проблем с обучением сопоставлению каждого фрагмента с дискретным набором кодов и обратно заключается в том, что векторное квантование – это не дифференцируемая операция. К счастью, здесь можно использовать технику, называемую *прямой оценкой градиента* (*straight-through gradient estimation*) (Bengio, Léonard and Courville, 2013), которая является простой аппроксимацией, позволяющей просто копировать градиенты через недифференцируемую функцию во время обратного распространения.

Использование авторегрессионных трансформеров для генерации изображений можно расширить на видео, рассматривая его как одну длинную последовательность этих квантованных векторных лексем (Rakhimov et al., 2020; Yan et al., 2021; Hu et al., 2023).

12.4.3. Аудиоданные

Теперь обратимся к теме применения трансформеров для обработки аудиоданных. Звук обычно хранится в виде формы волны, полученной путем измерения амплитуды давления воздуха через регулярные временные интервалы. Несмотря на то что эту сигнальную форму можно напрямую использовать в качестве входных данных для модели глубокого обучения, на практике более эффективным вариантом оказывается ее предварительная обработка в виде *спектrogramмы mel*. Это матрица, столбцы которой представляют собой временные шаги, а строки соответствуют частотам. Частотные диапазоны соответствуют стандартной шкале, которая была выбрана путем субъективной оценки для получения равных различий в восприятии между последовательными частотами («mel» происходит от слова «melody», мелодия). Пример спектrogramмы mel показан на рис. 12.25.

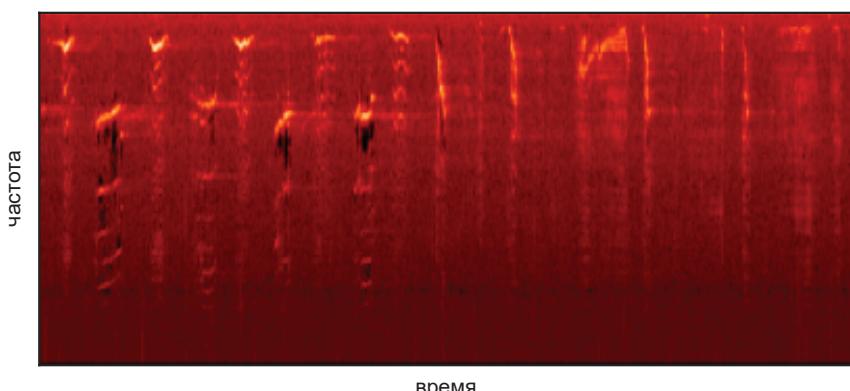


РИС. 12.25 Пример спектrogramмы mel песни горбатого кита. [Авторские права на исходные данные © 2013-2023, librosa development team]

Одним из применений трансформеров в области работы со звуком является классификация, при которой сегменты звука относят к одной из нескольких заранее определенных категорий. В частности, набор данных *AudioSet* (Gemmeke et al., 2017) считается широко известным эталоном. Он включает такие классы, как «автомобиль», «животное» и «смех». До разработки трансформеров самый передовой подход к классификации аудио был основан на анализе спектрограмм *mel* в виде изображений и использовании их в качестве входных данных для сверточной нейронной сети (CNN) (см. главу 10). Несмотря на то что CNN хорошо справляется с определением локальных связей, ее недостатком является проблема определения более дальних зависимостей, которые могут быть важны при обработке аудио.

Подобно тому как трансформеры вытеснили RNN в качестве передового метода обработки естественного языка, они также пришли на смену CNN для таких задач, как классификация аудио. Например, модель кодирования с помощью трансформера, структурно идентичного используемому как для языка, так и для зрения, как показано на рис. 12.18, может быть использована для прогнозирования класса входного аудиосигнала (Gong, Chung and Glass, 2021). Здесь спектрограмма *mel* рассматривается в качестве изображения, которое затем преобразуется в лексемы. Для этого изображение разбивается на фрагменты по аналогии с визуальными трансформерами, возможно, с некоторым перекрытием, чтобы не потерять важные взаимосвязи между соседними фрагментами. Затем каждый фрагмент преобразуется в одномерный массив, в этом случае его длина составляет 256. Затем к каждой лексеме добавляется уникальная позиционная кодировка, прикрепляется определенная лексема *<class>*, и все лексемы проходят через кодирующий трансформер. Выходная лексема, соответствующая входной лексеме *<class>* из последнего слоя трансформера, далее может быть декодирована с помощью линейного слоя, за которым следует функция активации softmax, и вся модель может быть обучена сквозным образом с использованием метода перекрестных энтропийных потерь.

12.4.4. Преобразование текста в речь

Классификация не является единственной задачей, которую глубокое обучение и, в частности, архитектура трансформеров коренным образом изменили в сфере обработки звука. Успех трансформеров в области синтеза речи, имитирующей голос определенного человека, – еще одна демонстрация их универсальности, а возможность их применения для решения этой задачи – наглядный пример того, как можно использовать трансформеры в совершенном контексте.

Генерирование речи, соответствующей заданному отрывку текста, называют *синтезом речи по тексту* (*text-to-speech synthesis*). При наиболее типичном подходе для синтеза речи необходимо собрать записи речевых фрагментов определенного человека и обучить контролируемую регрессионную модель для прогноза речевого вывода, возможно, в виде спектрограммы на основе

соответствующей транскрипции текста. В процессе вывода текст, для которого необходимо синтезировать речь, подается на вход, а полученная спектроGRAMМА *mel* может быть декодирована обратно в форму звукового сигнала, поскольку она является фиксированным отображением.

Однако у такого метода есть несколько существенных недостатков. Во-первых, если прогнозировать речевой поток на низком уровне, например, с помощью составных частей слов, известных как *фонемы (phonemes)*, то для придания плавности звучанию полученных предложений необходим более широкий контекст. Однако при прогнозировании более длинных сегментов пространство возможных входных данных значительно увеличивается, и для достижения хорошего обобщения может потребоваться слишком большой объем обучающих данных. Во-вторых, этот подход не позволяет переносить знания между различными дикторами, поэтому для каждого нового человека потребуется большое количество данных. Наконец, в действительности эта проблема представляет собой задачу генеративного моделирования, поскольку существует множество правильных речевых выходов для определенной пары из диктора и текста (см. раздел 4.2), поэтому регрессия может оказаться неприемлемой, поскольку имеет тенденцию к усреднению по целевым значениям.

Если же рассматривать аудиоданные так же, как естественный язык, и рассматривать преобразование текста в речь в качестве задачи условного моделирования языка, то обучение модели будет осуществляться примерно так же, как и в случае с большими языковыми моделями для работы с текстами. Есть две основные особенности реализации модели, которые требуют особого внимания. Первая – как проводить лексическую обработку обучающих данных и декодировать прогнозы, а вторая – как настроить модель на речь диктора.

Одним из способов синтеза речи по тексту, в котором используются трансформеры и методы языкового моделирования, является *Vall-E* (Wang et al., 2023). Новый текст может быть преобразован в речь, произнесенную голосом нового диктора, всего лишь через несколько секунд звучания речи этого человека. Речевые данные преобразуются в последовательность дискретных лексем (см. раздел 12.4.2) из обучаемого словаря или *кодовой книги*, полученной с помощью векторного квантования, и такие лексемы можно рассматривать как аналог одноточечных кодированных лексем из области естественного языка. На вход подаются текстовые лексемы из отрывка текста, в то время как целевые выходные данные для обучения состоят из соответствующих речевых лексем. К входным текстовым лексемам добавляются дополнительные речевые лексемы из короткого сегмента не связанный по смыслу речи одного и того же диктора, как показано на рис. 12.26. На вход модели трансформера подаются стандартные текстовые лексемы, которые подсказывают модели, какие слова должна содержать синтезированная речь, а также акустические лексемы, определяющие стиль диктора и информацию о тоне. Выборка выходных лексем модели декодируется обратно в речь с помощью обученного декодера. Для наглядности позиционные кодировки и линейные проекции на схеме не показаны. Включение примеров от разных дикторов позволяет

системе научиться зачитывать отрывок текста с имитацией голоса, представленного дополнительными речевыми лексемами. После обучения системе можно предоставить новый текст вместе со звуковыми лексемами из короткого фрагмента речи нового диктора, и полученные выходные лексемы можно декодировать с помощью той же кодовой книги, которая использовалась во время обучения, чтобы создать форму нового речевого сигнала. Это позволяет системе синтезировать речь, соответствующую входному тексту, голосом нового диктора.

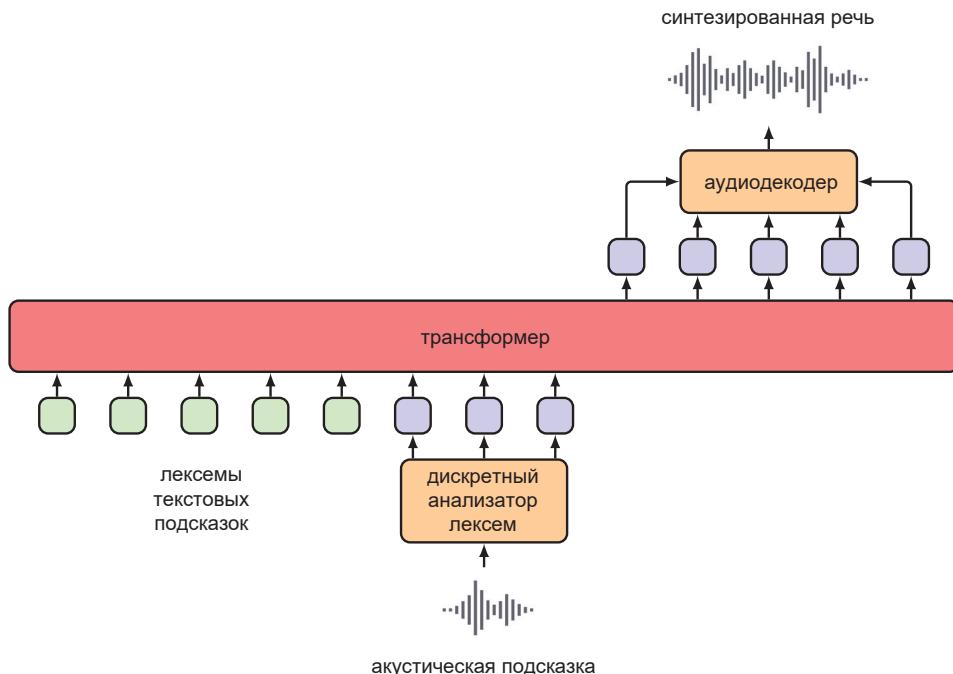


РИС. 12.26 Схема архитектуры высокого уровня Vall-E

12.4.5. Визуальные и языковые трансформеры

Ранее уже рассматривался процесс генерации дискретных лексем для текста, аудио и изображений, поэтому естественным следующим шагом будет выяснить, можно ли обучить модель с входными лексемами одной категории и выходными лексемами другой, а также можно ли использовать комбинацию различных моделей в качестве входных или выходных данных или и того, и другого. В качестве наиболее широко распространенного примера рассмотрим комбинацию текстовых и визуальных данных, но, в принципе, обсуждаемые здесь подходы могут быть применены и к другим комбинациям входных и выходных данных.

Первое требование – это наличие большого набора данных для обучения. Благодаря появлению массива данных LAION400M (Schuhmann et al.,

2021) исследования в области генерации изображений по тексту и создания текстовых подписей к изображениям значительно ускорились, подобно тому как ImageNet сыграл решающую роль в разработке моделей глубокой классификации изображений. Генерация изображений по тексту во многом похожа на безусловную генерацию изображений, рассмотренную ранее, за исключением того, что модель может принимать на вход текстовую информацию, которая определяет условия процесса генерации. При использовании трансформеров такая задача решается достаточно просто, поскольку при декодировании каждой лексемы изображения текстовые лексемы можно использовать в качестве дополнительного ввода.

Этот подход также можно рассматривать как решение вопроса преобразования текста в изображение в виде задачи языкового моделирования из последовательности в последовательность, например машинного перевода, за исключением того, что в этом случае целевыми лексемами являются дискретные лексемы изображений, а не языковые лексемы. Поэтому имеет смысл выбрать полную модель трансформера с кодированием и декодированием, как показано на рис. 12.20, где X соответствует лексемам входного текста, а Y – лексемам выходного изображения. Именно такой подход используется в модели под названием *Parti* (Yu et al., 2022), где трансформер масштабируется до 20 млрд параметров и демонстрирует постоянное улучшение производительности по мере увеличения размера модели.

Множество исследований было также посвящено использованию предварительно обученных языковых моделей и их модификации или точной настройке таким образом, чтобы они могли принимать на вход визуальные данные (Alayrac et al., 2022; Li et al., 2022). Эти методы используют преимущественно специализированные архитектуры, а также лексемы изображений с непрерывным значением, поэтому они не совсем подходят для генерирования визуальных данных. Более того, они не могут быть использованы напрямую при необходимости включения новых видов данных, таких как звуковые лексемы. Хотя это уже шаг к мультимодальности, в идеале желательно использовать как текстовые, так и графические лексемы в качестве входных и выходных данных. Самый простой подход – рассматривать все в виде последовательности лексем, как если бы это был естественный язык, но со словарем, который представляет собой объединение словаря языковых лексем и кодовой книги лексем изображений. Тогда любой поток аудио- и визуальных данных можно рассматривать всего лишь в качестве последовательности лексем.

В моделях CM3 (Aghajanyan et al., 2022) и CM3Leon (Yu et al., 2023) для обучения на HTML-документах, содержащих изображения и текст из онлайн-источников, используется разновидность языкового моделирования. В сочетании с большим количеством обучающих данных и масштабируемой архитектурой модели стали очень мощными. Более того, мультимодальная природа обучения означает, что модели получаются очень гибкими. Такие модели способны решать множество задач, которые в противном случае могли бы потребовать специфических архитектур моделей и режимов обучения,

например генерация текста в изображение, создание подписей к изображениям, редактирование изображений, дополнение текста и многое другое, включая все, на что способна обычная языковая модель. Примеры работы модели CM3Leon над выполнением различных задач показаны на рис. 12.27.

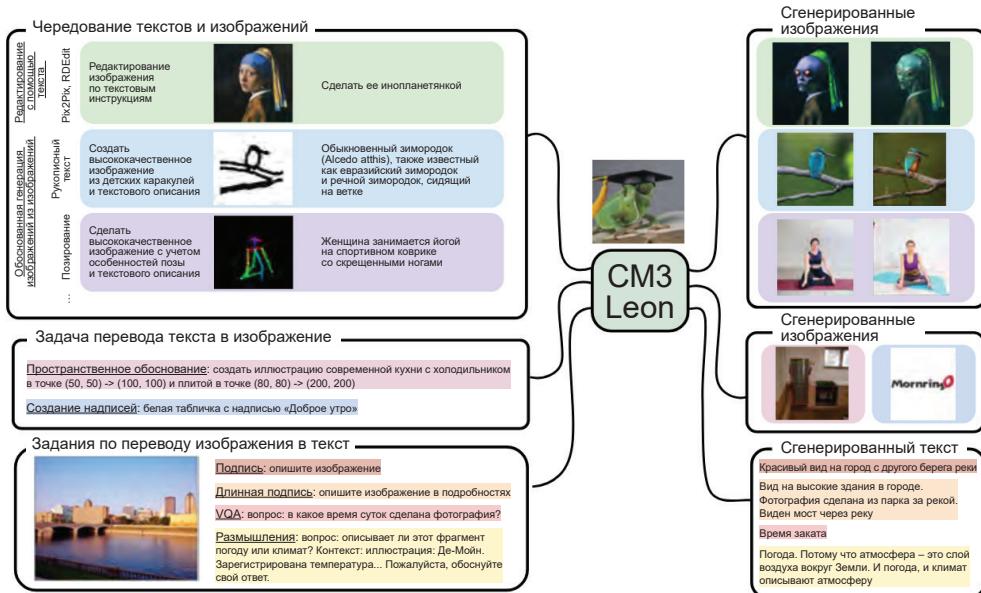


РИС. 12.27 Примеры выполнения различных задач моделью CM3Leon в совместном пространстве текста и изображений. [Из (Yu et al., 2023) с разрешения авторов]

Упражнения

12.1 (**) Рассмотрим набор коэффициентов a_{nm} для $m = 1, \dots, N$, где

$$a_{nm} > 0, \quad (12.39)$$

$$\sum_m a_{nm} = 1. \quad (12.40)$$

Используя множитель Лагранжа (см. приложение С), докажите, что коэффициенты также должны удовлетворять условию

$$a_{nm} \leq 1 \text{ для } n = 1, \dots, N. \quad (12.41)$$

12.2 (*) Докажите, что функция softmax (12.5) удовлетворяет ограничениям (12.3) и (12.4) для любых значений векторов $\mathbf{x}_1, \dots, \mathbf{x}_N$.

12.3 (*) Рассмотрим входные векторы \mathbf{x}_n в простом преобразовании, заданном в (12.2), где весовые коэффициенты a_{nm} определяются в (12.5). Докажите, что если все входные векторы ортогональны, так что $\mathbf{x}_n^T \mathbf{x}_m = 0$

для $n \neq m$, то выходные векторы будут просто равны входным векторам, и тогда $\mathbf{y}_n = \mathbf{x}_n$ для $n = 1, \dots, N$.

- 12.4** (*) Рассмотрим два независимых произвольных вектора \mathbf{a} и \mathbf{b} , каждый из которых имеет размерность D и каждый берется из гауссова распределения с нулевым средним значением и единичной дисперсией $\mathcal{N}(\cdot | \mathbf{0}, \mathbf{I})$. Докажите, что ожидаемое значение $(\mathbf{a}^T \mathbf{b})^2$ определяется D .
- 12.5** (*** Докажите, что многоголовое внимание, определяемое в (12.19), можно переписать в виде

$$\mathbf{Y} = \sum_{h=1}^H \mathbf{H}_h \mathbf{X} \mathbf{W}^{(h)}, \quad (12.42)$$

где \mathbf{H}_h задается в (12.15), и также определено

$$\mathbf{W}^{(h)} = \mathbf{W}_h^{(v)} \mathbf{W}_h^{(o)}. \quad (12.43)$$

Здесь матрица $\mathbf{W}^{(o)}$ разбита по горизонтали на подматрицы, обозначаемые $\mathbf{W}_h^{(o)}$, каждая из которых имеет размерность $D_v \times D$, что соответствует вертикальным сегментам конкатенированной матрицы внимания (рис. 12.7). Поскольку D_v обычно меньше D , например $D_v = D/H$ является обычным выбором, эта объединенная матрица оказывается неполного ранга. Поэтому использование полностью гибкой матрицы для замены $\mathbf{W}_h^{(v)} \mathbf{W}_h^{(o)}$ не будет эквивалентно исходной формулировке в тексте.

- 12.6** (**) Выразите функцию самовнимания (12.14) как полносвязную сеть в виде матрицы, которая отображает полную входную последовательность конкатенированных векторов слов в выходной вектор той же размерности. Обратите внимание, что такая матрица будет иметь $\mathcal{O}(N^2 D^2)$ параметров. Докажите, что сеть самовнимания соответствует разреженной версии этой матрицы с разделением параметров. Нарисуйте эскиз структуры этой матрицы, отметив общие элементы, а также элементы, в которых все элементы равны нулю.
- 12.7** (*) Докажите, что если отбросить позиционное кодирование входных векторов, то выходы многоголового слоя внимания, определяемого в (12.19), эквивариантны по отношению к переупорядочиванию входной последовательности.
- 12.8** (*** Рассмотрим два D -мерных единичных вектора \mathbf{a} и \mathbf{b} , взятых из случайного распределения и удовлетворяющих $\|\mathbf{a}\| = 1$ и $\|\mathbf{b}\| = 1$. Предположим, что распределение симметрично относительно начала координат, т. е. зависит только от расстояния до начала координат, но не от направления. Докажите, что при больших значениях D величина косинуса угла между этими векторами близка к нулю и, следовательно, эти случайные векторы почти ортогональны в пространстве высокого размера. Для этого нужно определить ортонормальный базисный набор $\{\mathbf{u}_i\}$, где $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$, и выразить \mathbf{a} и \mathbf{b} в виде расширений по этому базису.

- 12.9** (**) Рассмотрим позиционное кодирование, где входной вектор лексем \mathbf{x} объединяется с вектором позиционного кодирования \mathbf{e} . Докажите, что если этот объединенный вектор претерпит общее линейное преобразование путем умножения с помощью матрицы, то результат можно выразить как сумму линейно преобразованного входного и линейно преобразованного позиционного векторов.
- 12.10** (**) Докажите, что позиционное кодирование, определяемое в (12.25), обладает тем свойством, что для фиксированного смещения k необходимо представить кодирование в позиции $n + k$ в виде линейной комбинации кодирования в позиции n с коэффициентами, зависящими только от k , но не от n . Для этого воспользуйтесь следующими тригонометрическими тождествами:

$$\cos(A + B) = \cos A \cos B - \sin A \sin B, \quad (12.44)$$

$$\sin(A + B) = \cos A \sin B + \sin A \cos B. \quad (12.45)$$

Докажите, что если кодирование основано только на синусоидальных функциях, без косинусоидальных, то это свойство больше не выполняется.

- 12.11** (*) Рассмотрим модель мешка слов (12.28), в которой каждое из компонентных распределений $p(\mathbf{x}_n)$ задается общей для всех слов таблицей вероятностей. Докажите, что решение с максимальным правдоподобием, заданное обучающим набором векторов, задается таблицей, записи которой представляют собой доли случаев, когда каждое слово встречается в обучающем наборе.
- 12.12** (*) Рассмотрим авторегрессионную языковую модель, заданную (12.31), и предположим, что члены $p(\mathbf{x}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$ в правой части представлены общими таблицами вероятностей. Докажите, что количество записей в этих таблицах растет экспоненциально с увеличением значения n .
- 12.13** (*) При использовании метода n -grams обычно обучают модели n -grams и $(n - 1)$ -grams одновременно, а затем вычисляют условную вероятность с помощью правила произведения вероятностей в форме

$$p(\mathbf{x}_n | \mathbf{x}_{n-L+1}, \dots, \mathbf{x}_{n-1}) = \frac{p_L(\mathbf{x}_{n-L+1}, \dots, \mathbf{x}_n)}{p_{L-1}(\mathbf{x}_{n-L+1}, \dots, \mathbf{x}_{n-1})}. \quad (12.46)$$

Объясните, почему это удобнее, чем хранить левую часть отдельно, и докажите, что для получения правильных вероятностей при вычислении $p_{L-1}(\dots)$ необходимо опустить последнюю лексему из каждой последовательности.

- 12.14** (**) Запишите псевдокод для процесса вывода в обученной RNN с архитектурой, изображенной на рис. 12.13.

- 12.15** (**) Рассмотрим последовательность из двух лексем y_1 и y_2 , каждая из которых может принимать состояния A или B . В таблице ниже приведено совместное распределение вероятностей $p(y_1, y_2)$:

		$y_1 = A$	$y_1 = B$
		0,0	0,4
$y_2 = A$	0,1		0,25
$y_2 = B$			

Наиболее вероятной последовательностью, как видно, является $y_1 = B$, $y_2 = B$, и вероятность этого равна 0,4. Используя правила суммы и произведения вероятностей, запишите значения маргинального распределения $p(y_1)$ и условного распределения $p(y_2 | y_1)$. Докажите, что если сначала максимизировать $p(y_1)$ для получения значения y_1^* , а затем максимизировать $p(y_2 | y_1^*)$, то получится последовательность, отличная от общей наиболее вероятной последовательности. Найдите вероятность этой последовательности.

- 12.16** (*) Модель BERT-Large (Devlin et al., 2018) характеризуется максимальной длиной входного сигнала в 512 лексем, каждая из которых имеет размерность $D = 1024$ и взята из словарного запаса размером 30 000. Она имеет 24 слоя трансформеров с 16 головами самовнушения с $D_q = D_k = D_v = 64$, а позиционно ориентированные сети MLP имеют два слоя с 4096 скрытыми узлами. Докажите, что общее количество параметров в языковой модели кодирующего трансформера BERT равно примерно 340 млн.

Глава 13

Графовые нейронные сети

В предыдущих главах этой книги уже рассматривались структурированные данные в виде последовательностей и изображений, соответствующих одномерным и двумерным массивам переменных. В более широком контексте можно говорить о большом множестве разновидностей структурированных данных, которые лучше всего описываются графом, как показано на рис. 13.1. В общем случае граф состоит из набора объектов, называемых узлами (*nodes*), соединенных ребрами (*edges*). Как узлы, так и ребра могут содержать ассоциированные с ними данные. Например, в молекуле узлы и ребра связаны с дискретными переменными, соответствующими типам атомов (углерод, азот, водород и т. д.) и типам атомарных связей (одинарная связь, двойная связь и т. д.). В случае сети железных дорог каждая железнодорожная линия может быть связана с непрерывной переменной, представляющей собой среднее время в пути между двумя городами. Здесь предполагается, что ребра симметричны – например, время в пути из Лондона в Кембридж такое же, как и время в пути из Кембриджа в Лондон. Такие ребра изображаются неориентированными связями между узлами. Для Всемирной паутины ребра являются направленными, поскольку если на странице A есть гиперссылка, указывающая на страницу B, то совсем не факт, что на странице B есть гиперссылка обратно на страницу A.

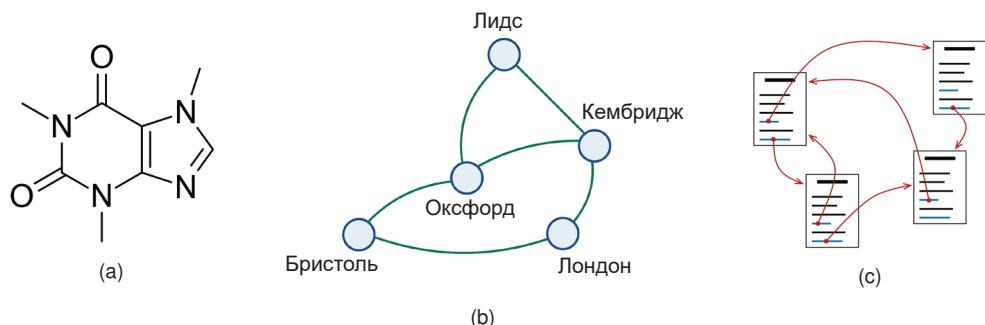


РИС. 13.1 Три примера графовых структурных данных: (a) молекула кофеина, состоящая из атомов с химическими связями, (b) железнодорожная сеть из городов, соединенных железнодорожными линиями, и (c) Всемирная паутина из страниц, соединенных гиперссылками

Среди других примеров данных, структурированных в виде графов, можно назвать сеть взаимодействия белков, в которой узлы – это белки, а ребра выражают степень взаимодействия пар белков, электрическую цепь, где узлы – это компоненты, а ребра – проводники, или социальную сеть, где узлы – это люди, а ребра представляют собой «дружеские отношения». Возможны и более сложные структуры графов, например граф знаний внутри компании включает в себя множество различных типов узлов, таких как люди, документы и встречи, а также множество типов ребер, отражающих различные свойства, например присутствие человека на встрече или ссылку из документа на другой документ.

В этой главе рассматривается вопрос применимости глубокого обучения к данным с графовой структурой. Пример структурированных данных уже встречался при обсуждении изображений, где отдельные элементы вектора данных изображения x соответствуют пикселям на регулярной сетке. Фактически изображение – это особый пример графовых структурированных данных, в которых узлами являются пиксели, а ребра определяют расположение соседних пикселей.

Сверточные нейронные сети (CNN, см. главу 10) учитывают эту структуру, опираясь на предварительные знания о взаимном расположении пикселей, а также на эквивариантность таких свойств, как сегментация, и инвариантность таких свойств, как классификация. На примере CNN для работы с изображениями нам предстоит изучить более общий принцип глубокого обучения для графических данных, известный как *графовые нейронные сети (graph neural networks)* (Zhou et al., 2018; Wu et al., 2019; Hamilton, 2020; Veličković, 2023). Далее будет показано, что ключевым моментом при применении глубокого обучения к графовым данным является обеспечение эквивариантности или инвариантности по отношению к переупорядочиванию узлов в графе.

13.1. Машинное обучение на графах

Существует множество видов прикладных задач, которые можно решить с помощью структурированных графов данных, и их можно сгруппировать в соответствии с целями прогнозирования свойств узлов, ребер или всего графа. Примером прогнозирования узлов может быть классификация документов по их темам на основе гиперссылок и ссылок между документами.

Что касается ребер, то, например, зная о некоторых видах взаимодействий в белковой сети, можно предсказать наличие каких-либо дополнительных связей. Такие виды работ называются задачами *предсказания ребер (edge prediction)* или *задачами заполнения графа (graph completion tasks)*. Существуют также задачи, в которых ребра известны заранее, а целью является обнаружение кластеров или «сообществ» в графе.

Наконец, можно прогнозировать свойства, относящиеся к графу в целом. Например, можно спрогнозировать, будет ли конкретная молекула раствори-

ма в воде. В этом случае вместо одного графа будет набор данных из различных графов, которые могут рассматриваться как полученные из некоторого общего распределения. Иными словами, подразумевается, что сами графы являются *независимыми и равномерно распределенными*. Такие задачи можно рассматривать в качестве задач регрессии или классификации графов.

В случае классификации растворимости молекул можно получить обучающий набор молекул с метками, а также тестовый набор новых молекул, растворимость которых необходимо предсказать. Это стандартный пример *индуктивной* задачи, которые неоднократно встречались в предыдущих главах. Однако некоторые примеры прогнозирования графов являются *трансдуктивными*, когда дана структура всего графа вместе с метками для некоторых узлов, а задача состоит в прогнозировании меток оставшихся узлов. Примером может служить большая социальная сеть, в которой необходимо классифицировать каждый узел как реального человека или автоматизированного бота. Здесь небольшое количество узлов может быть промаркировано вручную, но исследовать каждый узел по отдельности в большой и постоянно меняющейся социальной сети было бы непосильной задачей. Поэтому в процессе обучения можно получить доступ ко всему графу вместе с метками для подмножества узлов и с помощью этих меток попытаться спрогнозировать метки для остальных узлов. Это можно рассматривать как одну из разновидностей обучения с частичным наблюдением.

Помимо непосредственного решения задач прогнозирования, глубокое обучение на графах может также использоваться для выявления полезных внутренних представлений, которые впоследствии могут упростить решение ряда дальнейших задач. Это называют *обучением представлениям графов* (*graph representation learning*). Например, можно попытаться построить базовую модель для молекул, обучив систему глубокого обучения на большом корпусе молекулярных структур. Цель состоит в том, чтобы после обучения такую базовую модель можно было точно настроить под конкретные задачи с помощью небольшого набора маркированных данных.

Графовые нейронные сети определяют *вектор вложения* для каждого из узлов, обычно инициализируемый наблюдаемыми свойствами конкретного узла, которые затем преобразуются через ряд обучаемых слоев для создания обучаемого представления. Это аналогично тому, как встраивания слов, или лексем, обрабатываются с помощью ряда слоев в трансформере (см. главу 12) для получения представления, которое лучше отражает смысл слов в контексте остального содержания текста. Графовые нейронные сети также могут использовать обученные вложения, связанные с ребрами и графом в целом.

13.1.1. Свойства графов

В этой главе основное внимание будет уделено простым графикам, в которых между любой парой узлов существует не более одного ребра, при этом ребра являются неориентированными и не содержат самопересечений, соединя-

ющих узел с самим собой. Этого вполне достаточно для введения ключевых определений графовых нейронных сетей, и это также охватывает широкий спектр практических приложений. Затем все эти определения могут быть использованы для более сложных графовых структур.

Начнем с введения некоторых обозначений, связанных с графиками, и определения некоторых важных свойств. Граф $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ состоит из множества узлов (*nodes*) или вершин (*vertices*), обозначаемых \mathcal{V} , и множества ребер (*links*), обозначаемых \mathcal{E} . Узлы индексируются как $n = 1, \dots, N$, а ребро из узла n в узел m записывается как (n, m) . Если два узла связаны ребром, они называются соседними (*neighbours*), а множество всех соседних узлов обозначается $\mathcal{N}(n)$.

В дополнение к структуре графа обычно имеются наблюдаемые данные, связанные с узлами. Для каждого узла n соответствующие переменные узла можно представить в виде D -мерного вектора столбцов \mathbf{x}_n и сгруппировать их в матрицу данных \mathbf{X} размерности $N \times D$, где строка n задается \mathbf{x}_n^T . Также могут существовать переменные данных (см. раздел 13.3.2), связанные с ребрами графа, но для начала остановимся только на переменных узлов.

13.1.2. Матрица смежности

Удобным способом задания ребер в графе является использование *матрицы смежности* (*adjacency matrix*), обозначаемой \mathbf{A} . Для определения матрицы смежности сначала необходимо выбрать порядок следования узлов. Если в графе N узлов, их можно проиндексировать через $n = 1, \dots, N$. Матрица смежности имеет размерность $N \times N$ и содержит 1 в каждой позиции n, m , для которой существует ребро, идущее из узла n в узел m , а все остальные записи равны 0. Для графов с неориентированными ребрами матрица смежности будет симметричной, поскольку наличие ребра из вершины n в вершину m подразумевает наличие ребра из вершины m в вершину n , и поэтому $A_{mn} = A_{nm}$ для всех n и m . Пример матрицы смежности показан на рис. 13.2.

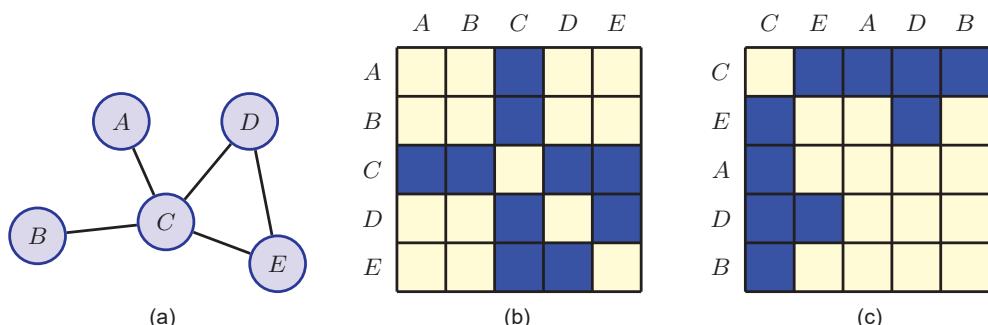


РИС. 13.2 Пример матрицы смежности: (a) пример графа с пятью узлами, (b) соответствующая матрица смежности для определенного выбора порядка узлов, и (c) матрица смежности, соответствующая другому выбору порядка узлов

Поскольку матрица смежности определяет структуру графа, ее можно использовать непосредственно в качестве входных данных для нейронной сети. Для этого ее необходимо «сплющить», например объединив столбцы в один длинный вектор-столбец. Однако основная проблема такого подхода заключается в зависимости матрицы смежности от произвольного выбора порядка следования узлов, как показано на рис. 13.2. Предположим, для примера, что требуется предсказать растворимость молекулы. Это определенно не должно зависеть от упорядочивания узлов при записи матрицы смежности. Поскольку число перестановок факториально увеличивается с ростом числа узлов, то попытки обучиться инвариантности перестановок с помощью больших наборов данных или путем их дополнения оказываются непрактичными. Вместо этого при построении архитектуры сети следует рассматривать это свойство инвариантности как индуктивное смещение.

13.1.3. Эквивариантность перестановок

Перестановку меток узлов можно выразить математически путем введения концепции *матрицы перестановок* (*permutation matrix*) \mathbf{P} , которая имеет тот же размер, что и матрица смежности, и задает определенную перестановку порядка следования узлов. Она содержит единственную 1 в каждой строке и единственную 1 в каждом столбце, а также 0 во всех остальных элементах, так что 1 в позиции n, m означает, что после перестановки узел n будет переименован в узел m . Рассмотрим, например, перестановку из $(A, B, C, D, E) \rightarrow (C, E, A, D, B)$, которая соответствует двум вариантам упорядочивания узлов на рис. 13.2.

Соответствующая матрица перестановок (см. упражнение 13.1) имеет вид:

$$\mathbf{P} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}. \quad (13.1)$$

Более формально матрицу перестановок можно определить следующим образом. Сначала введем стандартный единичный вектор \mathbf{u}_n для $n = 1, \dots, N$. Это вектор-столбец, в котором все элементы равны 0, кроме элемента n , который равен 1. В этой записи матрица тождества задается как

$$\mathbf{I} = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_N^T \end{pmatrix}. \quad (13.2)$$

Теперь можно ввести функцию перестановки $\pi(\cdot)$, которая переводит n в $m = \pi(n)$. Соответствующая матрица перестановок имеет вид:

$$\mathbf{P} = \begin{pmatrix} \mathbf{u}_{\pi(1)}^T \\ \mathbf{u}_{\pi(2)}^T \\ \vdots \\ \mathbf{u}_{\pi(N)}^T \end{pmatrix}. \quad (13.3)$$

Если переупорядочить метки на узлах графа, то в соответствующей матрице данных узлов \mathbf{X} произойдет перестановка строк в соответствии с $\pi(\cdot)$, которая может быть достигнута предварительным умножением на \mathbf{P} (см. упражнение 13.4), чтобы получить

$$\tilde{\mathbf{X}} = \mathbf{P}\mathbf{X}. \quad (13.4)$$

Для матрицы смежности как строки, так и столбцы становятся перестановочными. Опять же, строки можно переставлять с помощью предварительного умножения на \mathbf{P} , тогда как столбцы переставляются с помощью последующего умножения на \mathbf{P}^T (см. упражнение 13.5), что дает новую матрицу смежности:

$$\tilde{\mathbf{A}} = \mathbf{P}\mathbf{A}\mathbf{P}^T. \quad (13.5)$$

При применении глубокого обучения к данным с графовой структурой требуется представить структуру графа в числовой форме, чтобы ее можно было передать в нейронную сеть, а для этого необходимо присвоить узлам определенный порядок. Однако конкретное упорядочивание можно выбрать произвольно, поэтому важно убедиться, что любое глобальное свойство графа не зависит от этого упорядочивания. Другими словами, предсказания сети должны оставаться *инвариантными* к переупорядочиванию меток узлов, так что

$$y(\tilde{\mathbf{X}}, \tilde{\mathbf{A}}) = y(\mathbf{X}, \mathbf{A}), \quad (\text{Инвариантность}) \quad (13.6),$$

где $y(\cdot, \cdot)$ – это выход сети.

Также может возникнуть необходимость сделать прогнозы, относящиеся к отдельным узлам. В этом случае при изменении порядка обозначения узлов в соответствующих прогнозах должно наблюдаться такое же изменение порядка, чтобы конкретный прогноз всегда ассоциировался с одним и тем же узлом независимо от выбора порядка. Другими словами, прогнозы узлов должны быть *эквивариантны* по отношению к переупорядочиванию меток узлов. Это можно выразить как

$$y(\tilde{\mathbf{X}}, \tilde{\mathbf{A}}) = \mathbf{P}y(\mathbf{X}, \mathbf{A}), \quad (\text{Эквивариантность}) \quad (13.7),$$

где $y(\cdot, \cdot)$ – это вектор выходов сети с одним элементом на узел.

13.2. Нейронный обмен сообщениями

Обеспечение инвариантности или эквивариантности при перестановках меток узлов является одним из ключевых аспектов проектирования, когда глубокие нейронные сети работают с данными, структурированными графами. Другое соображение заключается в том, что необходимо использовать репрезентативные возможности глубоких нейронных сетей, поэтому концепция «слоя» сохраняется как вычислительное преобразование, которое может применяться многократно. Если каждый слой сети эквивариантен при переупорядочивании узлов, то несколько последовательно применяемых слоев также будут эквивариантны, позволяя каждому слою сети получать информацию от структуры графа.

Для сетей, выходы которых представляют собой предсказания на уровне узлов, вся сеть будет эквивариантной, как и требовалось. Если сеть используется для прогнозирования свойств на уровне графа, то можно включить последний слой, который будет инвариантен к перестановкам своих входов. Кроме того, необходимо, чтобы каждый слой представлял собой гибкую нелинейную функцию и был дифференцируемым относительно своих параметров, чтобы его можно было обучить методом стохастического градиентного спуска с использованием градиентов, полученных в результате автоматического дифференцирования.

Графы бывают разных размеров. Например, в разных молекулах может быть разное количество атомов, поэтому представление с фиксированной длиной, используемое в стандартных нейронных сетях, не подходит. Следовательно, сеть должна быть способна обрабатывать входные данные переменной длины, как это происходит с сетью трансформеров (см. главу 12). Некоторые графы могут быть очень большими, например для социальной сети с миллионами участников, и поэтому также необходимо строить модели, которые хорошо масштабируются. Неудивительно, что важную роль будет играть разделение параметров как для встраивания свойств инвариантности и эквивариантности в архитектуру сети, так и для облегчения масштабирования на большие графы.

13.2.1. Сверточные фильтры

Для разработки механизма, отвечающего всем этим требованиям, можно обратиться к примеру обработки изображений с помощью сверточных нейронных сетей (см. главу 10). Для начала следует отметить, что изображение можно рассматривать как конкретный пример графовой структуры данных, в которой узлами являются пиксели, а ребра представляют собой пары пикселей, смежных на изображении, при этом смежность включает узлы, смежные по диагонали, а также узлы, смежные по горизонтали или вертикали.

В сверточной сети осуществляются последовательные преобразования области изображения таким образом, что пиксель на определенном слое вычисляет функцию состояний пикселей (см. раздел 10.2) на предыдущем

слое с помощью локальной функции, называемой *фильтром* (*filter*). Рассмотрим сверточный слой с фильтрами 3×3 , как показано на рис. 13.3а. Здесь (а) фильтр, вычисляемый узлом i в слое $l + 1$ глубокой сверточной сети, является функцией значений активации в слое l над локальным участком пикселей. (б) Та же структура вычислений в виде графа, показывающего «сообщение», поступающие в узел i от его соседей. Вычисления, выполняемые одним фильтром для одного пикселя в слое $l + 1$, могут быть выражены как

$$z_i^{(l+1)} = f\left(\sum_j w_j z_j^{(l)} + b\right), \quad (13.8)$$

где $f(\cdot)$ – это дифференцируемая нелинейная функция активации, например ReLU, а сумма по j берется по всем девяти пикселям небольшого фрагмента в слое l . Одна и та же функция применяется к нескольким фрагментам изображения, так что веса w_j и смещение b являются общими для всех фрагментов (и поэтому не несут индекса i).

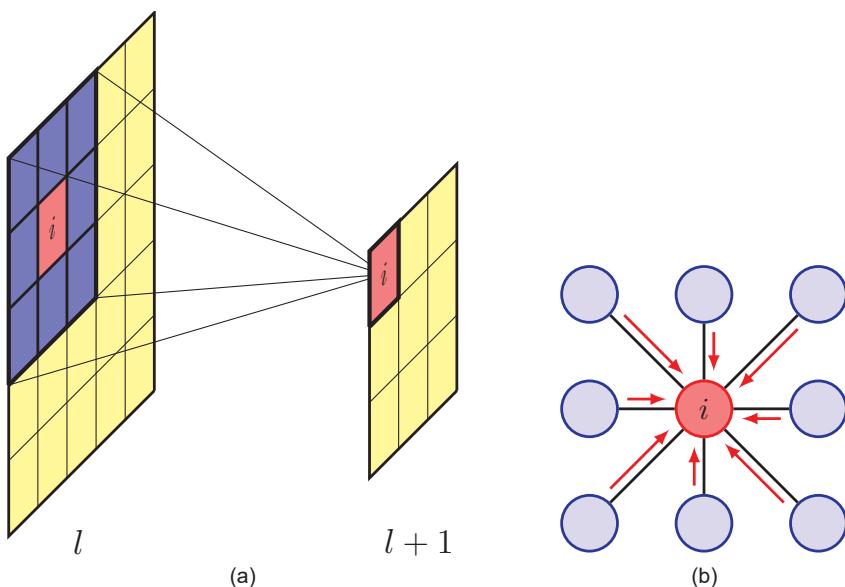


РИС. 13.3 Сверточный фильтр для изображений в виде графовой структуры вычислений

В существующем виде выражение (13.8) не является эквивариантным при перестановке узлов в слое l , поскольку весовой вектор с элементами w_j не является инвариантным при перестановке его элементов. Однако можно добиться эквивариантности с помощью следующих простых модификаций. Сначала рассмотрим фильтр в виде графа, как показано на рис. 13.3б, и выделим вклад узла i . Остальные восемь узлов являются его соседями $N(i)$. Затем предположим, что один весовой параметр w_{neigh} разделяется между соседями, так что

$$z_i^{(l+1)} = f \left(w_{\text{neigh}} \sum_{j \in \mathcal{N}(i)} z_j^{(l)} + w_{\text{self}} z_i^{(l)} + b \right), \quad (13.9),$$

где узел i имеет свой собственный весовой параметр w_{self} .

Можно интерпретировать (13.9) как обновление локального представления z_i в узле i путем сбора информации из соседних узлов за счет передачи сообщений (*messages*) из соседних узлов в узел i . В этом случае сообщения – это просто активации других узлов. Затем эти сообщения объединяются с информацией из узла i , и результат преобразуется с помощью нелинейной функции. Информация от соседних узлов агрегируется с помощью простого сложения в (13.9), и это однозначно инвариантно к любой перестановке меток, связанных с этими узлами. Более того, операция (13.9) применяется синхронно к каждому узлу графа, поэтому если переместить узлы, то результирующие вычисления не изменятся, но их упорядочивание также будет изменено, и, следовательно, эти вычисления эквивариантны при переупорядочивании узлов. Обратите внимание, что это зависит от того, что параметры w_{neigh} , w_{self} и b являются общими для всех узлов.

13.2.2. Графовые сверточные сети

Теперь воспользуемся примером свертки как шаблоном при построении глубоких нейронных сетей для данных с графовой структурой. Целью является определение гибкого нелинейного преобразования вложений узлов, дифференцируемого относительно набора весовых параметров и параметров смещения, которое отображает переменные в слое l на соответствующие переменные в слое $l + 1$. Для каждого узла n в графе и для каждого слоя l в сети введем D -мерный вектор-столбец $\mathbf{h}_n^{(l)}$ переменных вложений узлов, где $n = 1, \dots, N$ и $l = 1, \dots, L$.

Таким образом, преобразование, представленное в (13.9), сначала собирает и комбинирует информацию от соседних узлов, а затем обновляет ее в зависимости от существующего вложения узла и поступающих сообщений. Поэтому можно рассматривать каждый уровень обработки в виде двух последовательных этапов. Первый – это этап *агрегации*, на котором для каждого узла n сообщения передаются этому узлу от его соседей и объединяются для формирования нового вектора $\mathbf{z}_n^{(l)}$ способом, инвариантным к перестановкам. Затем следует этап *обновления*, на котором агрегированная информация от соседних узлов объединяется с локальной информацией от самого узла и используется для вычисления скорректированного вектора вложения для этого узла.

Рассмотрим конкретный узел n в графе. Сначала агрегируем векторы узлов от всех соседей узла n :

$$\mathbf{z}_n^{(l)} = \text{Aggregate} (\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}). \quad (13.10)$$

Форма этой агрегирующей функции очень гибкая, если она хорошо определена для переменного числа соседних узлов и не зависит от порядка этих

узлов. Потенциально она может содержать обучаемые параметры при условии, что она является дифференцируемой функцией по этим параметрам, что упрощает обучение методом градиентного спуска.

Затем используем другую операцию для обновления вектора вложения в узле n :

$$\mathbf{h}_n^{(l+1)} = \text{Update}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l)}). \quad (13.11)$$

Опять же, это может быть дифференцируемая функция по набору обучаемых параметров. Применение операции Aggregate с последующей операцией Update параллельно для каждого узла графа представляет один слой сети. Вложения узлов обычно инициализируются с помощью данных о наблюдаемых узлах, так что $\mathbf{h}_n^{(0)} = \mathbf{x}_n$. Обратите внимание, что каждый слой обычно имеет свои собственные независимые параметры, хотя они могут быть общими для всех слоев. Такая схема называется *нейронной сетью с передачей сообщений* (*message-passing neural network*) (Gilmer et al., 2017).¹⁷ Ее краткая структура представлена в алгоритме 13.1.

АЛГОРИТМ 13.1 Простая нейронная сеть с передачей сообщений

Input: неориентированный граф $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Начальные вложения узлов $\{\mathbf{h}_n^{(0)} = \mathbf{x}_n\}$

Функция $\text{Aggregate}(\cdot)$

Функция $\text{Update}(\cdot, \cdot)$

Output: итоговые вложения узлов $\{\mathbf{h}_n^{(L)}\}$

// Итеративная передача сообщений

for $l \in \{0, \dots, L - 1\}$ **do**

$\mathbf{z}_n^{(l)} \leftarrow \text{Aggregate}(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\})$

$\mathbf{h}_n^{(l+1)} \leftarrow \text{Update}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l)})$

end for

return $\{\mathbf{h}_n^{(L)}\}$

13.2.3. Операторы агрегации

Возможны различные формы агрегатной функции, но при этом она должна зависеть только от набора входов, а не от их порядка. Она также должна быть дифференцируемой функцией любых обучаемых параметров. Простейшей такой агрегирующей функцией, вытекающей из (13.9), является суммирование:

$$\text{Aggregate}(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}) = \sum_{m \in \mathcal{N}(n)} \mathbf{h}_m^{(l)}. \quad (13.12)$$

Простое суммирование явно не зависит от порядка соседних узлов и также является хорошо определенным независимо от того, сколько узлов находится в соседнем множестве. Обратите внимание, что у него нет обучаемых параметров.

Суммирование дает более сильное влияние на узлы с большим количеством соседей по сравнению с узлами с малым количеством соседей, и это может привести к количественным проблемам, особенно в таких приложениях, как социальные сети, где размер набора соседей может варьироваться в пределах нескольких порядков. Разновидностью этого подхода является определение операции агрегации как среднего значения соседних векторов вложения, так что

$$\text{Aggregate}(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}) = \frac{1}{|\mathcal{N}(n)|} \sum_{m \in \mathcal{N}(n)} \mathbf{h}_m^{(l)}, \quad (13.13)$$

где $|\mathcal{N}(n)|$ указывает на количество узлов в наборе соседей $\mathcal{N}(n)$. Однако эта нормализация также отбрасывает информацию о структуре сети и доказательно менее эффективна, чем простое суммирование (Hamilton, 2020), поэтому выбор в пользу ее использования зависит от относительной значимости характеристик узлов по сравнению со структурой графа.

Другая вариация этого подхода (Kipf and Welling, 2016) учитывает количество соседей для каждого из соседних узлов:

$$\text{Aggregate}(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}) = \sum_{m \in \mathcal{N}(n)} \frac{\mathbf{h}_m^{(l)}}{\sqrt{|\mathcal{N}(n)||\mathcal{N}(m)|}}. \quad (13.14)$$

Еще одним вариантом является взятие поэлементного максимума (или минимума) соседних векторов вложения, что также удовлетворяет требуемым свойствам – быть хорошо определенным для переменного числа соседей и не зависеть от их порядка.

Поскольку каждый узел в данном слое сети обновляется путем агрегирования информации от своих соседей в предыдущем слое, это задает *рецептивное поле*, аналогичное рецептивным полям фильтров CNN (см. главу 10). По мере обработки информации на последующих слоях обновления данного узла становятся зависимыми от постоянно увеличивающейся доли других узлов на предыдущих слоях, пока эффективное рецептивное поле потенциально не охватит весь график, как показано на рис. 13.4. Здесь в третьем слое один узел выделен красным цветом. Он получает информацию от двух своих соседей в предыдущем слое, а те, в свою очередь, получают информацию от своих соседей в первом слое. Как и в случае со сверточными нейронными сетями для изображений, можно заметить, что эффективное рецептивное поле, соответствующее количеству узлов, выделенных красным цветом, растет с увеличением числа слоев обработки. Однако большие разреженные графы могут потребовать слишком большого числа слоев, прежде чем каждый выход начнет зависеть от каждого входа. Поэтому некоторые архитектуры вводят дополнительный «суперузел», который соединяется непосредственно с каждым узлом исходного графа для быстрой передачи информации.

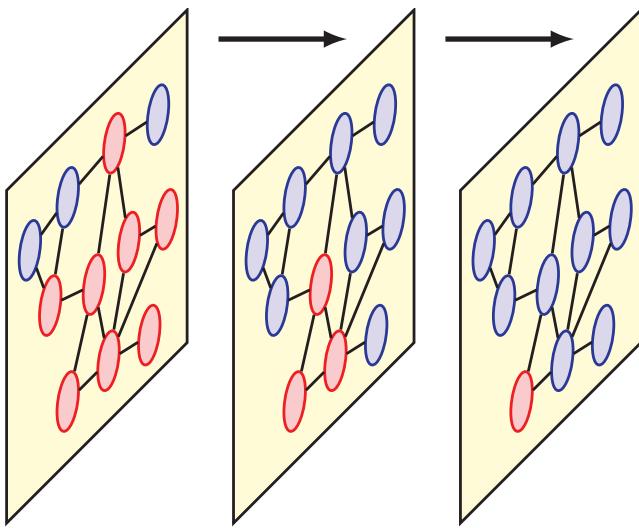


РИС. 13.4 Схематическое изображение потока информации через последовательные слои графовой нейронной сети

Рассмотренные ранее операторы агрегирования не имеют обучаемых параметров. Такие параметры можно задать, если сначала преобразовать каждый из векторов вложения из соседних узлов с помощью многослойной нейронной сети, обозначаемой MLP_φ , а затем объединить их выходы, где MLP обозначает «многослойный перцептрон» (multilayer perceptron), при этом φ – это параметры сети. Пока сеть имеет структуру и значения параметров, которые разделяются между узлами, этот оператор агрегирования остается инвариантным к перестановкам. Можно также преобразовать объединенный вектор с помощью другой нейронной сети MLP_θ с параметрами θ для получения общего оператора агрегирования:

$$\text{Aggregate}(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}) = \text{MLP}_\theta \left(\sum_{m \in \mathcal{N}(n)} \text{MLP}_\varphi(\mathbf{h}_m^{(l)}) \right), \quad (13.15)$$

где MLP_φ и MLP_θ являются общими для слоя l . Благодаря гибкости MLP преобразование, определяемое в (13.15), представляет собой универсальный аппроксиматор для любой инвариантной к перестановкам функции, которая отображает набор вложений в одно вложение (Zaheer et al., 2017). Обратите внимание, что суммирование может быть заменено другими инвариантными функциями, такими как средние значения, поэлементный максимум или минимум.

Особый случай графовых нейронных сетей возникает в случае, если в графе нет ребер, что соответствует простейшему неструктурированному набору узлов. В этом случае, если использовать (13.15) для каждого вектора $\mathbf{h}_n^{(l)}$ в наборе, где суммирование происходит по всем остальным векторам кроме

$\mathbf{h}_n^{(l)}$, то получится общая структура для обучения функций по неструктурированным наборам переменных, называемая *глубокими наборами* (*deep sets*).

13.2.4. Операторы обновления

Выбрав подходящий оператор агрегации (Aggregate), необходимо также определиться с формой оператора обновления (Update). По аналогии с (13.9) для CNN простая форма этого оператора будет выглядеть как

$$\text{Update}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l)}) = f(\mathbf{W}_{\text{self}} \mathbf{h}_n^{(l)} + \mathbf{W}_{\text{neigh}} \mathbf{z}_n^{(l)} + \mathbf{b}), \quad (13.16)$$

где $f(\cdot)$ – это нелинейная функция активации, такая как ReLU, применяемая поэлементно к своему векторному аргументу, и где \mathbf{W}_{self} , $\mathbf{W}_{\text{neigh}}$ и \mathbf{b} – это обучаемые веса и смещения, а $\mathbf{z}_n^{(l)}$ определяется оператором Aggregate в (13.10).

Если в качестве функции агрегирования использовать простое суммирование (13.12), а также одну и ту же матрицу весов между узлами и их соседями, так что $\mathbf{W}_{\text{self}} = \mathbf{W}_{\text{neigh}}$, то получится особенно простая форма оператора Update в виде

$$\mathbf{h}_n^{(l+1)} = \text{Update}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l)}) = f\left(\mathbf{W}_{\text{neigh}} \sum_{m \in \mathcal{N}(n), n} \mathbf{h}_m^{(l)} + \mathbf{b}\right). \quad (13.17)$$

Алгоритм передачи сообщений обычно инициализируется установкой $\mathbf{h}_n^{(0)} = \mathbf{x}_n$. Однако иногда возникает необходимость иметь внутренний вектор представления для каждого узла, который имеет большую или меньшую размерность, чем \mathbf{x}_n . Такое представление может быть инициализировано путем заполнения векторов узлов \mathbf{x}_n дополнительными нулями (для достижения большей размерности) или просто путем преобразования векторов узлов с помощью обучаемого линейного преобразования в пространство с желаемым числом измерений. Альтернативной формой инициализации, особенно когда нет переменных данных, связанных с узлами, является использование прямого унитарного (one-hot) вектора, обозначающего степень каждого узла (т. е. количество соседей).

В целом графовую нейронную сеть можно представить в виде последовательности слоев, которые будут последовательно преобразовывать вложения узлов. Если сгруппировать эти вложения в матрицу \mathbf{H} с n -й строкой – вектором \mathbf{h}_n^T , который инициализируется матрицей данных \mathbf{X} , то последовательные преобразования можно записать в виде

$$\begin{aligned} \mathbf{H}^{(1)} &= \mathbf{F}(\mathbf{X}, \mathbf{A}, \mathbf{W}^{(1)}) \\ \mathbf{H}^{(2)} &= \mathbf{F}(\mathbf{H}^{(1)}, \mathbf{A}, \mathbf{W}^{(2)}) \\ &\vdots \quad = \quad \vdots \\ \mathbf{H}^{(L)} &= \mathbf{F}(\mathbf{H}^{(L-1)}, \mathbf{A}, \mathbf{W}^{(L)}), \end{aligned} \quad (13.18)$$

где \mathbf{A} – это матрица смежности, а $\mathbf{W}^{(l)}$ представляет собой полный набор весов и смещений слоя l сети. При переупорядочении узлов, определяемом

матрицей перестановок \mathbf{P} , преобразование вложений узлов, вычисляемых слоем l , эквивариантно:

$$\mathbf{PH}^{(l)} = \mathbf{F}(\mathbf{PH}^{(l-1)}, \mathbf{PAP}^T, \mathbf{W}^{(l)}). \quad (13.19)$$

Как следствие, полная сеть вычисляет эквивариантное преобразование (см. упражнение 13.7).

13.2.5. Классификация узлов

Графовую нейронную сеть можно рассматривать как серию слоев, каждый из которых преобразует набор векторов узловых вложений $\{\mathbf{h}_n^{(l)}\}$ в новый набор $\{\mathbf{h}_n^{(l+1)}\}$ того же масштаба и размерности. После последнего сверточного слоя сети необходимо получить прогнозы для определения функции стоимости обучения, а также для прогнозирования новых данных с помощью обученной сети.

Рассмотрим для начала задачу классификации узлов в графе, которая является одним из наиболее распространенных применений графовых нейронных сетей. Можно определить выходной слой, иногда называемый *считывающим слоем* (*readout layer*). Он вычисляет функцию softmax для каждого узла, отвечающую классификации по C классам, в виде

$$y_{ni} = \frac{\exp(\mathbf{w}_i^T \mathbf{h}_n^{(L)})}{\sum_j \exp(\mathbf{w}_j^T \mathbf{h}_n^{(L)})}, \quad (13.20)$$

где $\{\mathbf{w}_i\}$ – это набор обучаемых весовых векторов, при этом $i = 1, \dots, N$. Затем можно определить функцию потерь как сумму потерь перекрестной энтропии по всем узлам и всем классам:

$$\mathcal{L} = - \sum_{n \in \mathcal{V}_{\text{train}}} \sum_{i=1}^C y_{ni}^{t_{ni}}, \quad (13.21)$$

где $\{t_{ni}\}$ – это целевые значения с однократным кодированием для каждого значения n . Поскольку весовые векторы $\{\mathbf{w}_i\}$ разделяются между выходными узлами, выходы y_{ni} эквивариантны к перестановке порядка узлов, и, следовательно, функция потерь (13.21) инвариантна. Если целью является прогнозирование непрерывных значений на выходах, то для определения подходящей функции потерь можно использовать простое линейное преобразование в сочетании с суммой квадратов ошибок.

Сумма по n в (13.21) берется по подмножеству узлов, обозначенном $\mathcal{V}_{\text{train}}$, и используется для обучения. Можно выделить следующие три типа узлов.

1. Узлы $\mathcal{V}_{\text{train}}$ маркируются и включаются в операции передачи сообщений графовой нейронной сети, а также используются для вычисления функции потерь, применяемой при обучении.
2. Потенциально существует также *трансдуктивное* подмножество узлов, обозначаемое $\mathcal{V}_{\text{trans}}$, которые не маркируются и не участвуют в оценке

функции потерь, используемой для обучения. Однако они все равно участвуют в операциях передачи сообщений во время обучения и вывода, и их метки могут быть предсказаны в рамках процесса вывода.

3. Оставшиеся узлы, обозначенные $\mathcal{V}_{\text{induct}}$, представляют собой набор индуктивных узлов, которые не используются для вычисления функции потерь, и ни эти узлы, ни связанные с ними ребра не участвуют в передаче сообщений на этапе обучения. Однако они участвуют в передаче сообщений на этапе вывода, и их метки предсказываются как результат вывода.

Если нет трансдуктивных узлов и, следовательно, тестовые узлы (и связанные с ними ребра) недоступны на этапе обучения, то такой процесс обычно называют *индуктивным обучением* (*inductive learning*), которое можно рассматривать как разновидность *контролируемого обучения* (*supervised learning*). Однако если имеются трансдуктивные узлы, то такой процесс называется *трансдуктивным обучением* (*transductive learning*), которое можно рассматривать как форму обучения с частичным контролем (*semi-supervised learning*).

13.2.6. Классификация ребер

В некоторых приложениях требуется составлять прогнозы относительно ребер графа, а не его узлов. Распространенной формой задачи классификации ребер является завершение ребер, в котором целью является определение наличия ребра между двумя вершинами. При заданном наборе вложений узлов для определения вероятности $p(n, m)$ наличия ребра между узлами n и m можно использовать точечное произведение между парами вложений посредством логистической сигмоидной функции:

$$p(n, m) = \sigma(\mathbf{h}_n^T \mathbf{h}_m). \quad (13.22)$$

В качестве примера можно привести прогноз того, имеют ли два человека в социальной сети общие интересы и поэтому могут ли они захотеть наладить общение.

13.2.7. Классификация графов

В некоторых приложениях графовых нейронных сетей целью является предсказание свойств новых графов по обучающему набору маркированных графов $\mathcal{G}_1, \dots, \mathcal{G}_N$. Для этого необходимо объединить векторы вложения конечного слоя так, чтобы они не зависели от произвольного упорядочивания узлов, и тем самым гарантировать, что выходные предсказания будут инвариантны к этому упорядочиванию. Цель примерно такая же, как у функции Aggregate, за исключением того, что учитываются все узлы графа, а не только наборы соседей отдельных узлов. Самый простой подход заключается в определении суммы векторов, включающих узлы:

$$\mathbf{y} = \mathbf{f}\left(\sum_{n \in \mathcal{V}} \mathbf{h}_n^{(L)}\right), \quad (13.23)$$

где функция \mathbf{f} может содержать обучаемые параметры, такие как линейное преобразование или нейронная сеть. Могут использоваться и другие инвариантные функции агрегирования, такие как средние значения или минимум или максимум по элементам.

Для решения задач классификации, таких как определение степени токсичности или безопасности молекулы-кандидата в лекарственные препараты, обычно используется потеря перекрестной энтропии, а для решения задач регрессии, таких как предсказание растворимости молекулы-кандидата в лекарственные препараты, – потеря квадратичной ошибки. Предсказания на уровне графов соответствуют индуктивной задаче, поскольку для обучения и для вывода необходимо иметь отдельные наборы графов.

13.3. Общие графовые сети

Существует множество вариаций и расширений для рассмотренных ранее графовых сетей. В этом разделе приведены некоторые ключевые концепции, а также ряд практических аспектов.

13.3.1. Графовые сети с вниманием

Механизм внимания обладает огромными возможностями при его использовании в качестве основы архитектуры трансформера (см. раздел 12.1). Он может быть использован в контексте графовых нейронных сетей для построения агрегатной функции, которая объединяет сообщения от соседних узлов. Входящие сообщения взвешиваются коэффициентами внимания A_{nm} , что дает

$$\mathbf{z}_n^{(l)} = \text{Aggregate}(\{\mathbf{h}_m^{(l)} : m \in \mathcal{N}(n)\}) = \sum_{m \in \mathcal{N}(n)} A_{nm} \mathbf{h}_m^{(l)}, \quad (13.24)$$

где коэффициенты внимания соответствуют

$$A_{nm} > 0, \quad (13.25)$$

$$\sum_{m \in \mathcal{N}(n)} A_{nm} = 1. \quad (13.26)$$

Такая структура известна как *графовая сеть с вниманием* (*graph attention network*) (Veličković et al, 2017). Она может учитывать индуктивное смещение, в соответствии с которым некоторые соседние узлы будут более важны, чем другие, при определении наилучшего обновления в зависимости от самих данных.

Существует множество способов определения коэффициентов внимания, и обычно в них используется функция softmax. Например, можно воспользоваться билинейной формой:

$$A_{nm} = \frac{\exp(\mathbf{h}_n^T \mathbf{W} \mathbf{h}_m)}{\sum_{m' \in \mathcal{N}(n)} \exp(\mathbf{h}_n^T \mathbf{W} \mathbf{h}_{m'})}, \quad (13.27)$$

где \mathbf{W} – это матрица $D \times D$ обучаемых параметров. Более общим вариантом является использование нейронной сети для объединения векторов вложения из узлов на каждом конце ребра:

$$A_{nm} = \frac{\exp\{\text{MLP}(\mathbf{h}_n, \mathbf{h}_m)\}}{\sum_{m' \in \mathcal{N}(n)} \exp\{\text{MLP}(\mathbf{h}_n, \mathbf{h}_{m'})\}}, \quad (13.28)$$

где MLP имеет единственную непрерывную выходную переменную, значение которой неизменно, если входные векторы меняются местами. При условии, что MLP является общей для всех узлов сети (см. упражнение 13.8), эта агрегирующая функция будет эквивариантной при перестановке узлов.

Графовую сеть с вниманием можно расширить путем введения нескольких голов внимания, где определены H различных наборов весов внимания $A_{nm}^{(h)}$ для $h = 1, \dots, H$ (см. раздел 12.1.6), при этом каждая голова оценивается с помощью одного из описанных выше механизмов и со своими независимыми параметрами. Затем они объединяются на этапе агрегирования с помощью конкатенации и линейной проекции. Обратите внимание, что для полно связной сети многоголовая графовая сеть с вниманием превращается в стандартный кодировщик трансформера (см. упражнение 13.9).

13.3.2. Встраивание ребер

В рассмотренных выше графовых нейронных сетях применяются векторы вложения, связанные с узлами. В некоторых сетях данные также связаны с ребрами. Даже если с ребрами не связаны наблюдаемые значения, все равно можно поддерживать и обновлять скрытые переменные по ребрам, и они могут вносить вклад во внутренние представления, которым обучается графовая нейронная сеть.

В дополнение к вложениям узлов, задаваемым $\mathbf{h}_n^{(l)}$, введем вложения ребер $\mathbf{e}_{nm}^{(l)}$. Тогда можно определить общие уравнения передачи сообщений в виде

$$\mathbf{e}_{nm}^{(l+1)} = \text{Update}_{\text{edge}}(\mathbf{e}_{nm}^{(l)}, \mathbf{h}_n^{(l)}, \mathbf{h}_m^{(l)}), \quad (13.29)$$

$$\mathbf{z}_n^{(l+1)} = \text{Aggregate}_{\text{node}}(\{\mathbf{e}_{nm}^{(l+1)} : m \in \mathcal{N}(n)\}), \quad (13.30)$$

$$\mathbf{h}_n^{(l+1)} = \text{Update}_{\text{node}}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l+1)}). \quad (13.31)$$

Обученные вложения ребер $\mathbf{e}_{nm}^{(L)}$ из последнего слоя могут быть использованы непосредственно для составления прогнозов, связанных с ребрами.

13.3.3. Вложения графов

В дополнение к вложениям узлов и ребер можно также использовать и обновлять вектор вложений $\mathbf{g}^{(l)}$, относящийся к графу в целом. Объединение всех этих аспектов позволяет определить более общий набор функций передачи сообщений и более богатый набор обучаемых представлений для задач с графовой структурой. В частности, возможно определение общих уравнений передачи сообщений (Battaglia et al., 2018):

$$\mathbf{e}_{nm}^{(l+1)} = \text{Update}_{\text{edge}}(\mathbf{e}_{nm}^{(l)}, \mathbf{h}_n^{(l)}, \mathbf{h}_m^{(l)}, \mathbf{g}^{(l)}), \quad (13.32)$$

$$\mathbf{z}_n^{(l+1)} = \text{Aggregate}_{\text{node}}(\{\mathbf{e}_{nm}^{(l+1)} : m \in \mathcal{N}(n)\}), \quad (13.33)$$

$$\mathbf{h}_n^{(l+1)} = \text{Update}_{\text{node}}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l+1)}, \mathbf{g}^{(l)}), \quad (13.34)$$

$$\mathbf{g}^{(l+1)} = \text{Update}_{\text{graph}}(\mathbf{g}^{(l)}, \{\mathbf{h}_n^{(l+1)} : n \in \mathcal{V}\}, \{\mathbf{e}_{nm}^{(l+1)} : (n, m) \in \mathcal{E}\}). \quad (13.35)$$

Эти уравнения обновления стартуют в (13.32) с обновления векторов вложения ребер $\mathbf{e}_{nm}^{(l+1)}$ на основе предыдущих состояний этих векторов, вложений узлов для узлов, соединенных каждым ребром, и вектора вложения уровня графа $\mathbf{g}^{(l)}$. Эти обновленные вложения ребер затем агрегируются по каждому ребру, соединенному с каждым узлом, с помощью (13.33), что дает набор агрегированных векторов. Они, в свою очередь, участвуют в обновлении вектора вложения узла $\{\mathbf{h}_n^{(l+1)}\}$ по актуальным векторам вложения узла и вектору вложения уровня графа с использованием (13.34).

Наконец, вектор вложения уровня графа обновляется с помощью (13.35) на основе информации от всех узлов и всех ребер графа вместе с вложением уровня графа из предыдущего слоя. Эти обновления с передачей сообщений показаны на рис. 13.5 и обобщены в алгоритме 13.2.

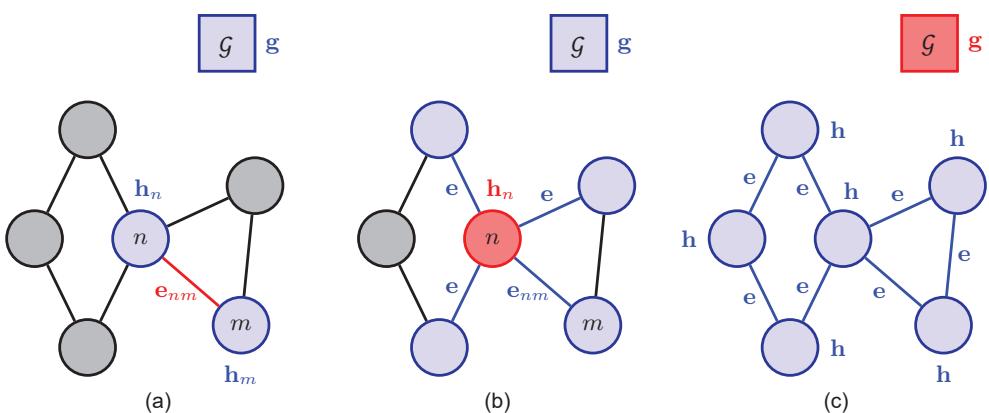


РИС. 13.5 Иллюстрация общих обновлений с передачей сообщений по графу, определенных в (13.32)–(13.35). (a) Обновления ребер, (b) обновления узлов и (c) обновления глобального графа. В каждом случае обновляемая переменная показана красным цветом, а переменные, которые вносят вклад в это обновление, показаны красным и синим цветом

АЛГОРИТМ 13.2 Графовая нейронная сеть с вложениями узлов, ребер и графов

Input: неориентированный граф $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Начальные вложения узлов $\{\mathbf{h}_n^{(0)}\}$

Начальные вложения ребер $\{\mathbf{e}_{nm}^{(0)}\}$

Начальные вложения графа $\mathbf{g}^{(0)}$

Output: итоговые вложения узлов $\{\mathbf{h}_n^{(L)}\}$

Итоговые вложения ребер $\{\mathbf{e}_{nm}^{(L)}\}$

Итоговые вложения графа $\mathbf{g}^{(L)}$

// Итеративная передача сообщений

for $l \in \{0, \dots, L - 1\}$ do

| $\mathbf{e}_{nm}^{(l+1)} \leftarrow \text{Update}_{\text{edge}}(\mathbf{e}_{nm}^{(l)}, \mathbf{h}_n^{(l)}, \mathbf{h}_m^{(l)}, \mathbf{g}^{(l)})$

$\mathbf{z}_n^{(l+1)} \leftarrow \text{Aggregate}_{\text{node}}(\{\mathbf{e}_{nm}^{(l+1)} : m \in \mathcal{N}(n)\})$

$\mathbf{h}_n^{(l+1)} \leftarrow \text{Update}_{\text{node}}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l+1)}, \mathbf{g}^{(l)})$

$\mathbf{g}^{(l+1)} \leftarrow \text{Update}_{\text{graph}}(\mathbf{g}^{(l)}, \{\mathbf{h}_n^{(l+1)}\}, \{\mathbf{e}_{nm}^{(l+1)}\})$

end for

return $\{\mathbf{h}_n^{(L)}\}, \{\mathbf{e}_{nm}^{(L)}\}, \mathbf{g}^{(L)}$

13.3.4. Чрезмерное сглаживание

Одна из существенных проблем, которая может возникнуть при использовании некоторых графовых нейронных сетей, называется *чрезмерным сглаживанием* (*over-smoothing*), когда векторы вложения узлов становятся очень похожими друг на друга после нескольких итераций передачи сообщений, что существенно ограничивает глубину сети. Один из способов ослабить эту проблему состоит в добавлении остаточных связей (см. раздел 9.5). Например, можно модифицировать оператор обновления в (13.34):

$$\mathbf{h}_n^{(l+1)} = \text{Update}_{\text{node}}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l+1)}, \mathbf{g}^{(l)} + \mathbf{h}_n^{(l)}). \quad (13.36)$$

Другой подход к смягчению эффекта избыточного сглаживания заключается в том, чтобы позволить выходному слою получать информацию от всех предыдущих слоев сети, а не только от последнего сверточного слоя. Это можно сделать, например, путем объединения представлений предыдущих слоев:

$$\mathbf{y}_n = \mathbf{f}(\mathbf{h}_n^{(1)} \oplus \mathbf{h}_n^{(2)} \oplus \dots \oplus \mathbf{h}_n^{(L)}), \quad (13.37)$$

где $\mathbf{a} \oplus \mathbf{b}$ обозначает объединение векторов \mathbf{a} и \mathbf{b} . В качестве варианта можно объединить векторы с помощью максимального пулинга вместо конкатенации. В этом случае каждый элемент выходного вектора задается максимальным значением всех соответствующих элементов векторов вложения из предыдущих слоев.

13.3.5. Регуляризация

Стандартные методы регуляризации (см. главу 9) применимы к графовым нейронным сетям, в том числе для добавления штрафных членов, таких как сумма квадратов значений параметров, к функции потерь. Кроме того, некоторые методы регуляризации были разработаны специально для графовых нейронных сетей.

Графовые нейронные сети уже используют совместное использование весов для достижения перестановочной эквивариантности и инвариантности, но, как правило, они имеют независимые параметры в каждом слое. Однако веса и смещения также могут быть общими для всех слоев, чтобы уменьшить количество независимых параметров.

В контексте графовых нейронных сетей отсев (dropout) подразумевает исключение случайных подмножеств узлов графа во время обучения, при этом для каждого прохода вперед выбирается новое случайное подмножество. Аналогичным образом это можно применить и к ребрам графа, когда в процессе обучения удаляются или маскируются случайно выбранные подмножества записей в матрице смежности.

13.3.6. Геометрическое глубокое обучение

Ранее было показано, что симметрия перестановок является ключевым аспектом при разработке моделей глубокого обучения для данных с графовой структурой. Она действует как форма индуктивного смещения, значительно снижая требования к данным и улучшая эффективность прогнозирования. В задачах, где графовые нейронные сети связаны с пространственными свойствами, такими как графические сетки, моделирование потоков жидкостей или молекулярных структур, появляются дополнительные свойства эквивариантности и инвариантности, которые могут быть встроены в архитектуру сети.

Рассмотрим задачу прогнозирования свойств молекулы, например, при исследовании пространства лекарств-кандидатов. Молекула может быть представлена как список атомов заданных типов (углерод, водород, азот и т. д.) вместе с пространственными координатами каждого атома, выраженным в виде трехмерного вектора-столбца. Для каждого атома n в каждом слое l можно ввести соответствующий вектор вложения, обозначаемый $\mathbf{r}_n^{(l)}$, и эти векторы могут быть инициализированы известными координатами атомов. Однако значения элементов этих векторов зависят от произвольного выбора системы координат, в то время как свойства молекулы не зависят. Например, растворимость молекулы не изменится, если ее повернуть в пространстве или перевести в новое положение относительно начала системы координат, или если сама система координат будет отражена для получения зеркальной версии молекулы. Поэтому свойства молекулы должны быть инвариантны при таких преобразованиях.

В результате тщательного выбора функциональных форм для операций обновления и агрегирования (Satorras, Hoogeboom and Welling, 2021) для до-

стижения требуемых свойств симметрии можно включить новые вложения $\mathbf{r}_n^{(l)}$ в уравнения обновления графовой нейронной сети (13.29)–(13.31):

$$\mathbf{e}_{nm}^{(l+1)} = \text{Update}_{\text{edge}}(\mathbf{e}_{nm}^{(l)}, \mathbf{h}_n^{(l)}, \mathbf{h}_m^{(l)}, \|\mathbf{r}_n^{(l)} - \mathbf{r}_m^{(l)}\|^2), \quad (13.38)$$

$$\mathbf{r}_n^{(l+1)} = \mathbf{r}_n^{(l)} + C \sum_{(n,m) \in \mathcal{E}} (\mathbf{r}_n^{(l)} - \mathbf{r}_m^{(l)}) \varphi(\mathbf{e}_{nm}^{(l+1)}), \quad (13.39)$$

$$\mathbf{z}_n^{(l+1)} = \text{Aggregate}_{\text{node}}(\{\mathbf{e}_{nm}^{(l+1)} : m \in \mathcal{N}(n)\}), \quad (13.40)$$

$$\mathbf{h}_n^{(l+1)} = \text{Update}_{\text{node}}(\mathbf{h}_n^{(l)}, \mathbf{z}_n^{(l+1)}). \quad (13.41)$$

Обратите внимание, что величина $\|\mathbf{r}_n^{(l)} - \mathbf{r}_m^{(l)}\|^2$ представляет собой квадратичное расстояние между координатами $\mathbf{r}_n^{(l)}$ и $\mathbf{r}_m^{(l)}$ и не зависит от преобразований, поворотов или отражений. Кроме того, координаты $\mathbf{r}_n^{(l)}$ обновляются через линейную комбинацию относительных разностей $\mathbf{r}_n^{(l)} - \mathbf{r}_m^{(l)}$. Здесь $\varphi(\mathbf{e}_{nm}^{(l+1)})$ – это общая скалярная функция вложений ребер, представленная нейронной сетью, а коэффициент C обычно равен обратной величине числа членов в сумме. Отсюда следует, что при таких преобразованиях сообщения в (13.38), (13.40) и (13.41) инвариантны (упражнение 13.10), а координатные вложения, задаваемые (13.39), эквивариантны.

Можно привести множество примеров симметрий в структурированных данных: от преобразований объектов в изображениях и перестановки порядков узлов в графах до вращений и преобразований молекул в трехмерном пространстве. Отслеживание этих симметрий в структуре глубокой нейронной сети является мощной формой индуктивного смещения и составляет основу для обширной области исследований, известной как *геометрическое глубокое обучение* (*geometric deep learning*) (Bronstein et al., 2017; Bronstein et al., 2021).

Упражнения

- 13.1** (*) Докажите, что перестановка $(A, B, C, D, E) \rightarrow (C, E, A, D, B)$, соответствующая двум вариантам упорядочения узлов на рис. 13.2, может быть выражена в форме (13.5) с помощью матрицы перестановок, заданной в (13.1).
- 13.2** (**) Докажите, что количество ребер, соединенных с каждым узлом графа, задается соответствующим диагональным элементом матрицы \mathbf{A}^2 , где \mathbf{A} – это матрица смежности.
- 13.3** (*) Нарисуйте граф, матрица смежности которого задана как

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}. \quad (13.42)$$

- 13.4** (**) Докажите, что в результате предварительного умножения матрицы данных \mathbf{X} с помощью матрицы перестановок \mathbf{P} , определенной в (13.3), образуется новая матрица данных $\tilde{\mathbf{X}}$, заданная в (13.4), строки которой переставляются в соответствии с функцией перестановки $\pi(\cdot)$.
- 13.5** (**) Докажите, что преобразованная матрица смежности $\tilde{\mathbf{A}}$, определенная в (13.5), где \mathbf{P} определена в (13.3), такова, что и строки, и столбцы в ней переставлены в соответствии с перестановочной функцией $\pi(\cdot)$ относительно исходной матрицы смежности \mathbf{A} .
- 13.6** (**) В этом упражнении необходимо записать уравнения обновления (13.16) в виде уравнений графа с помощью матриц. Чтобы не усложнять обозначения, опускаем индекс слоя l . Сначала соберем векторы вложения узлов $\{\mathbf{h}_n\}$ в матрицу \mathbf{H} размером $N \times D$, в которой строка n задается \mathbf{h}_n^T . Затем докажите, что агрегированные по соседству векторы \mathbf{z}_n , заданные

$$\mathbf{z}_n = \sum_{m \in N(n)} \mathbf{h}_m, \quad (13.43)$$

могут быть записаны в матричной форме как $\mathbf{Z} = \mathbf{AH}$, где \mathbf{Z} – это матрица $N \times D$, в которой строка n задана \mathbf{z}_n^T , а \mathbf{A} – это матрица смежности. Наконец, докажите, что аргумент нелинейной функции активации в (13.16) может быть записан в матричной форме как

$$\mathbf{AHW}_{\text{neigh}} + \mathbf{HW}_{\text{self}} + \mathbf{1}_D \mathbf{b}^T, \quad (13.44)$$

где $\mathbf{1}_D$ – это вектор столбцов в D -мерном пространстве, где все элементы равны 1.

- 13.7** (**) Используя свойство эквивариантности (13.19) для слоя l глубокой графовой сверточной сети вместе со свойством перестановки (13.4) для переменных узлов, докажите, что полная глубокая графовая сверточная сеть, определяемая в (13.18), также эквивариантна.
- 13.8** (**) Объясните, почему агрегатная функция, определенная в (13.24), где веса внимания заданы в (13.28), эквивариантна при перестановке узлов в графе.
- 13.9** (*) Докажите, что сеть внимания графа, где граф является полносвязным, так что между каждой парой узлов есть ребро, эквивалентна стандартной архитектуре трансформера.
- 13.10** (**) При преобразовании системы координат местоположение объекта, определяемое этой системой координат, преобразуется с помощью функции

$$\tilde{\mathbf{r}} = \mathbf{r} + \mathbf{c}, \quad (13.45)$$

где \mathbf{c} – это фиксированный вектор, описывающий преобразование. Аналогичным образом, если система координат повернута и/или зеркально отражена, вектор местоположения объекта преобразуется с помощью

$$\tilde{\mathbf{r}} = \mathbf{R}\mathbf{r}, \quad (13.46)$$

где \mathbf{R} – это *ортогональная матрица*, инверсия которой задается ее транспонированием, так что

$$\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}. \quad (13.47)$$

Используя эти свойства, докажите, что при преобразованиях, поворотах и отражениях сообщения в (13.38), (13.40) и (13.41) инвариантны и что вложения координат, заданные (13.39), эквивариантны.

Глава 14

Выборка

В глубоком обучении встречается множество ситуаций, когда необходимо создать синтетические выборки по переменной \mathbf{z} на основе распределения вероятностей $p(\mathbf{z})$. Здесь \mathbf{z} может быть скаляром, а распределение – одномерным гауссовым, или \mathbf{z} может быть изображением высокого разрешения, а $p(\mathbf{z})$ – генеративной моделью, определенной глубокой нейронной сетью. Такой процесс называется *выборкой* (*sampling*), также известной как *выборка Монте-Карло* (*Monte Carlo sampling*). Для многих простых распределений разработаны численные методы, позволяющие генерировать подходящие выборки напрямую, тогда как для более сложных распределений, в том числе определяемых неявным образом, могут понадобиться более сложные методы. В отличие от классической статистики, где под «выборкой» понимается набор значений, в этой главе применяется правило, согласно которому каждое конкретное значение называется выборкой.

В этой главе основное внимание уделяется аспектам составления выборок, которые наиболее актуальны для глубокого обучения. Дополнительную информацию о методах Монте-Карло в целом можно найти в работах (Gilks, Richardson and Spiegelhalter, 1996) и (Robert and Casella, 1999).

14.1. Основные алгоритмы выборки

В этом разделе рассмотрены различные относительно простые стратегии генерации случайных выборок из заданного распределения. Поскольку выборки будут формироваться компьютерным алгоритмом, они фактически будут *псевдослучайными*, т. е. будут рассчитаны с помощью детерминированного алгоритма, но тем не менее обязательно пройдут соответствующие тесты на определение случайности. Здесь подразумевается, что предоставлен алгоритм, который генерирует псевдослучайные числа, равномерно распределенные по $(0, 1)$, и действительно в большинство программных сред такая возможность уже встроена.

14.1.1. Ожидаемые значения

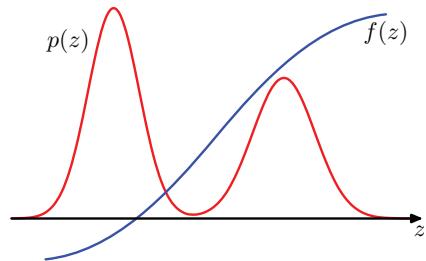
Несмотря на то что в некоторых случаях непосредственный интерес представляют сами выборки, в других ситуациях целью является оценка ожиданий

ний (*expectations*) относительно распределения. Предположим, необходимо найти ожидание функции $f(\mathbf{z})$ относительно распределения вероятностей $p(\mathbf{z})$. Здесь компоненты \mathbf{z} могут представлять собой дискретные переменные, непрерывные переменные или их сочетание. Для непрерывных переменных математическое ожидание определяется как

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z}, \quad (14.1)$$

где интеграл заменяется суммированием в случае с дискретными переменными. Схематически это показано на рис. 14.1 для одной непрерывной переменной. Будем считать, что такие ожидания слишком сложны, чтобы их можно было точно оценить аналитическими методами.

РИС. 14.1 Схематическое изображение функции $f(z)$, чье ожидание необходимо оценить относительно распределения $p(z)$



Основная суть методов выборки заключается в получении набора выборок $\mathbf{z}^{(l)}$ (где $l = 1, \dots, L$), взятых независимо из распределения $p(\mathbf{z})$. Это позволяет аппроксимировать ожидание (14.1) в виде конечной суммы:

$$\bar{f} = \frac{1}{L} \sum_{l=1}^L f(\mathbf{z}^{(l)}). \quad (14.2)$$

Если выборки $\mathbf{z}^{(l)}$ взяты из распределения $p(\mathbf{z})$, то $\mathbb{E}[\bar{f}] = \mathbb{E}[f(\mathbf{z})]$, а значит оценка \bar{f} имеет корректное среднее значение (см. упражнение 14.1). Это также можно записать в виде

$$\mathbb{E}[f(\mathbf{z})] \simeq \frac{1}{L} \sum_{l=1}^L f(\mathbf{z}^{(l)}), \quad (14.3)$$

где символ \simeq означает, что правая часть является несмещенной оценкой левой части, т. е. обе части равны при усреднении по распределению шума.

Дисперсия (см. упражнение 14.2) оценки (14.2) задается как

$$\text{var}[\bar{f}] = \frac{1}{L} \mathbb{E}[(f - \mathbb{E}[f])^2], \quad (14.4)$$

что представляет собой дисперсию функции $f(\mathbf{z})$ при распределении $p(\mathbf{z})$. Обратите внимание, что линейное уменьшение этой дисперсии с ростом L не зависит от размерности \mathbf{z} , и, в принципе, высокая точность может быть

достигнута при относительно небольшом числе выборок $\{\mathbf{z}^{(l)}\}$. Но проблема заключается в том, что выборки $\{\mathbf{z}^{(l)}\}$ могут быть не независимыми, и поэтому эффективный размер выборки может быть намного меньше, чем видимый размер выборки. Также, возвращаясь к рис. 14.1, следует отметить, что если $f(\mathbf{z})$ мала в областях, где $p(\mathbf{z})$ велико, и наоборот, то в ожидании могут преобладать области с малой вероятностью, а это означает, что для достижения достаточной точности потребуется относительно большой объем выборки.

14.1.2. Стандартные распределения

Теперь рассмотрим вопрос о получении случайных чисел из простых неравномерных распределений, исходя из предположения, что для этого уже имеется источник равномерно распределенных случайных чисел. Предположим, что z равномерно распределено на интервале $(0, 1)$ и что значения z преобразуются с помощью некоторой функции $g(\cdot)$ так, что $y = g(z)$. Распределение y (см. раздел 2.4) будет определяться

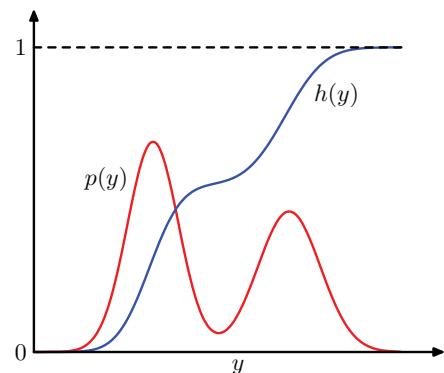
$$p(y) = p(z) \left| \frac{dz}{dy} \right|, \quad (14.5)$$

где в данном случае $p(z) = 1$. Целью является выбор функции $g(z)$ так, чтобы результирующие значения y имели некоторое конкретное требуемое распределение $p(y)$. При интегрировании (14.5) получается:

$$z = \int_{-\infty}^y p(\hat{y}) d\hat{y} \equiv h(y) dy, \quad (14.6)$$

что является неопределенным интегралом по $p(y)$. Таким образом, $y = h^{-1}(z)$ (см. упражнение 14.3), и поэтому необходимо преобразовать равномерно распределенные случайные числа с помощью функции, обратной неопределенному интегралу от искомого распределения. Это показано на рис. 14.2. Здесь $h(y)$ – это неограниченный интеграл от искомого распределения $p(y)$. Если равномерно распределенная случайная величина z преобразуется с помощью $y = h^{-1}(z)$, то y будет распределена в соответствии с $p(y)$.

РИС. 14.2 Геометрическая интерпретация метода преобразования для генерации неравномерно распределенных случайных чисел



Рассмотрим, например, экспоненциальное распределение

$$p(y) = \lambda \exp(-\lambda y), \quad (14.7)$$

где $0 \leq y < \infty$. В этом случае нижний предел интеграла в (14.6) равен 0, и поэтому $h(y) = 1 - \exp(-\lambda y)$. Получается, если преобразовать равномерно распределенную переменную z с помощью $y = -\lambda^{-1} \ln(1 - z)$, то y будет иметь экспоненциальное распределение.

Другим примером распределения, к которому можно применить метод преобразования, является *распределение Коши (Cauchy distribution)*:

$$p(y) = \frac{1}{\pi} \frac{1}{1 + y^2}. \quad (14.8)$$

В этом случае обратная часть неопределенного интеграла может быть выражена через функцию \tan (см. упражнение 14.4).

Обобщение на несколько переменных предполагает использование якобиана (определителя Якоби) при замене переменных (см. раздел 2.4), так что

$$p(y_1, \dots, y_M) = p(z_1, \dots, z_M) \left| \frac{\partial(z_1, \dots, z_M)}{\partial(y_1, \dots, y_M)} \right|. \quad (14.9)$$

В качестве последнего примера использования техники преобразования рассмотрим метод Бокса–Мюллера (Box–Muller) для генерации выборок из гауссова распределения. Сначала сгенерируем пары равномерно распределенных случайных чисел $z_1, z_2 \in (-1, 1)$, что можно сделать путем преобразования переменной, равномерно распределенной по $(0, 1)$, при помощи $z \rightarrow 2z - 1$. Далее отбросим каждую пару, если она не удовлетворяет условию $z_1^2 + z_2^2 \leq 1$. Это приводит к равномерному распределению точек внутри единичной окружности с $p(z_1, z_2) = 1/\pi$, как показано на рис. 14.3. Затем для каждой пары z_1, z_2 оценим величины:

$$y_1 = z_1 \left(\frac{-2 \ln r^2}{r^2} \right)^{1/2}, \quad (14.10)$$

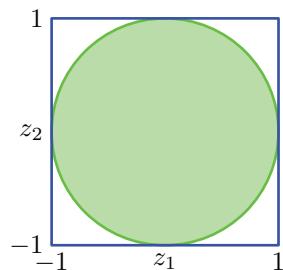
$$y_2 = z_2 \left(\frac{-2 \ln r^2}{r^2} \right)^{1/2}, \quad (14.11)$$

где $r^2 = z_1^2 + z_2^2$ (см. упражнение 14.5). Тогда совместное распределение y_1 и y_2 определяется:

$$\begin{aligned} p(y_1, y_2) &= p(z_1, z_2) \left| \frac{\partial(z_1, z_2)}{\partial(y_1, y_2)} \right| \\ &= \left[\frac{1}{\sqrt{2\pi}} \exp(-y_1^2/2) \right] \left[\frac{1}{\sqrt{2\pi}} \exp(-y_2^2/2) \right], \end{aligned} \quad (14.12)$$

и поэтому y_1 и y_2 независимы и имеют гауссово распределение с нулевым средним значением и единичной дисперсией.

РИС. 14.3 Метод Бокса–Мюллера для генерации случайных чисел с гауссовым распределением начинается с генерации выборок из равномерного распределения внутри единичного круга



Если \mathbf{y} имеет гауссово распределение с нулевым средним и единичной дисперсией, то $\sigma\mathbf{y} + \mu$ будет иметь гауссово распределение со средним значением μ и дисперсией σ^2 . Для генерации векторных переменных, имеющих многомерное гауссово распределение со средним μ и ковариацией Σ , можно воспользоваться *разложением Холецкого (Cholesky decomposition)*, которое имеет вид $\Sigma = \mathbf{L}\mathbf{L}^T$ (Deisenroth, Faisal and Ong, 2020). Тогда если \mathbf{z} – это случайный вектор, компоненты которого независимы и имеют гауссово распределение с нулевым средним значением и единичной дисперсией (см. упражнение 14.6), то $\mathbf{y} = \mu + \mathbf{L}\mathbf{z}$ будет гауссовым распределением со средним значением μ и ковариацией Σ .

Разумеется, успех метода преобразования зависит от возможности вычислить и затем инвертировать неопределенный интеграл требуемого распределения. Такие операции выполнимы лишь для ограниченного числа простых распределений, поэтому в поисках более общей стратегии следует обратиться к альтернативным подходам. Далее будут рассмотрены две техники, называемые *выборкой с отклонением (rejection sampling)* и *выборкой по важности (importance sampling)*. Несмотря на то что они ограничены преимущественно одномерными распределениями и, следовательно, не применимы напрямую к сложным задачам с большим количеством измерений, они являются важными компонентами более общих стратегий.

14.1.3. Выборка с отклонением

Концепция выборки с отклонением (*rejection sampling*) позволяет получать выборки из относительно сложных распределений при соблюдении определенных ограничений. Начнем с одномерных распределений, а затем перейдем к многомерным.

Предположим, что необходимо сделать выборку из распределения $p(\mathbf{z})$, которое не является ни одним из простых, стандартных распределений, рассмотренных до сих пор, и что выборка непосредственно из $p(\mathbf{z})$ затруднена. Кроме того, предположим, как это часто бывает, что для любого заданного значения \mathbf{z} можно просто оценить $p(\mathbf{z})$ в пределах некоторой нормализующей константы Z , так что

$$p(z) = \frac{1}{Z_p} \tilde{p}(z), \quad (14.13)$$

где оценить $\tilde{p}(z)$ несложно, но Z_p неизвестно.

Чтобы применить выборку с отклонением, нужно иметь некоторое более простое распределение $q(z)$, иногда называемое *вспомогательным распределением (proposal distribution)*, из которого достаточно просто взять выборку. Далее вводится константа k , значение которой выбирается таким образом, чтобы $kq(z) \geq \tilde{p}(z)$ для всех значений z . Функция $kq(z)$ называется *функцией сравнения (comparison function)*, и она показана для одномерного распределения на рис. 14.4. На каждом шаге выборки с отклонением генерируются два случайных числа. Сначала из распределения $q(z)$ генерируется число z_0 . Затем генерируется число u_0 из равномерного распределения по $[0, kq(z_0)]$. Эта пара случайных чисел имеет равномерное распределение по кривой функции $kq(z)$. Наконец, если $u_0 > \tilde{p}(z_0)$, то выборка отклоняется, в противном случае u_0 сохраняется. Таким образом, пара отклоняется, если она лежит в серой заштрихованной области на рис. 14.4. Оставшиеся пары имеют равномерное распределение по кривой $\tilde{p}(z)$ (см. упражнение 14.7), и, следовательно, соответствующие значения z распределены по $p(z)$, как и было задумано.

Исходные значения z генерируются из распределения $q(z)$, и эти выборки принимаются с вероятностью $\tilde{p}(z)/kq(z)$, поэтому вероятность того, что выборка будет принята, определяется как

$$\begin{aligned} p(\text{accept}) &= \int \{\tilde{p}(z)/kq(z)\} q(z) dz \\ &= \frac{1}{k} \int \tilde{p}(z) dz. \end{aligned} \quad (14.14)$$

Получается, что доля точек, которые отклоняются этим методом, зависит от отношения площади под ненормализованным распределением $\tilde{p}(z)$ к площади под кривой $kq(z)$. Поэтому очевидно, что постоянная k должна быть как можно меньше при условии, что $kq(z)$ нигде не должно быть меньше, чем $\tilde{p}(z)$.

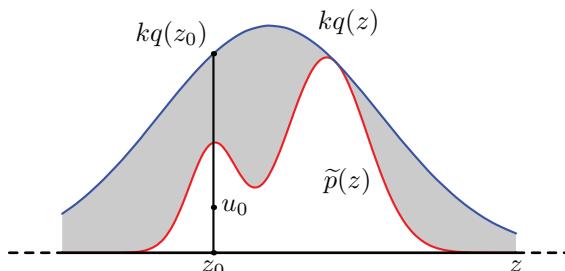


РИС. 14.4 В методе выборки с отклонением выборки берутся из простого распределения $q(z)$ и отклоняются, если они попадают в серую область между ненормализованным распределением $\tilde{p}(z)$ и масштабированным распределением $kq(z)$. Полученные выборки распределяются в соответствии с $p(z)$, которое является нормализованной версией $\tilde{p}(z)$

В качестве иллюстрации использования выборки с отклонением рассмотрим задачу выборки из гамма-распределения

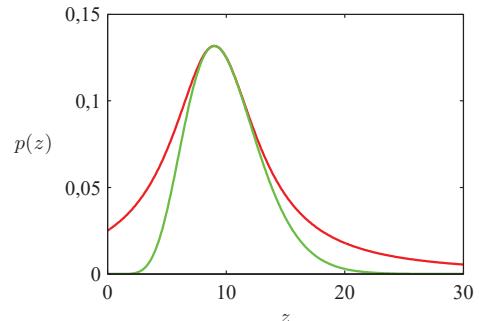
$$\text{Gam}(z|a, b) = \frac{b^a z^{a-1} \exp(-bz)}{\Gamma(a)}, \quad (14.15)$$

которое для $a > 1$ имеет колоколообразную форму, как показано на рис. 14.5. Подходящим вспомогательным распределением является распределение Коши (14.8), потому что оно тоже имеет колоколообразную форму и потому что для его выборки можно использовать рассмотренный выше метод преобразования. Необходимо немного обобщить распределение Коши, чтобы убедиться, что оно нигде не имеет меньшего значения, чем гамма-распределение. Для этого можно преобразовать равномерную случайную величину y с помощью $z = b \tan y + c$, что дает случайные числа, распределенные (см. упражнение 14.8) в соответствии с

$$q(z) = \frac{k}{1 + (z - c)^2/b^2}. \quad (14.16)$$

Минимальный процент отклонения получается, если задать $c = a - 1$, $b^2 = 2a - 1$ и выбрать константу k как можно меньшей, но при этом удовлетворяющей требованию $kq(z) > p(z)$. Полученная функция сравнения также показана на рис. 14.5.

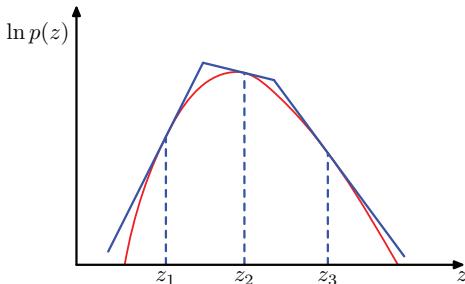
РИС. 14.5 График, на котором гамма-распределение, заданное в (14.15), показано зеленой кривой, а масштабированное распределение предложения Коши – красной кривой. Выборки из гамма-распределения могут быть получены путем выборки из распределения Коши с последующим применением критерия выборки с отклонением



14.1.4. Адаптивная выборка с отклонением

Во многих случаях, когда возникает необходимость применить выборку с отклонением, бывает трудно найти подходящую аналитическую форму для огибающей распределения $q(z)$. Альтернативным подходом является построение огибающей функции «на лету» на основе измеренных значений распределения $p(z)$ (Gilks and Wild, 1992). Построение огибающей функции становится особенно простым, когда $p(z)$ является логарифмически вогнутой, другими словами, когда $\ln p(z)$ имеет производные, которые являются невозрастающими функциями от z . Построение подходящей огибающей функции графически показано на рис. 14.6.

РИС. 14.6 При выборке с отклонением, если распределение является логарифмически вогнутым, можно построить огибающую функцию по касательным линиям, вычисленным в наборе точек сетки. Если точка выборки отклоняется, она добавляется к набору точек сетки и используется для уточнения огибающей распределения



Функция $\ln p(z)$ и ее градиент оцениваются в некотором начальном наборе точек сетки, а пересечения полученных касательных линий используются для построения огибающей функции. Затем из огибающей распределения берется выборочное значение. Это несложно (см. упражнение 14.10), поскольку логарифм распределения огибающей представляет собой последовательность линейных функций, и поэтому само распределение огибающей представляет собой фрагментарное экспоненциальное распределение вида

$$q(z) = k_i \lambda_i \exp\{-\lambda_i(z - z_{i-1})\}, \quad z_{i-1} < z \leq z_i. \quad (14.17)$$

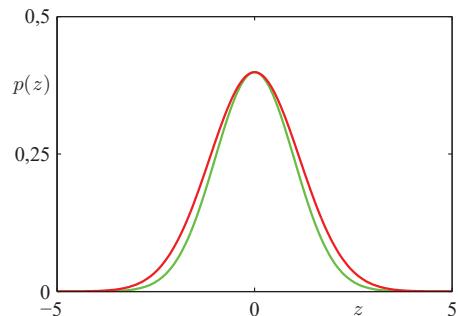
После получения выборки можно применить обычный критерий отклонения. Если выборка будет принята, следовательно, она будет взята из желаемого распределения. Если же выборка отвергается, то она включается в набор точек сетки, вычисляется новая касательная линия, и, таким образом, функция огибающей уточняется. С увеличением числа точек сетки функция огибающей становится более точной аппроксимацией желаемого распределения $p(z)$, а вероятность отклонения уменьшается.

Существует вариант алгоритма, позволяющий обойтись без оценки производных (Gilks, 1992). Адаптивную выборку с отклонением можно распространить и на распределения, которые не являются логарифмически вогнутыми, просто сопровождая каждый шаг выборки с отклонением шагом Метрополиса–Хастингса (будет обсуждаться в разделе 14.2.3), что приводит к *адаптивной выборке Метрополиса с отклонением* (*adaptive rejection Metropolis sampling*) (Gilks, Best and Tan, 1995).

Для практической ценности выборки с отклонением необходимо, чтобы функция сравнения была близка к требуемому распределению и чтобы процент отклонений был минимальным. Теперь рассмотрим, что происходит при попытке использовать выборку с отклонением в пространствах высокой размерности. Для примера возьмем довольно искусственную задачу, где необходимо сделать выборку из нулевого усредненного многомерного гауссова распределения с ковариацией $\sigma_p^2 \mathbf{I}$, где \mathbf{I} – это единичная матрица, путем выборки с отклонением из промежуточного распределения, которое само является нулевым усредненным гауссовым распределением с ковариацией $\sigma_q^2 \mathbf{I}$. Разумеется, необходимо, чтобы $\sigma_q^2 \geq \sigma_p^2$ гарантировало, что существует такое k , что $kq(z) \geq p(z)$. В D -размерности оптимальное значение k задается $k = (\sigma_q/\sigma_p)^D$, как показано для $D = 1$ на рис. 14.7. Коэффициент принятия будет равен отно-

шению объемов под $p(z)$ и $kq(z)$, которое, поскольку оба распределения нормализованы, равно $1/k$. Таким образом, коэффициент принятия уменьшается экспоненциально с увеличением размерности. Даже если σ_q превышает σ_p всего на 1 %, для $D = 1000$ коэффициент принятия будет равен примерно $1/20\,000$. В этом наглядном примере функция сравнения близка к требуемому распределению. Для более практических примеров, где требуемое распределение может быть мультимодальным и с резким пиком, будет крайне сложно найти хорошее промежуточное распределение и функцию сравнения. Кроме того, экспоненциальное уменьшение коэффициента принятия с ростом размерности является общим свойством выборки с отклонением. Хотя отклонение может быть полезной процедурой в одном или двух измерениях, для задач высокой размерности оно не подходит. Однако оно может играть роль подпрограммы в более сложных алгоритмах выборки в пространствах высокой размерности.

РИС. 14.7 Иллюстративный пример показывает ограничение выборки с отклонением. Выборки берутся из гауссова распределения $p(z)$, показанного зеленой кривой, с помощью выборки с отклонением из распределения предложений $q(z)$, которое также является гауссовым и чья масштабированная версия $kq(z)$ показана красной кривой



14.1.5. Выборка по важности

Одной из причин, по которой возникает необходимость в выборке из сложных распределений вероятностей, является оценка ожиданий вида (14.1). Метод *выборки по важности* (*importance sampling*) предоставляет возможность аппроксимировать ожидания напрямую, но не обеспечивает самого механизма выборки из распределения $p(\mathbf{z})$.

Приближение конечной суммы к ожиданию, задаваемое в (14.2), зависит от возможности получать выборки из распределения $p(\mathbf{z})$. Предположим тем не менее, что выборка непосредственно из $p(\mathbf{z})$ непрактична, но зато для любого заданного значения \mathbf{z} можно быстро оценить $p(\mathbf{z})$. Одна из упрощенных стратегий оценки ожиданий заключается в дискретизации пространства \mathbf{z} на равномерную сетку и вычислении подынтегрального выражения в виде суммы вида

$$\mathbb{E}[f] \approx \sum_{l=1}^L p(\mathbf{z}^{(l)}) f(\mathbf{z}^{(l)}). \quad (14.18)$$

Очевидная проблема такого подхода заключается в том, что количество членов в суммировании растет экспоненциально с увеличением размер-

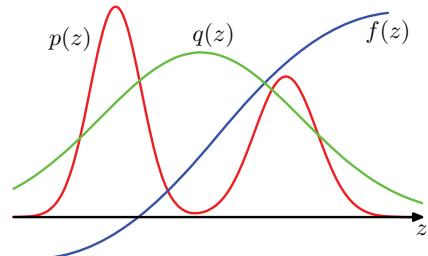
ности \mathbf{z} . Кроме того, как уже отмечалось, в интересующих вероятностных распределениях большая часть массы часто будет приходиться на относительно небольшие области пространства \mathbf{z} , и поэтому равномерная выборка будет малоэффективной, поскольку в задачах с высокой размерностью только очень малая часть выборок будет вносить существенный вклад в общую сумму. В действительности желательно выбирать точки выборки из областей, где $p(\mathbf{z})$ велико или – в идеале – где велико произведение $p(\mathbf{z})f(\mathbf{z})$.

Как и в случае выборки с отклонением, выборка по важности основана на вспомогательном распределении $q(\mathbf{z})$, из которого легко взять выборки, как показано на рис. 14.8. Тогда ожидание можно выразить в виде конечной суммы по выборкам $\{\mathbf{z}^{(l)}\}$, взятым из $q(\mathbf{z})$:

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} \\ &= \int f(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z})d\mathbf{z} \\ &\approx \frac{1}{L} \sum_{l=1}^L \frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})} f(\mathbf{z}^{(l)}).\end{aligned}\quad (14.19)$$

Величины $r_l = p(\mathbf{z}^{(l)})/q(\mathbf{z}^{(l)})$ известны как *веса важности* (*importance weight*), и они корректируют смещение, вносимое выборкой из неправильного распределения. Обратите внимание, что, в отличие от выборки с отклонением, здесь сохраняются все сгенерированные выборки.

РИС. 14.8 Выборка по важности решает проблему оценки ожидания функции $f(z)$ относительно распределения $p(z)$, из которого трудно взять выборки напрямую. Вместо этого выборки $\{\mathbf{z}^{(l)}\}$ берутся из более простого распределения $q(z)$, а соответствующие члены в суммировании взвешиваются по коэффициентам $p(\mathbf{z}^{(l)})/q(\mathbf{z}^{(l)})$



Зачастую распределение $p(\mathbf{z})$ может быть оценено только до константы нормализации, так что $p(\mathbf{z}) = \tilde{p}(\mathbf{z})/Z_p$, где $\tilde{p}(\mathbf{z})$ оценивается легко, в то время как Z_p остается неизвестным. Аналогичным образом можно использовать распределение выборки по важности $q(\mathbf{z}) = \tilde{q}(\mathbf{z})/Z_q$, которое обладает тем же свойством. Тогда получается:

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} \\ &= \frac{Z_q}{Z_p} \int f(\mathbf{z}) \frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})} q(\mathbf{z})d\mathbf{z} \\ &\approx \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^L \tilde{r}_l f(\mathbf{z}^{(l)}),\end{aligned}\quad (14.20)$$

где $\tilde{\mathbf{r}}_l = \tilde{p}(\mathbf{z}^{(l)})/\tilde{q}(\mathbf{z}^{(l)})$. С помощью того же набора выборок можно оценить отношение Z_p/Z_q с результатом

$$\begin{aligned}\frac{Z_p}{Z_q} &= \frac{1}{Z_q} \int \tilde{p}(\mathbf{z}) d\mathbf{z} = \int \frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})} q(\mathbf{z}) d\mathbf{z} \\ &\approx \frac{1}{L} \sum_{l=1}^L \tilde{r}_l,\end{aligned}\quad (14.21)$$

и, следовательно, ожидание в (14.20) определяется взвешенной суммой:

$$\mathbb{E}[f] \approx \sum_{l=1}^L w_l f(\mathbf{z}^{(l)}), \quad (14.22)$$

где определено

$$w_l = \frac{\tilde{r}_l}{\sum_m \tilde{r}_m} = \frac{\tilde{p}(\mathbf{z}^{(l)})/q(\mathbf{z}^{(l)})}{\sum_m \tilde{p}(\mathbf{z}^{(m)})/q(\mathbf{z}^{(m)})}. \quad (14.23)$$

Обратите внимание, что $\{w_l\}$ – это неотрицательные числа, сумма которых равна единице.

Как и в случае выборки с отклонением, эффективность выборки по важности зависит от того, насколько хорошо распределение выборки $q(\mathbf{z})$ соответствует желаемому распределению $p(\mathbf{z})$. Если, как часто бывает, $p(\mathbf{z})f(\mathbf{z})$ сильно изменяется и значительная часть его массы сосредоточена в относительно небольших областях пространства \mathbf{z} , то в наборе весов важности $\{r_l\}$ могут доминировать несколько весов с большими значениями, а остальные веса будут относительно малозначимыми. Таким образом, эффективный размер выборки может быть намного меньше, чем кажущийся размер выборки L . Проблема становится еще более серьезной, если ни одна из выборок не попадает в области, где $p(\mathbf{z})f(\mathbf{z})$ велика. В этом случае кажущиеся дисперсии r_l и $r_l f(\mathbf{z}^{(l)})$ могут быть малы, хотя оценка математического ожидания может быть сильно ошибочной. Получается, что основным недостатком выборки по важности является возможность получения результатов с произвольными ошибками без каких-либо диагностических признаков. Это также подчеркивает ключевое требование к распределению выборки $q(\mathbf{z})$, а именно: оно не должно быть малым или нулевым в областях, где $p(\mathbf{z})$ может быть значимым.

14.1.6. Выборка и повторная выборка по значимости

Метод выборки с отклонением, рассмотренный в разделе 14.1.3, частично зависит от определения подходящего значения константы k . Для многих пар распределений $p(\mathbf{z})$ и $q(\mathbf{z})$ определить подходящее значение k практически невозможно, поскольку любое значение, достаточно большое, чтобы гарантировать границу желаемого распределения, приведет к неприемлемо малым коэффициентам принятия.

Как и в случае выборки с отклонением, метод *выборки по значимости с повторной выборкой* (*sampling-importance-resampling, SIR*) также основан на использовании выборочного распределения $q(\mathbf{z})$, но позволяет избежать задачи определения константы k . В этой схеме есть два этапа. На первом этапе L выборок $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(L)}$ отбираются из $q(\mathbf{z})$. На втором этапе веса w_1, \dots, w_L определяются с помощью (14.23). Наконец, второй набор из L выборок берется из дискретного распределения $(\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(L)})$ с вероятностями, заданными весами (w_1, \dots, w_L) .

Полученные L выборок лишь приблизительно распределены по $p(\mathbf{z})$, но распределение получается правильным в пределе $L \rightarrow \infty$. Чтобы убедиться в этом, рассмотрим одномерный случай и отметим, что кумулятивное распределение повторно отобранных значений имеет вид:

$$\begin{aligned} p(z \leq a) &= \sum_{l: z^{(l)} \leq a} w_l \\ &= \frac{\sum_l I(z^{(l)} \leq a) \tilde{p}(z^{(l)}) / q(z^{(l)})}{\sum_l \tilde{p}(z^{(l)}) / q(z^{(l)})}, \end{aligned} \quad (14.24)$$

где $I(\cdot)$ – это индикаторная функция (которая равна 1, если ее аргумент истинный, и 0 в противном случае). Взяв предел $L \rightarrow \infty$ и предполагая подходящую регулярность распределений, можно заменить суммы интегралами, взвешенными в соответствии с исходным распределением выборки $q(\mathbf{z})$:

$$\begin{aligned} p(z \leq a) &= \frac{\int I(z \leq a) \{\tilde{p}(z)/q(z)\} q(z) dz}{\int \{\tilde{p}(z)/q(z)\} q(z) dz} \\ &= \frac{\int I(z \leq a) \tilde{p}(z) dz}{\int \tilde{p}(z) dz} \\ &= \int I(z \leq a) p(z) dz, \end{aligned} \quad (14.25)$$

которая является кумулятивной функцией распределения $p(z)$. И вновь становится очевидным, что нормализация $p(z)$ не требуется.

Для конечного значения L и заданного начального набора выборок повторно отобранные значения будут лишь приблизительно взяты из желаемого распределения. Как и в случае выборки с отклонением, приближение улучшается по мере приближения выборочного распределения $q(\mathbf{z})$ к желаемому распределению $p(\mathbf{z})$. В случае $q(\mathbf{z}) = p(\mathbf{z})$ начальные выборки $(\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(L)})$ имеют желаемое распределение и веса $w_n = 1/L$, так что повторно отобранные значения также имеют желаемое распределение.

Если необходимы моменты относительно распределения $p(z)$, то они могут быть оценены непосредственно по исходным выборкам вместе с весами, поскольку

$$\begin{aligned}
 \mathbb{E}(f(\mathbf{z})) &= \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} \\
 &= \frac{\int f(\mathbf{z})[\tilde{p}(\mathbf{z})/q(\mathbf{z})]q(\mathbf{z})d\mathbf{z}}{\int [\tilde{p}(\mathbf{z})/q(\mathbf{z})]q(\mathbf{z})d\mathbf{z}} \\
 &\approx \sum_{l=1}^L w_l f(\mathbf{z}_l).
 \end{aligned} \tag{14.26}$$

14.2. Метод Монте-Карло с цепями Маркова

В предыдущем разделе были рассмотрены стратегии выборки с отклонением и выборки по важности для оценки ожиданий функций, и выяснилось, что они обладают серьезными ограничениями, особенно в пространствах высокой размерности. Поэтому в этом разделе рассматривается очень общая и мощная схема, называемая *методом Монте-Карло с цепями Маркова* (*Markov chain Monte Carlo, Марковские цепи Монте-Карло*), которая предоставляет возможность выборки из большого класса распределений и хорошо масштабируется при увеличении размерности пространства выборки. Методы Монте-Карло с цепями Маркова берут свое начало в физике (Metropolis and Ulam, 1949), и только к концу 1980-х годов они получили значительное распространение в области статистики.

Как и в случае с выборкой по отклонению и выборкой по важности, здесь вновь используется выборка из распределения предложений. Однако на этот раз сохраняется запись о текущем состоянии $\mathbf{z}^{(r)}$, где распределение предложений $q(\mathbf{z} \mid \mathbf{z}^{(r)})$ обусловлено этим текущим состоянием, и поэтому последовательность выборок $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots$ образует цепь Маркова (см. раздел 14.2.2). Опять же, если записать $p(\mathbf{z}) = \tilde{p}(\mathbf{z})/Z_p$, то можно предположить, что $\tilde{p}(\mathbf{z})$ может быть легко оценена для любого заданного значения \mathbf{z} , хотя значение Z_p может быть неизвестно. Вспомогательное распределение выбирается достаточно простым, чтобы было удобно брать из него выборки напрямую. На каждом цикле работы алгоритма из распределения предложений генерируется выборка-кандидат \mathbf{z}^* , которая затем принимается в соответствии с определенным критерием.

14.2.1. Алгоритм Метрополиса

В базовом алгоритме *Метрополиса* (*Metropolis algorithm*) (Metropolis et al., 1953) подразумевается, что вспомогательное распределение симметрично, т. е. $q(\mathbf{z}_A \mid \mathbf{z}_B) = q(\mathbf{z}_B \mid \mathbf{z}_A)$ для всех значений \mathbf{z}_A и \mathbf{z}_B . Тогда выборка-кандидат принимается с вероятностью:

$$A(\mathbf{z}^*, \mathbf{z}^{(r)}) = \min\left(1, \frac{\tilde{p}(\mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(r)})}\right). \tag{14.27}$$

Этого можно достичь путем выбора случайного числа u с равномерным распределением на единичном интервале $(0, 1)$ и последующего принятия выборки при $A(\mathbf{z}^*, \mathbf{z}^{(t)}) > u$. Обратите внимание, что если шаг от $\mathbf{z}^{(t)}$ к \mathbf{z}^* приводит к увеличению значения $p(\mathbf{z})$, то точка-кандидат обязательно будет сохранена.

Если выборка-кандидат принята, то $\mathbf{z}^{(t+1)} = \mathbf{z}^*$, в противном случае точка-кандидат \mathbf{z}^* отклоняется, $\mathbf{z}^{(t+1)}$ устанавливается равной $\mathbf{z}^{(t)}$, и из распределения $q(\mathbf{z} | \mathbf{z}^{(t+1)})$ берется другая выборка-кандидат. Это отличается от выборки с отклонением, когда отвергнутые выборки просто отбрасываются. В алгоритме Метрополиса, когда точка-кандидат отвергается, предыдущая выборка включается в конечный список выборок, что приводит к множественным копиям выборок. Конечно, в практической реализации будет храниться только одна копия каждой сохраненной выборки, а также целочисленный весовой коэффициент, фиксирующий количество появлений этого состояния. Как будет показано далее, если $q(\mathbf{z}_A | \mathbf{z}_B)$ положительно при любых значениях \mathbf{z}_A и \mathbf{z}_B (это достаточно, но не необходимое условие), то распределение $\mathbf{z}^{(t)}$ стремится к $p(\mathbf{z})$ при $t \rightarrow \infty$. Следует все же подчеркнуть, что последовательность $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots$ не является набором независимых выборок из $p(\mathbf{z})$, поскольку последовательные выборки сильно коррелированы. Если необходимо получить независимые выборки, тогда можно отбросить большую часть последовательности и сохранить только каждую M -ю выборку. При достаточно большом числе M сохраняемые выборки будут независимыми в любом практическом случае. Суть алгоритма Метрополиса кратко изложена в алгоритме 14.1. На рис. 14.9 показан простой пример выборки из двумерного гауссова распределения с помощью алгоритма Метрополиса, где вспомогательное распределение является изотропным гауссовым.

АЛГОРИТМ 14.1 Алгоритм выборки Метрополиса

Input: ненормализованное распределение $\tilde{p}(\mathbf{z})$

Вспомогательное распределение $q(\mathbf{z} | \hat{\mathbf{z}})$

Начальное состояние $\mathbf{z}^{(0)}$

Количество итераций T

Output: $\mathbf{z} \sim \tilde{p}(\mathbf{z})$

$\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}^{(0)}$

// Итеративная передача сообщений

for $t \in \{1, \dots, T\}$ **do**

$\mathbf{z}^* \sim q(\mathbf{z} | \mathbf{z}_{\text{prev}})$ // Выборка из вспомогательного распределения

$u \sim \mathcal{U}(0, 1)$ // Выборка из равномерного распределения

if $\tilde{p}(\mathbf{z}^*) / \tilde{p}(\mathbf{z}_{\text{prev}}) > u$ **then**

$\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}^*$ // $\mathbf{z}^{(t)} = \mathbf{z}^*$

else

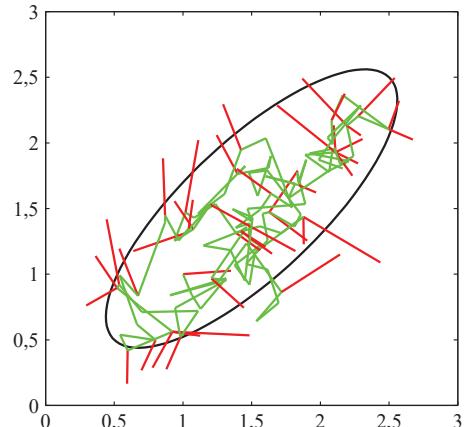
$\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}_{\text{prev}}$ // $\mathbf{z}^{(t)} = \mathbf{z}^{(t-1)}$

end if

end for

return $\mathbf{z}_{\text{prev}} // \mathbf{z}^{(T)}$

РИС. 14.9 Простая иллюстрация использования алгоритма Метрополиса для выборки из гауссова распределения, контур которого с одним стандартным отклонением показан эллипсом. Промежуточное распределение – это изотропное гауссово распределение со стандартным отклонением 0,2. Принятые шаги показаны зелеными линиями, а отклоненные – красными. Всего генерируется 150 выборок-кандидатов, из которых 43 отвергаются



Более глубокое понимание природы алгоритмов марковской цепи Монте-Карло можно получить на конкретном примере, а именно примере так называемого *простого случайного блуждания* (*simple random walk*). Рассмотрим пространство состояний z из целых чисел с вероятностями:

$$p(z^{(\tau+1)} = z^{(\tau)}) = 0,5, \quad (14.28)$$

$$p(z^{(\tau+1)} = z^{(\tau)} + 1) = 0,25, \quad (14.29)$$

$$p(z^{(\tau+1)} = z^{(\tau)} - 1) = 0,25, \quad (14.30)$$

где $z^{(\tau)}$ обозначает состояние на шаге τ . Если начальное состояние $z^{(0)} = 0$, то в силу симметрии ожидаемое состояние в момент времени τ также будет равно нулю $\mathbb{E}[z^{(\tau)}] = 0$, и точно так же видно, что $\mathbb{E}[(z^{(\tau)})^2] = \tau / 2$ (см. упражнение 14.11). Таким образом, после τ шагов случайное блуждание прошло лишь расстояние, которое в среднем пропорционально квадратному корню из τ . Эта зависимость от квадратного корня типична для поведения случайных блужданий и показывает, что случайные блуждания очень неэффективны при исследовании пространства состояний. Как будет показано далее, одной из главных задач при разработке методов марковской цепи Монте-Карло является предотвращение возникновения случайных блужданий.

14.2.2. Марковские цепи

До начала более детального знакомства с методами марковских цепей Монте-Карло будет полезно изучить некоторые общие свойства марковских цепей. В частности, возникает вопрос, при каких обстоятельствах цепь Маркова сходится к желаемому распределению. Цепь Маркова первого порядка определяется как ряд случайных величин $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(M)}$ таких, что для $m \in \{1, \dots, M-1\}$:

$$p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}) = p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)}), \quad (14.31)$$

что можно представить как направленную графовую модель в виде цепи на рис. 11.29. Тогда цепь Маркова можно задать распределением вероятностей для начальной переменной $p(\mathbf{z}^{(0)})$ вместе с условными распределениями для последующих переменных в виде *вероятностей перехода* $T_m(\mathbf{z}^{(m)}, \mathbf{z}^{(m+1)}) \equiv p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)})$. Марковская цепь называется *однородной* (*homogeneous*) в том случае, если вероятности перехода одинаковы для всех m .

Маргинальная вероятность для конкретной переменной может быть выражена в виде маргинальной вероятности для предыдущей переменной в цепи:

$$p(\mathbf{z}^{(m+1)}) = \int p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)}) p(\mathbf{z}^{(m)}) d\mathbf{z}^{(m)}, \quad (14.32)$$

где интеграл заменен суммированием по дискретным переменным. Распределение считается инвариантным, или стационарным, по отношению к цепи Маркова, если каждый шаг в цепи оставляет это распределение инвариантным. Так, для однородной цепи Маркова с вероятностями перехода $T(\mathbf{z}', \mathbf{z})$ распределение $p^*(\mathbf{z})$ инвариантно, если

$$p^*(\mathbf{z}) = \int T(\mathbf{z}', \mathbf{z}) p^*(\mathbf{z}') d\mathbf{z}'. \quad (14.33)$$

Обратите внимание, что у данной цепи Маркова может быть более одного инвариантного распределения. Например, если вероятности перехода задаются преобразованием тождества, то любое распределение будет инвариантным.

Достаточным (но не необходимым) условием обеспечения инвариантности требуемого распределения $p(\mathbf{z})$ является выбор вероятностей перехода, удовлетворяющих свойству *детального равновесия* (*detailed balance*), которое определяется как

$$p^*(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = p^*(\mathbf{z}') T(\mathbf{z}', \mathbf{z}) \quad (14.34)$$

для конкретного распределения $p^*(\mathbf{z})$. Отсюда видно, что вероятность перехода, удовлетворяющаяциальному балансу для конкретного распределения, оставляет это распределение инвариантным, поскольку

$$\int p^*(\mathbf{z}') T(\mathbf{z}', \mathbf{z}) d\mathbf{z}' = \int p^*(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') d\mathbf{z}' \quad (14.35)$$

$$= p^*(\mathbf{z}) \int p(\mathbf{z}' | \mathbf{z}) d\mathbf{z}' \quad (14.36)$$

$$= p^*(\mathbf{z}). \quad (14.37)$$

Цепь Маркова, в которой соблюдается детальное равновесие, называется *обратимой* (*reversible*).

Нашей целью является использование цепей Маркова для выборки из заданного распределения. Этого можно добиться с помощью такой цепи Маркова, чтобы желаемое распределение было инвариантным. Однако при этом необходимо, чтобы для $m \rightarrow \infty$ распределение $p(\mathbf{z}^{(m)})$ сходилось к требуемому инвариантному распределению $p^*(\mathbf{z})$ независимо от выбора начального распределения $p(\mathbf{z}^{(0)})$. Это свойство называется *эргодичностью* (*ergodicity*),

а инвариантное распределение – *равновесным распределением (equilibrium distribution)*. Разумеется, эргодическая марковская цепь может иметь только одно равновесное распределение. Можно также доказать, что однородная марковская цепь будет эргодической при условии слабых ограничений на инвариантное распределение и вероятности перехода (Neal, 1993).

На практике вероятности переходов зачастую определяются из набора «базовых» переходов B_1, \dots, B_K . Для этого можно использовать смешанное распределение вида

$$T(\mathbf{z}', \mathbf{z}) = \sum_{k=1}^K \alpha_k B_k(\mathbf{z}', \mathbf{z}) \quad (14.38)$$

для некоторого набора коэффициентов смешивания $\alpha_1, \dots, \alpha_K$, удовлетворяющих условию $\alpha_k \geq 0$ и $\sum_k \alpha_k = 1$. В качестве альтернативы базовые переходы могут быть объединены путем их последовательной обработки, так что

$$T(\mathbf{z}', \mathbf{z}) = \sum_{\mathbf{z}_1} \dots \sum_{\mathbf{z}_{n-1}} B_1(\mathbf{z}', \mathbf{z}_1) \dots B_{K-1}(\mathbf{z}_{K-2}, \mathbf{z}_{K-1}) B_K(\mathbf{z}_{K-1}, \mathbf{z}). \quad (14.39)$$

Если распределение инвариантно относительно каждого из базовых переходов, то очевидно, что оно также будет инвариантно относительно любого из $T(\mathbf{z}', \mathbf{z})$, заданных в (14.38) или в (14.39). Если каждый из базовых переходов удовлетворяет детальному балансу, то и переход T для смеси (14.38) будет удовлетворять детальному балансу. Этого нельзя сказать о вероятности перехода, построенной с помощью (14.39), хотя, если симметрично изменить порядок применения базовых переходов, а именно $B_1, B_2, \dots, B_K, B_K, \dots, B_2, B_1$, детальное равновесие вполне может быть восстановлено. Частым примером использования составных вероятностей перехода является случай, когда каждый базовый переход приводит к изменению только подмножества переменных.

14.2.3. Алгоритм Метрополиса–Гастингса

Ранее был представлен базовый алгоритм Метрополиса без демонстрации того факта, что он берет выборку из нужного распределения. Прежде чем привести доказательство, рассмотрим обобщение под названием *алгоритм Метрополиса–Гастингса (Metropolis-Hastings algorithm)* (Hastings, 1970), который применяется в тех случаях, когда вспомогательное распределение уже не является симметричной функцией своих аргументов. В частности, на шаге τ алгоритма, на котором актуальным состоянием является $\mathbf{z}^{(\tau)}$, из распределения $q_k(\mathbf{z} | \mathbf{z}^{(\tau)})$ берется выборка \mathbf{z}^* и затем принимается с вероятностью $A_k(\mathbf{z}^*, \mathbf{z}^{(\tau)})$, где

$$A_k(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min \left(1, \frac{\tilde{p}(\mathbf{z}^*) q_k(\mathbf{z}^{(\tau)} | \mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)}) q_k(\mathbf{z}^* | \mathbf{z}^{(\tau)})} \right). \quad (14.40)$$

Здесь k маркирует члены рассматриваемого множества возможных переходов. И вновь для оценки критерия приемлемости не требуется знать нормализующую константу Z_p в распределении вероятностей $p(\mathbf{z}) = \tilde{p}(\mathbf{z})/Z_p$. Для симметричного вспомогательного распределения критерий Метрополиса–Гастингса (14.40) сводится к стандартному критерию Метрополиса, представленному в (14.27). Выборка Метрополиса–Гастингса кратко изложена в алгоритме 14.2.

АЛГОРИТМ 14.2 Выборка Метрополиса–Гастингса

Input: ненормализованное распределение $\tilde{p}(\mathbf{z})$

Вспомогательные распределения $\{q_k(\mathbf{z} | \hat{\mathbf{z}}) : k \in 1, \dots, K\}$

Отображение индекса итерации на индекс распределения $M(\cdot)$

Начальное состояние $\mathbf{z}^{(0)}$

Количество итераций T

Output: $\mathbf{z} \sim \tilde{p}(\mathbf{z})$

$\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}^{(0)}$

// Итеративная передача сообщений

for $\tau \in \{1, \dots, T\}$ **do**

$k \leftarrow M(\tau)$ // Получение индекса распределения для этой итерации

$\mathbf{z}^* \sim q_k(\mathbf{z} | \mathbf{z}_{\text{prev}})$ // Выборка из вспомогательного распределения

$u \sim \mathcal{U}(0, 1)$ // Выборка из равномерного распределения

if $\tilde{p}(\mathbf{z}^*)q(\mathbf{z}_{\text{prev}} | \mathbf{z}^*)/\tilde{p}(\mathbf{z}_{\text{prev}})q(\mathbf{z}^* | \mathbf{z}_{\text{prev}}) > u$ **then**

$\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}^*$ // $\mathbf{z}^{(\tau)} = \mathbf{z}^*$

else

$\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}_{\text{prev}}$ // $\mathbf{z}^{(\tau)} = \mathbf{z}^{(\tau-1)}$

end if

end for

return \mathbf{z}_{prev} // $\mathbf{z}^{(T)}$

Теперь можно доказать, что $p(\mathbf{z})$ является инвариантным распределением цепи Маркова, определенным алгоритмом Метрополиса–Гастингса, убедившись в справедливости детального баланса, заданного в (14.34). Используя (14.40), получаем:

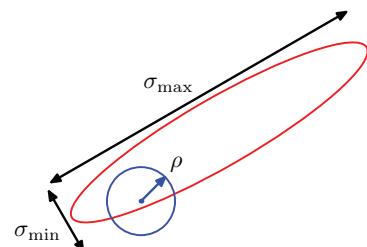
$$\begin{aligned} p(\mathbf{z})q_k(\mathbf{z}' | \mathbf{z})A_k(\mathbf{z}', \mathbf{z}) &= \min(p(\mathbf{z})q_k(\mathbf{z}' | \mathbf{z}), p(\mathbf{z}')q_k(\mathbf{z} | \mathbf{z}')) \\ &= \min(p(\mathbf{z}')q_k(\mathbf{z} | \mathbf{z}'), p(\mathbf{z})q_k(\mathbf{z}' | \mathbf{z})) \\ &= p(\mathbf{z}')q_k(\mathbf{z} | \mathbf{z}')A_k(\mathbf{z}, \mathbf{z}'), \end{aligned} \quad (14.41)$$

как и требовалось.

Выбор конкретного вспомогательного распределения может оказывать заметное влияние на производительность алгоритма. Для непрерывных пространств состояний общим выбором является гауссово распределение с центром на актуальном состоянии, что приводит к важному компромиссу при определении параметра дисперсии этого распределения. Если дисперсия

мала, то доля принятых переходов будет высокой, но продвижение по пространству состояний будет иметь форму медленного случайного блуждания и приведет к длительному времени корреляции. Однако если параметр дисперсии велик, то доля отказов будет высокой, поскольку в рассматриваемых сложных задачах многие из предлагаемых шагов будут приходиться на состояния, для которых вероятность $p(\mathbf{z})$ мала. Рассмотрим многомерное распределение $p(\mathbf{z})$ с сильными корреляциями между компонентами \mathbf{z} , как показано на рис. 14.10. Масштаб ρ вспомогательного распределения должен быть настолько большим, насколько это возможно, чтобы не получить высокий процент отказов. Это предполагает, что ρ должен быть того же порядка, что и наименьший масштаб длины σ_{\min} . Затем система исследует распределение вдоль более вытянутого направления с помощью случайного блуждания, и поэтому число шагов для достижения состояния, которое более или менее независимо от исходного состояния, имеет порядок $(\sigma_{\max}/\sigma_{\min})^2$. На самом деле в двух измерениях увеличение числа отказов с ростом ρ компенсируется большими размерами шагов для тех переходных процессов, которые принимаются, а в более общем случае для многомерного гауссова распределения число шагов, необходимых для получения независимых выборок, имеет масштаб $(\sigma_{\max}/\sigma^2)^2$, где σ^2 – это второе наименьшее стандартное отклонение (Neal, 1993). Если не принимать во внимание эти детали, то если шкалы длин, на которых меняются распределения, сильно отличаются в разных направлениях, то алгоритм Метрополиса–Гастингса может иметь очень медленную сходимость.

РИС. 14.10 Схематическое изображение использования изотропного вспомогательного гауссова распределения (синий круг) для выборки из коррелированного многомерного гауссова распределения (красный эллипс), имеющего очень разные стандартные отклонения в разных направлениях, с помощью алгоритма Метрополиса–Гастингса. Чтобы процент отказов был низким, масштаб ρ распределения предложений должен быть порядка наименьшего стандартного отклонения σ_{\min} , а это приводит к поведению случайного блуждания, при котором число шагов, разделяющих приблизительно независимые состояния, имеет порядок $(\sigma_{\max}/\sigma_{\min})^2$, где σ_{\max} – это наибольшее стандартное отклонение



14.2.4. Выборка Гиббса

Выборка Гиббса (*Gibbs sampling*) (Geman and Geman, 1984) – это простой и широко используемый алгоритм марковской цепи Монте-Карло, который можно рассматривать как частный случай алгоритма Метрополиса–Гастингса. Рассмотрим распределение $p(\mathbf{z}) = p(z_1, \dots, z_M)$, из которого необходимо сделать выборку, и предположим, что для цепи Маркова выбрано некоторое началь-

ное состояние. Каждый шаг процедуры выборки Гиббса заключается в замене значения одной из переменных значением, взятым из распределения этой переменной, обусловленного значениями остальных переменных. Таким образом, z_i заменяется значением, взятым из распределения $p(z_i | \mathbf{z}_{\setminus i})$, где z_i обозначает i -ю компоненту \mathbf{z} , а $\mathbf{z}_{\setminus i}$ обозначает $\{z_1, \dots, z_M\}$, но без z_i . Эта процедура повторяется либо путем циклического перебора переменных в определенном порядке, либо путем случайного выбора из некоторого распределения переменной, которая будет обновляться на каждом шаге.

Допустим, имеется распределение $p(z_1, z_2, z_3)$ по трем переменным, и на шаге τ алгоритма были выбраны значения $z_1^{(\tau)}, z_2^{(\tau)}$ и $z_3^{(\tau)}$. Сначала заменим $z_1^{(\tau)}$ на новое значение $z_1^{(\tau+1)}$, полученное путем выборки из условного распределения

$$p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}). \quad (14.42)$$

Далее заменим $z_2^{(\tau)}$ на значение $z_2^{(\tau+1)}$, полученное путем выборки из условного распределения

$$p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}), \quad (14.43)$$

так что новое значение для z_1 немедленно используется в последующих шагах выборки. Затем обновим z_3 с помощью выборки $z_3^{(\tau+1)}$, взятой из

$$p(z_3 | z_1^{(\tau+1)}, z_2^{(\tau+1)}), \quad (14.44)$$

и т. д., поочередно перебирая все три переменные. Выборка Гиббса кратко представлена в алгоритме 14.3.

АЛГОРИТМ 14.3 Выборка Гиббса

Input: начальные значения $\{z_i: i \in 1, \dots, M\}$

Условные распределения $\{p(z_i | \{z_{j \neq i}\}) : i \in 1, \dots, M\}$

Количество итераций T

Output: итоговые значения $\{z_i: i \in 1, \dots, M\}$

```

for  $\tau \in \{1, \dots, T\}$  do
  for  $i \in \{1, \dots, M\}$  do
     $| z_i \sim p(z_i | \{z_{j \neq i}\})$ 
    end for
  end for
return  $\{z_i: i \in 1, \dots, M\}$ 

```

Чтобы доказать, что эта процедура берет выборку из требуемого распределения, сначала нужно отметить, что распределение $p(\mathbf{z})$ является инвариантом каждого шага выборки Гиббса в отдельности и, следовательно, всей марковской цепи. Это следует из того, что при выборке из $p(z_i | \mathbf{z}_{\setminus i})$ маргинальное распределение $p(\mathbf{z}_{\setminus i})$ явно инвариантно, поскольку значение $\mathbf{z}_{\setminus i}$ не изменяется. Кроме того, каждый шаг по определению берет выборку из правильного

условного распределения $p(z_i | z_{\setminus i})$. Поскольку эти условные и маргинальные распределения вместе определяют совместное распределение, становится очевидно, что совместное распределение само по себе инвариантно.

Второе требование, выполнение которого гарантирует получение выборки Гиббса из правильного распределения, – это эргодичность. Достаточным условием эргодичности является то, что ни одно из условных распределений нигде не равно нулю. Если это так, то любая точка в пространстве z может быть достигнута из любой другой точки за конечное число шагов, включающих одно обновление каждой из компонентных переменных. Если это требование не выполняется и в некоторых условных распределениях есть нули, то эргодичность, если она применима, должна быть доказана в явном виде.

Для завершения работы алгоритма также необходимо задать распределение начальных состояний, несмотря на то что выборки, полученные после многих итераций, фактически станут независимыми от этого распределения. Конечно, последовательные выборки из цепи Маркова будут сильно коррелировать, поэтому для получения практически независимых выборок необходимо произвести предварительную выборку последовательности.

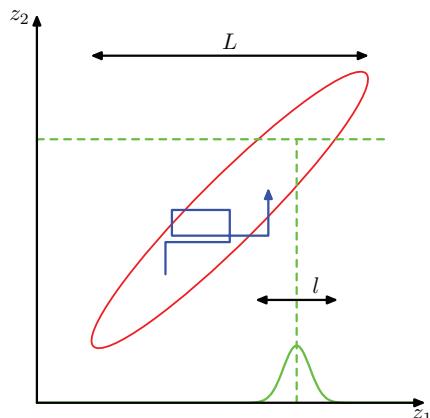
Процедуру выборки Гиббса в качестве частного случая алгоритма Метрополиса–Гастингса можно определить следующим образом. Рассмотрим шаг выборки Метрополиса–Гастингса с переменной z_k , в котором остальные переменные $z_{\setminus k}$ остаются фиксированными и для которого вероятность перехода от z к z^* определяется $q_k(z^* | z) = p(z_k^* | z_{\setminus k})$. Обратите внимание, что $z_{\setminus k}^* = z_{\setminus k}$, поскольку эти компоненты не изменяются в процессе выборки. Кроме того, $p(z) = p(z_k | z_{\setminus k})p(z_{\setminus k})$. Получается, что фактор, определяющий вероятность принятия в методе Метрополиса–Гастингса (14.40), задается как

$$A(z^*, z) = \frac{p(z^*)q_k(z|z^*)}{p(z)q_k(z^*|z)} = \frac{p(z_k^* | z_{\setminus k})p(z_{\setminus k}^*)p(z_k | z_{\setminus k})}{p(z_k | z_{\setminus k})p(z_{\setminus k})p(z_k^* | z_{\setminus k})} = 1, \quad (14.45)$$

где используется $z_{\setminus k}^* = z_{\setminus k}$. Таким образом, все шаги Метрополиса–Гастингса всегда принимаются.

Как и в случае с алгоритмом Метрополиса, некоторое представление о поведении выборки Гиббса можно получить на примере ее применения к гауссову распределению. Рассмотрим коррелированное гауссово распределение по двум переменным, как показано на рис. 14.11, с условными распределениями шириной l и маргинальными распределениями шириной L . Типичный размер шага определяется условными распределениями и составляет примерно l . Поскольку состояние изменяется по принципу случайного блуждания, количество шагов, необходимых для получения независимых выборок из распределения, будет порядка $(L/l)^2$. Конечно, если бы гауссово распределение было некоррелированным, то процедура выборки Гиббса была бы оптимально эффективной. Для этой простой задачи можно было бы повернуть систему координат таким образом, чтобы новые переменные были не коррелированы. Однако в практических задачах найти такие преобразования, как правило, не представляется возможным.

РИС. 14.11 Иллюстрация выборки Гиббса путем поочередного обновления двух переменных с коррелированным гауссовым распределением. Размер шага зависит от стандартного отклонения условного распределения (зеленая кривая) и составляет $\mathcal{O}(l)$, что приводит к медленному продвижению в направлении удлинения совместного распределения (красный эллипс). Количество шагов для получения независимой выборки из распределения равно $\mathcal{O}((L/l)^2)$



Одним из способов уменьшения поведения случайного блуждания в выборке Гиббса является метод *верхней релаксации* (*over-relaxation*) (Adler, 1981). В своей первоначальной форме он применяется к задачам, для которых условные распределения являются гауссовыми, что представляет собой более общий класс, нежели многомерные гауссовые распределения, поскольку, например, негауссово распределение $p(z, y) \propto \exp(-z^2 y^2)$ имеет гауссово условное распределение. На каждом этапе шаге алгоритма выборки Гиббса условное распределение для конкретной компоненты z_i характеризуется некоторым средним значением μ_i и некоторой дисперсией σ_i^2 . В рамках метода верхней релаксации значение z_i заменяется на

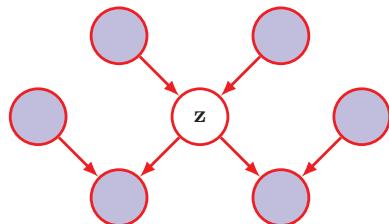
$$z'_i = \mu_i + \alpha_i(z_i - \mu_i) + \sigma_i(1 - \alpha_i^2)^{1/2}\nu, \quad (14.46)$$

где ν – это гауссова случайная величина с нулевым средним значением и единичной дисперсией, а α – это параметр по условию $-1 < \alpha < 1$. При $\alpha = 0$ метод эквивалентен стандартной выборке Гиббса, а при $\alpha < 0$ шаг смещен в противоположную сторону от среднего значения. Этот шаг оставляет желаемое распределение инвариантным, поскольку если z_i имеет среднее значение μ_i и дисперсию σ_i^2 , значит то же самое происходит и с z'_i (см. упражнение 14.14). Эффект верхней релаксации заключается в стимулировании направленного движения в пространстве состояний при сильной корреляции переменных. В рамках *упорядоченной верхней релаксации* (*ordered over-relaxation*) (Neal, 1999) этот подход обобщается применительно к негауссовым распределениям.

Практическая применимость выборки Гиббса зависит от того, насколько легко можно получить выборки из условных распределений $p(z_k | z_{\setminus k})$. Для вероятностных распределений, заданных с помощью направленных графовых моделей, условные распределения для отдельных узлов зависят только от переменных в соответствующем марковском покрытии, как показано на рис. 14.12. Для направленных графов широкий выбор условных распределений для отдельных узлов с учетом их родителей приведет к тому, что услов-

ные распределения для выборки Гиббса будут логарифмически вогнутыми. Поэтому методы адаптивной выборки с отклонением, рассмотренные в разделе 14.1.4, обеспечивают основу для выборки Монте-Карло из направленных графов с широким спектром возможностей.

РИС. 14.12 Метод выборки Гиббса требует выборки из условного распределения переменной z , зависящего от остальных переменных. Для направленных графовых моделей это условное распределение является функцией только состояний узлов марковского покрытия (заштриховано синим цветом), состоящего из родителей, детей и сородичей



Поскольку базовая техника выборки Гиббса рассматривает одну переменную за один раз, между последовательными выборками существуют сильные зависимости. В противоположном случае если бы удалось взять выборку непосредственно из совместного распределения (операция, предположительно неразрешимая), то последовательные выборки были бы независимыми. Можно надеяться на улучшение простого метода выборки Гиббса с помощью промежуточной стратегии, где выборка производится последовательно из групп переменных, а не из отдельных переменных. Это достигается в алгоритме блокирующей выборки Гиббса (*blocking Gibbs sampling*) путем выбора блоков переменных, не обязательно непересекающихся, а затем совместной выборки из переменных в каждом блоке по очереди с учетом оставшихся переменных (Jensen, Kong and Kjaerulff, 1995).

14.2.5. Выборка по предкам

Для многих моделей совместное распределение $p(\mathbf{z})$ удобно задавать в виде модели графа. Для направленного графа, не имеющего наблюдаемых переменных, выборка из совместного распределения не представляет сложности при использовании метода выборки по предкам (*ancestral sampling*). Совместное распределение задается в виде

$$p(\mathbf{z}) = \prod_{i=1}^M p(\mathbf{z}_i | \text{pa}(i)), \quad (14.47)$$

где \mathbf{z}_i – это набор переменных, связанных с узлом i , а $\text{pa}(i)$ обозначает набор переменных, связанных с родителями узла i . Чтобы получить выборку из совместного распределения, нужно сделать один проход по набору переменных в порядке $\mathbf{z}_1, \dots, \mathbf{z}_M$, делая выборку из условных распределений $p(\mathbf{z}_i | \text{pa}(i))$. Это всегда возможно, поскольку на каждом шаге все родительские значения будут уже инстанцированы. После одного прохода по графу будет получена выборка из совместного распределения. Это предполагает, что в каждом узле можно сделать выборку из отдельных условных распределений.

Теперь рассмотрим направленный граф, в котором некоторые из узлов, составляющих набор доказательств (*evidence set*) \mathcal{E} , инстанцированы наблюдаемыми значениями. В принципе, описанную выше процедуру можно расширить, по крайней мере, для узлов, представляющих дискретные переменные, чтобы создать следующий подход логической выборки (*logic sampling*) (Henrion, 1988), который можно рассматривать как частный случай выборки по важности (см. раздел 14.1.5). На каждом шаге при получении выборочного значения для переменной z_i с наблюдаемым значением выборочное значение сравнивается с наблюдаемым значением, и если они совпадают, то выборочное значение сохраняется и алгоритм переходит к следующей переменной в очереди. Если же выборочное и наблюдаемое значения различаются, то вся выборка отбрасывается, и алгоритм начинает работу с первого узла графа. Этот алгоритм корректно производит выборку из апостериорного распределения, поскольку он соответствует простому извлечению выборок из совместного распределения скрытых переменных и переменных данных, а затем отбрасывает те выборки, которые не согласуются с наблюдаемыми данными (с тем небольшим отличием, что не продолжает выборку из совместного распределения, как только наблюдается одно противоречивое значение). Однако общая вероятность принятия выборки из апостериорного распределения быстро уменьшается с увеличением числа наблюдаемых переменных и числа состояний, которые могут принимать эти переменные, поэтому такой подход редко используется на практике.

Улучшение этого подхода называется выборкой со взвешенной вероятностью (*likelihood weighted sampling*) (Fung and Chang, 1990; Shachter and Peot, 1990). Этот подход основан на выборке предков в сочетании с выборкой по важности. Для каждой переменной по очереди, если она есть в наборе доказательств, просто устанавливается ее фактическое значение. Если ее нет в наборе доказательств, то она выбирается из условного распределения $p(z_i | pa(i))$, в котором условные переменные устанавливаются на свои актуальные выборочные значения. Весовые коэффициенты, относящиеся к результатирующей выборке z (см. упражнение 14.15), определяются как

$$r(z) = \prod_{z_i \in e} \frac{p(z_i | pa(i))}{p(z_i | pa(i))} \prod_{z_i \in e} \frac{p(z_i | pa(i))}{1} = \prod_{z_i \in e} p(z_i | pa(i)). \quad (14.48)$$

Этот метод может быть дополнительно расширен с помощью выборки по собственной важности (*self-importance sampling*) (Shachter and Peot, 1990), при которой распределение выборки по важности постоянно обновляется для отражения актуального оцененного апостериорного распределения.

14.3. Выборка Ланжевена

Алгоритм Метрополиса–Гастингса берет выборки из распределения вероятностей, создавая цепь Маркова из выборок-кандидатов с помощью распределения предложений, а затем принимая или отвергая их с помощью критерия

(14.40). Этот метод может быть достаточно неэффективным, поскольку распределение предложений часто является простым, фиксированным распределением, которое может генерировать обновления в любом направлении в пространстве данных, что приводит к случайному блужданию.

Ранее уже было замечено, что при обучении нейронных сетей очень выгодно использовать градиент логарифма правдоподобия относительно обучаемых параметров модели с целью максимизации функции правдоподобия. По аналогии можно представить алгоритмы выборки цепей Маркова, которые используют градиент плотности распределения вероятности относительно вектора данных, чтобы делать шаги преимущественно в сторону областей с более высокой вероятностью. Один из таких методов называется *гамильтоновым Монте-Карло* (*Hamiltonian Monte Carlo*), также известным как *гибридный Монте-Карло* (*hybrid Monte Carlo*). В нем вновь используется приемочный тест Метрополиса (Duane et al., 1987; Bishop, 2006). Здесь основное внимание будет уделено другому подходу, широко используемому в глубоком обучении, который называется *выборкой Ланжевена* (*Langevin sampling*). Несмотря на то что он позволяет избежать использования приемочного теста, для получения несмешенных выборок алгоритм должен быть тщательно продуман. Одно из важнейших применений выборки Ланжевена связано с моделями машинного обучения, определяемыми в виде энергетических функций.

14.3.1. Модели на основе энергии

Многие генеративные модели могут быть выражены в виде условных распределений вероятности $p(\mathbf{x} | \mathbf{w})$, где \mathbf{x} – это вектор данных, а \mathbf{w} – это вектор обучаемых параметров. Такие модели можно обучать путем максимизации соответствующей функции правдоподобия, определенной относительно обучающего набора данных. Однако для представления достоверного распределения вероятностей модель должна удовлетворять следующим требованиям:

$$\int p(\mathbf{x} | \mathbf{w}) p(\mathbf{x}) d\mathbf{x} = 1. \quad (14.49)$$

Выполнение этого требования может существенно ограничить допустимые формы модели. Если отбросить ограничение на нормализацию, то можно рассмотреть гораздо более широкий класс моделей, называемых *моделями на основе энергии* (*energy-based models*) (LeCun et al., 2006). Предположим, имеется функция $E(\mathbf{x}, \mathbf{w})$, называемая *функцией энергии* (*energy function*), которая является вещественной функцией собственных аргументов, но не имеет других ограничений. Экспоненциальное выражение $\exp\{-E(\mathbf{x}, \mathbf{w})\}$ является неотрицательной величиной и поэтому может рассматриваться как ненормализованное распределение вероятностей по \mathbf{x} . Здесь введение знака минус в экспоненте является простой условностью и означает, что большие значения энергии соответствуют меньшим значениям вероятности. Затем можно определить нормализованное распределение, используя

$$p(\mathbf{x}|\mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp\{-E(\mathbf{x}, \mathbf{w})\}, \quad (14.50)$$

где нормализующая константа $Z(\mathbf{w})$ (см. упражнение 14.16), известная как *функция разбиения (partition function)*, определяется как

$$Z(\mathbf{w}) = \int \exp\{-E(\mathbf{x}, \mathbf{w})\} d\mathbf{x}. \quad (14.51)$$

Функция энергии часто моделируется с помощью глубокой нейронной сети с входным вектором \mathbf{x} и скалярным выходом $E(\mathbf{x}, \mathbf{w})$, где \mathbf{w} представляет собой веса и смещения в сети.

Обратите внимание, что функция разбиения зависит от \mathbf{w} , что создает проблемы при обучении. Например, функция логарифмического правдоподобия для набора данных $\mathcal{D} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ из i.i.d.-данных (см. упражнение 14.17) имеет вид:

$$\ln p(\mathcal{D}|\mathbf{w}) = -\sum_{n=1}^N E(\mathbf{x}_n, \mathbf{w}) - N \ln Z(\mathbf{w}). \quad (14.52)$$

Чтобы вычислить градиент $\ln p(\mathcal{D}|\mathbf{w})$ по \mathbf{w} , необходимо знать форму $Z(\mathbf{w})$. Однако для многих вариантов энергетической функции $E(\mathbf{x}, \mathbf{w})$ будет нецелесообразно оценивать функцию разбиения в (14.51), поскольку для этого необходимо интегрировать (или суммировать для дискретных переменных) по всему пространству \mathbf{x} . Термин «модель на основе энергии» обычно используется для моделей, в которых этот интеграл неразрешим. Тем не менее обратите внимание, что вероятностные модели можно рассматривать как частные случаи энергетических моделей. По этой причине многие из моделей, обсуждаемых в этой книге, можно рассматривать именно в качестве энергетических моделей. Таким образом, большим преимуществом моделей на основе энергии является их гибкость, поскольку они не требуют нормализации. Соответствующим недостатком, однако, является тот факт, что, поскольку нормализующая константа неизвестна, их обучение может быть более сложным.

14.3.2. Максимизация правдоподобия

Для обучения моделей на основе энергии без необходимости оценки функции разбиения были разработаны различные методы аппроксимации (Song and Kingma, 2021). Здесь будут рассмотрены методы на основе метода марковской цепи Монте-Карло. Альтернативный подход, называемый согласованием баллов, будет обсуждаться в контексте диффузионных моделей (см. главу 20).

Как уже говорилось, для моделей на основе энергии функция правдоподобия не может быть оценена в явном виде из-за неопределенности функции разбиения $Z(\mathbf{w})$. Однако с помощью методов выборки Монте-Карло можно аппроксимировать градиент логарифмического правдоподобия в зависимости от параметров модели. После обучения модели на основе энергии любым

способом необходимо получить выборки из модели, и здесь вновь можно воспользоваться методами Монте-Карло.

Градиент функции логарифмического правдоподобия для модели на основе энергии с учетом параметров модели при использовании (14.50) можно записать в виде

$$\nabla_w \ln p(\mathbf{x} | \mathbf{w}) = -\nabla_w E(\mathbf{x}, \mathbf{w}) - \nabla_w \ln Z(\mathbf{w}). \quad (14.53)$$

Это функция правдоподобия для одной точки данных \mathbf{x} , но на практике возникает необходимость максимизировать правдоподобие, определенное для обучающего набора точек данных из некоторого неизвестного распределения $p_D(\mathbf{x})$. Если предположить, что точки данных являются i.i.d., то можно рассмотреть градиент ожидания логарифмического правдоподобия относительно $p_D(\mathbf{x})$, которое в этом случае определяется как

$$\mathbb{E}_{\mathbf{x} \sim p_D} [\nabla_w \ln p(\mathbf{x} | \mathbf{w})] = -\mathbb{E}_{\mathbf{x} \sim p_D} [\nabla_w E(\mathbf{x}, \mathbf{w})] - \nabla_w \ln Z(\mathbf{w}), \quad (14.54)$$

где используется тот факт, что последний член $-\nabla_w \ln Z(\mathbf{w})$ не зависит от \mathbf{x} и поэтому может быть выведен за пределы математического ожидания. Предполагается, что функция разбиения $Z(\mathbf{w})$ (см. упражнение 14.18) неизвестна, но с помощью (14.51) и перестановки можно получить

$$-\nabla_w \ln Z(\mathbf{w}) = \int \{\nabla_w E(\mathbf{x}, \mathbf{w})\} p(\mathbf{x} | \mathbf{w}) d\mathbf{x}. \quad (14.55)$$

Правая часть (14.55) соответствует ожиданию по распределению модели $p(\mathbf{x} | \mathbf{w})$, которое определяется как

$$\int \{\nabla_w E(\mathbf{x}, \mathbf{w})\} p(\mathbf{x} | \mathbf{w}) d\mathbf{x} = \mathbb{E}_{\mathbf{x} \sim \mathcal{M}} [\nabla_w E(\mathbf{x}, \mathbf{w})]. \quad (14.56)$$

Комбинируя (14.54), (14.55) и (14.56) (см. упражнение 14.18), получаем:

$$\nabla_w \mathbb{E}_{\mathbf{x} \sim p_D} [\ln p(\mathbf{x} | \mathbf{w})] = -\mathbb{E}_{\mathbf{x} \sim p_D} [\nabla_w E(\mathbf{x}, \mathbf{w})] + \mathbb{E}_{\mathbf{x} \sim p_{\mathcal{M}(\mathbf{x})}} [\nabla_w E(\mathbf{x}, \mathbf{w})]. \quad (14.57)$$

Этот результат проиллюстрирован на рис. 14.13 и имеет следующую удобную интерпретацию. Целью задачи является нахождение значений параметров \mathbf{w} , которые максимизируют функцию правдоподобия, и поэтому необходимо рассмотреть небольшое изменение \mathbf{w} в направлении градиента $\nabla_w \ln p(\mathbf{x} | \mathbf{w})$. Из (14.57) видно, что ожидаемое значение этого градиента можно выразить в виде двух членов, имеющих противоположные знаки. Первый член в правой части (14.57) уменьшает $E(\mathbf{x}, \mathbf{w})$, а значит, увеличивает плотность вероятности, определяемую моделью, для точек \mathbf{x} из $p_D(\mathbf{x})$. Второй член в правой части (14.57) увеличивает значение $E(\mathbf{x}, \mathbf{w})$ и, следовательно, уменьшает плотность вероятности, определяемую моделью, для точек данных из самой модели. В областях, где плотность модели превышает плотность обучающих данных, конечным результатом будет увеличение энергии и, следовательно, уменьшение вероятности. И наоборот, в областях, где плотность обучающих данных превышает плотность модели, эффект будет заключаться в уменьшении энергии и, следовательно, в увеличении плотности вероят-

ности. Вместе эти два условия перемещают массу вероятности от областей с низкой плотностью обучающих данных к областям с высокой плотностью данных, как и требуется. Эти два условия будут равны по величине при совпадении распределения модели с распределением данных, и в этот момент градиент в левой части (14.57) будет равен нулю.

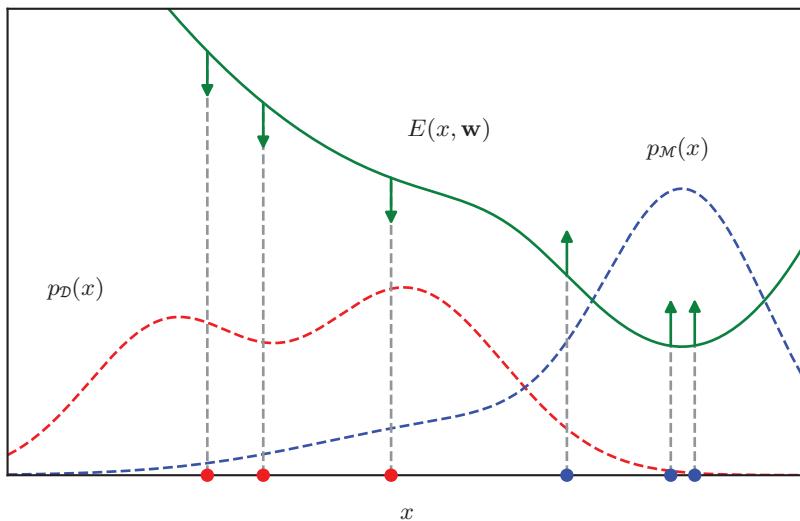


РИС. 14.13 Иллюстрация обучения модели на основе энергии путем максимизации правдоподобия. Зеленым цветом показана функция энергии $E(x, w)$ с соответствующим распределением модели $p_m(x)$ и истинным распределением данных $p_d(x)$. Увеличение ожидаемого логарифмического правдоподобия с помощью (14.57) поднимает энергетическую функцию в точках, соответствующих выборкам из модели (показаны синими точками), и опускает ее в точках, соответствующих выборкам из набора данных (показаны красными точками)

14.3.3. Динамика Ланжевена

При практическом применении (14.57) в качестве метода обучения возникает необходимость аппроксимировать два члена в правой части. Для любого заданного значения x можно оценить $\nabla_w E(x, w)$ с помощью автоматического дифференцирования. Для первого члена в (14.57) можно использовать набор обучающих данных для оценки ожидания по x :

$$\mathbb{E}_{x \sim p_m(x)} [\nabla_w E(x, w)] = \frac{1}{M} \sum_{m=1}^M \nabla_w E(\mathbf{x}_m, \mathbf{w}). \quad (14.58)$$

Второй член является более сложным, поскольку требуется взять выборки из модельного распределения, определяемого энергетической функцией, а соответствующая функция разбиения является неразрешимой. Для этого можно воспользоваться методами марковской цепи Монте-Карло. Один из популярных подходов называется *стохастической градиентной динамикой Ланжевена* (*stochastic gradient Langevin dynamics*) или просто *выборкой Ланже-*

вена (*Langevin sampling*) (Parisi, 1981; Welling and Teh, 2011). Эта функция зависит от распределения $p(\mathbf{x} | \mathbf{w})$ только через функцию отсчета (*score function*), которая определяется как градиент логарифмического правдоподобия по отношению к вектору данных \mathbf{x} и задается как

$$\mathbf{s}(\mathbf{x}, \mathbf{w}) = \nabla_{\mathbf{x}} \ln p(\mathbf{x} | \mathbf{w}). \quad (14.59)$$

Стоит подчеркнуть, что этот градиент определяется относительно точки данных \mathbf{x} и поэтому не является обычным градиентом по обучаемым параметрам \mathbf{w} . Если подставить (14.50) в (14.59), то получается

$$\mathbf{s}(\mathbf{x}, \mathbf{w}) = -\nabla_{\mathbf{x}} E(\mathbf{x}, \mathbf{w}), \quad (14.60)$$

где видно, что функция разбиения больше не появляется, поскольку она не зависит от \mathbf{x} .

Для начала возьмем начальное значение $\mathbf{x}^{(0)}$ из априорного распределения, а затем выполним следующие шаги цепи Маркова:

$$\mathbf{x}^{(\tau+1)} = \mathbf{x}^{(\tau)} + \eta \nabla_{\mathbf{x}} \ln p(\mathbf{x}^{(\tau)}, \mathbf{w}) + \sqrt{2\eta} \epsilon^{(\tau)}, \quad \tau \in 1, \dots, T, \quad (14.61)$$

где $(\tau) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ – это независимые выборки из гауссова распределения с нулевым средним значением и единичной ковариацией, а параметр η управляет размером шага. На каждой итерации уравнения Ланжевена делается шаг в направлении градиента логарифмического правдоподобия, а затем добавляется гауссов шум. Можно показать, что в пределах $\eta \rightarrow 0$ и $T \rightarrow \infty$ значение $\mathbf{z}^{(T)}$ является независимой выборкой из распределения $p(\mathbf{x})$. Выборка Ланжевена кратко описана в алгоритме 14.4.

АЛГОРИТМ 14.4 Выборка Ланжевена

Input: начальное значение $\mathbf{x}^{(0)}$

Плотность вероятности $p(\mathbf{x}, \mathbf{w})$

Параметр скорости обучения η

Количество итераций T

Output: итоговое значение $\mathbf{x}^{(T)}$

```

 $\mathbf{x} \leftarrow \mathbf{x}_0$ 
for  $\tau \in \{1, \dots, T\}$  do
|    $\epsilon \sim \mathcal{N}(\epsilon | \mathbf{0}, \mathbf{I})$ 
|    $\mathbf{x} \leftarrow \mathbf{x} + \eta \nabla_{\mathbf{x}} \ln p(\mathbf{x}, \mathbf{w}) + \sqrt{2\eta} \epsilon$ 
end for
return  $\mathbf{x}$  // Итоговое значение  $\mathbf{x}^{(T)}$ 

```

Для генерации набора выборок $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ необходимо повторить процесс и затем аппроксимировать второй член в (14.57) при помощи

$$\mathbb{E}_{\mathbf{x} \sim p_{\mathcal{M}}(\mathbf{x})} [\nabla_{\mathbf{w}} E(\mathbf{x}, \mathbf{w})] \simeq \frac{1}{M} \sum_{m=1}^M \nabla_{\mathbf{w}} E(\mathbf{x}_m, \mathbf{w}). \quad (14.62)$$

Запуск длинных цепей Маркова для получения независимых выборок может быть вычислительно затратным, поэтому приходится рассматривать практические приближения. Один из подходов называется *контрастной дивергенцией* (*contrastive divergence*) (Hinton, 2002). В этом случае выборки, используемые для оценки (14.62), получаются путем запуска цепочки Монте-Карло, начинающейся с одной из обучающих точек данных \mathbf{x}_n . Если цепочка выполняется за большое количество шагов, то полученное значение будет, по сути, несмещенной выборкой из распределения модели. Вместо этого в работе (Hinton, 2002) было предложено выполнять всего несколько шагов Монте-Карло, возможно, даже всего один шаг, что гораздо менее затратно в плане вычислительных ресурсов. Полученная выборка будет далека от несмещенной и будет лежать близко к многообразию данных. В результате эффект от использования градиентного спуска будет заключаться в формировании энергетической поверхности, и, следовательно, плотности вероятности только в окрестностях многообразия данных. Такой подход может оказаться эффективным для таких задач, как дискриминация, но, как можно ожидать, будет менее эффективным при обучении генеративной модели.

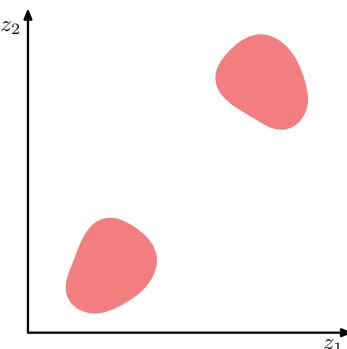
Упражнения

- 14.1** (*) Докажите, что \bar{f} , определенная в (14.2), является несмещенной оценкой, или, другими словами, что ожидание правой части равно $\mathbb{E}[f(\mathbf{z})]$.
- 14.2** (*) Докажите, что \bar{f} , определенная в (14.2), имеет дисперсию, заданную в (14.4).
- 14.3** (*) Предположим, что z – это случайная величина с равномерным распределением по $(0, 1)$ и что z преобразуется через $y = h^{-1}(z)$, где $h(y)$ задается в (14.6). Докажите, что y имеет распределение $p(y)$.
- 14.4** (**) Для случайной величины z , равномерно распределенной по $(0, 1)$, найдите преобразование $y = f(z)$, такое, что y имеет распределение Коши, заданное в (14.8).
- 14.5** (**) Предположим, что z_1 и z_2 равномерно распределены по единичной окружности, как показано на рис. 14.3, и произведем замену переменных в соответствии с (14.10) и (14.11). Докажите, что (y_1, y_2) будут распределены согласно (14.12).
- 14.6** (**) Пусть \mathbf{z} – это D -мерная случайная величина, имеющая гауссово распределение с нулевым средним и единичной ковариационной матрицей, и предположим, что положительно определенная симметричная матрица Σ имеет разложение Холецкого $\Sigma = \mathbf{L}\mathbf{L}^T$, где \mathbf{L} – это нижняя треугольная матрица (т. е. матрица с нулями над главной диагональю). Докажите, что переменная $\mathbf{y} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z}$ имеет гауссово распределение со средним значением $\boldsymbol{\mu}$ и ковариацией Σ . Это дает возможность по-

лучить выборки из общего многомерного гауссова распределения по выборкам из одномерного гауссова распределения с нулевым средним и единичной дисперсией.

- 14.7** (**) В этом упражнении более тщательно демонстрируется тот факт, что выборка с отклонением действительно берет выборки из желаемого распределения $p(\mathbf{z})$. Предположим, что вспомогательное распределение – это $q(\mathbf{z})$. Докажите, что вероятность принятия значения \mathbf{z} из выборки равна $\tilde{p}(\mathbf{z})/kq(\mathbf{z})$, где \tilde{p} – это любое ненормализованное распределение, пропорциональное $p(\mathbf{z})$, а константа k – это наименьшее значение, обеспечивающее $kq(\mathbf{z}) \geq \tilde{p}(\mathbf{z})$ для всех значений \mathbf{z} . Обратите внимание, что вероятность принятия значения \mathbf{z} равна вероятности принятия этого значения из $q(\mathbf{z})$, умноженной на вероятность принятия этого значения, если оно уже было принято. Используя это, а также правила суммы и произведения вероятностей, запишите нормализованную форму распределения по \mathbf{z} и докажите, что она равна $p(\mathbf{z})$.
- 14.8** (*) Предположим, что z имеет равномерное распределение в интервале $[0, 1]$. Докажите, что переменная $y = b \tan z + c$ имеет распределение Коши, заданное в (14.16).
- 14.9** (**) Определите выражения для коэффициентов k_i в огибающей распределения (14.17) для аддитивной выборки с отклонением, используя требования непрерывности и нормализации.
- 14.10** (**) Используя методику, рассмотренную в разделе 14.1.2 для выборки из единичного экспоненциального распределения, разработайте алгоритм выборки из кусочно-экспоненциального распределения, заданного в (14.17).
- 14.11** (*) Докажите, что простое случайное блуждание по целым числам, определяемое в (14.28), (14.29) и (14.30), имеет свойство $\mathbb{E}[(z^{(r)})^2] = \mathbb{E}[(z^{(r-1)})^2] + 1/2$ и, следовательно, по индукции, $\mathbb{E}[(z^{(r)})^2] = r/2$.
- 14.12** (**) Докажите, что алгоритм выборки Гиббса, рассмотренный в разделе 14.2.4, удовлетворяет детальному балансу, определенному в (14.34).
- 14.13** (*) Рассмотрим распределение, показанное на рис. 14.14. Определите, является ли стандартная процедура выборки Гиббса для этого распределения эргодической и, следовательно, будет ли она корректно производить выборку из этого распределения.

РИС. 14.14 Распределение вероятностей для двух переменных z_1 и z_2 , равномерное в заштрихованных областях и нулевое во всех остальных



- 14.14** (*) Подтвердите, что обновление с верхней релаксацией (14.46), в котором z_i имеет среднее значение μ_i и дисперсию σ_i и где v имеет нулевое среднее значение и единичную дисперсию, дает значение z'_i со средним значением μ_i и дисперсией σ_i^2 .
- 14.15** (*) Докажите, что при взвешенной по вероятности выборке из направленного графа значения весов выборки важности определяются в (14.48).
- 14.16** (*) Докажите, что распределение (14.50) нормализовано относительно x при условии, что $Z(w)$ удовлетворяет (14.51).
- 14.17** (**) Используя (14.50), докажите, что градиент функции логарифмического правдоподобия для модели на основе энергии может быть записан в виде (14.52).
- 14.18** (**) Используя (14.54), (14.55) и (14.56), докажите, что градиент функции логарифмического правдоподобия для модели на основе энергии может быть записан в виде (14.57).

Глава 15

Дискретные латентные переменные

Ранее уже рассматривался вопрос о возможности построения сложных распределений путем объединения нескольких простых распределений и описания полученных моделей с помощью направленных графов (см. главу 11). В дополнение к наблюдаемым переменным, которые являются частью набора данных, в таких моделях часто вводятся дополнительные скрытые, или латентные, переменные. Они могут соответствовать определенным величинам, участвующим в процессе получения данных, например неизвестной ориентации объекта в трехмерном пространстве в случае с изображениями, или могут быть введены просто в качестве моделирующих элементов для построения гораздо более содержательных моделей. Если задать совместное распределение по наблюдаемым и латентным переменным, то соответствующее распределение только наблюдаемых переменных можно получить путем маргинализации. Это дает возможность выразить относительно сложные маргинальные распределения по наблюдаемым переменным в виде более простых совместных распределений по расширенному пространству наблюдаемых и латентных переменных.

В этой главе рассматривается метод маргинализации дискретных латентных переменных, что приводит к появлению смешанных распределений. Основное внимание будет уделено изучению композиций (смесей) гауссовых распределений, которые отлично иллюстрируют смешанные распределения и также широко используются в машинном обучении. Одним из наиболее простых применений моделей смещивания является определение кластеров в данных, и поэтому для начала речь пойдет о методе кластеризации, называемом алгоритмом K -средних (K -means), который соответствует особому не вероятностному пределу смесей гауссовых (нормальных) распределений. Затем будет рассмотрено латентно-переменное представление смесей распределений, где дискретные латентные переменные могут быть интерпретированы как определяющие отнесение точек данных к конкретным компонентам смеси.

Общей методикой поиска оценок максимального правдоподобия в моделях с латентными переменными является алгоритм ожидания-максимизации (expectation–maximization, EM). Сначала на примере распределения

смеси гауссовых распределений рассматривается неформальная трактовка алгоритма EM, а затем он рассматривается более подробно с учетом латентных переменных. Наконец, для общего представления представим нижнюю границу доказательств (evidence lower bound, ELBO), которая будет играть важную роль в генеративных моделях, таких как вариационные автокодировщики и диффузионные модели.

15.1. Кластеризация K-средних

Начнем с решения задачи определения групп, или кластеров, из точек данных в многомерном пространстве. Предположим, имеется набор данных $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, состоящий из N наблюдений за D -мерной евклидовой переменной \mathbf{x} . Задача состоит в том, чтобы разбить набор данных на некоторое количество K кластеров, при этом пока будем считать, что значение K определено заранее. Кластер можно представить в виде группы точек данных, где расстояния между точками малы по сравнению с расстояниями до точек вне кластера. Формализовать это понятие можно путем введения набора D -мерных векторов $\boldsymbol{\mu}_k$, где $k = 1, \dots, K$, а $\boldsymbol{\mu}_k$ – это «прототип», связанный с k -м кластером. Как будет показано далее, можно считать, что $\boldsymbol{\mu}_k$ представляют собой центры кластеров. Тогда задача состоит в определении набора кластерных векторов $\{\boldsymbol{\mu}_k\}$ и распределении точек данных по кластерам таким образом, чтобы сумма квадратов расстояний каждой точки данных до ближайшего к ней кластерного вектора $\boldsymbol{\mu}_k$ была минимальной.

На этом этапе целесообразно ввести некоторые обозначения для описания распределения точек данных по кластерам. Для каждой точки данных \mathbf{x}_n введем соответствующий набор бинарных индикаторных переменных $r_{nk} \in \{0, 1\}$, где $k = 1, \dots, K$. Эти индикаторы определяют принадлежность точки данных \mathbf{x}_n к одному из K кластеров, так что если точка данных \mathbf{x}_n относится к кластеру k , то $r_{nk} = 1$, а $r_{nj} = 0$ для $j \neq k$. Это пример схемы кодирования «1 из K ». Затем определим функцию ошибки:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2, \quad (15.1)$$

которая представляет собой сумму квадратов расстояний каждой точки данных до назначенного ей вектора $\boldsymbol{\mu}_k$. Целью задачи является нахождение таких значений $\{r_{nk}\}$ и $\{\boldsymbol{\mu}_k\}$, чтобы минимизировать J . Это можно сделать с помощью итерационной процедуры, где каждая итерация включает в себя два последовательных шага, соответствующих последовательным оптимизациям в отношении $\{r_{nk}\}$ и $\{\boldsymbol{\mu}_k\}$. Сначала выберем некоторые начальные значения для $\{\boldsymbol{\mu}_k\}$. Затем на первом шаге минимизируем J относительно $\{r_{nk}\}$, сохраняя $\{\boldsymbol{\mu}_k\}$ фиксированным. На втором шаге минимизируем J относительно $\{\boldsymbol{\mu}_k\}$, сохраняя $\{r_{nk}\}$ фиксированным. Эта двухэтапная оптимизация повторяется до сходимости. В дальнейшем будет показано, что эти два этапа обновления $\{r_{nk}\}$

и $\{\mu_k\}$ согласуются, соответственно, с шагами E (expectation, ожидание) и M (maximization, максимизация) алгоритма EM (см. раздел 15.3), и чтобы это подчеркнуть, в дальнейшем термины E step (шаг E) и M step (шаг M) будут использоваться в контексте алгоритма K-средних.

Сначала рассмотрим определение $\{r_{nk}\}$ при фиксированном $\{\mu_k\}$ (шаг E). Поскольку J в (15.1) является линейной функцией $\{r_{nk}\}$, эту оптимизацию можно выполнить без проблем и получить решение в замкнутой форме. Члены, включающие различные n , являются независимыми, и поэтому для каждого n можно провести отдельную оптимизацию, выбрав r_{nk} равным 1 для того значения k , которое дает минимальное значение $\|\mathbf{x}_n - \mu_k\|^2$. Другими словами, нужно просто назначить n -ю точку данных в ближайший центр кластера. Более формально это можно выразить как

$$r_{nk} = \begin{cases} 1, & \text{если } k = \operatorname{argmin}_j \|\mathbf{x}_n - \mu_j\|^2, \\ 0 & \text{в ином случае.} \end{cases} \quad (15.2)$$

Теперь рассмотрим оптимизацию $\{\mu_k\}$ при фиксированном $\{r_{nk}\}$ (шаг M). Целевая функция J является квадратичной функцией μ_k , и ее можно минимизировать, обратив ее производную по μ_k в ноль, что дает

$$2 \sum_{n=1}^N r_{nk} (\mathbf{x}_n - \mu_k) = 0. \quad (15.3)$$

Это легко решается для μ_k и дает

$$\mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}. \quad (15.4)$$

Знаменатель в этом выражении равен количеству точек, отнесенных к кластеру k , и поэтому этот результат имеет простую интерпретацию, а именно: μ_k равен среднему значению всех точек данных \mathbf{x}_n , отнесенных к кластеру k . Поэтому такая процедура известна как *алгоритм K-средних (K-means algorithm)* (Lloyd, 1982). В обобщенном виде она представлена в алгоритме 15.1. Поскольку значения $\{r_{nk}\}$ дискретны и каждая итерация не приводит к увеличению функции ошибки, алгоритм K-средних гарантированно сходится за конечное число шагов (см. упражнение 15.1).

АЛГОРИТМ 15.1 Алгоритм K-средних

Input: начальные векторы прототипов μ_1, \dots, μ_k

Набор данных $\mathbf{x}_1, \dots, \mathbf{x}_N$

Output: итоговые векторы прототипов μ_1, \dots, μ_k

$\{r_{nk} \leftarrow 0\} //$ Изначально все задания устанавливаются в ноль

repeat

$\{r_{nk}^{(\text{old})}\} \leftarrow \{r_{nk}\}$

```

// Обновление заданий
for  $N \in \{1, \dots, N\}$  do
     $k \leftarrow \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2$ 
     $r_{nk} \leftarrow 1$ 
     $r_{nj} \leftarrow 0, j \in \{1, \dots, K\}, j \neq k$ 
end for
// Обновление векторов прототипов
for  $k \in \{1, \dots, K\}$  do
     $\boldsymbol{\mu}_k \leftarrow \sum_n r_{nk} \mathbf{x}_n / \sum_n r_{nk}$ 
end for
until  $\{r_{nk}\} = \{r_{nk}^{(\text{old})}\}$  // Задания не изменены
return  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \{r_{nk}\}$ 

```

Эти два этапа, включающие переназначение точек данных в кластеры и повторное вычисление кластерных средних значений, повторяются по-очередно до тех пор, пока не произойдет дальнейших изменений в заданиях (или пока не будет превышено некоторое максимальное число итераций). Однако такой подход может привести к локальному, а не глобальному минимуму J . Свойства сходимости алгоритма K -средних были изучены в работе (MacQueen, 1967).

Алгоритм K -средних представлен на рис. 15.1 на примере данных, полученных в результате извержений гейзера Олд-Фейтфул в Йеллоустонском национальном парке, США (см. раздел 3.2.9). Набор данных состоит из 272 точек данных, каждая из которых отражает продолжительность извержения по горизонтальной оси и время до следующего извержения по вертикальной оси. Здесь проведено линейное перемасштабирование данных, известное как *стандартизация* (*standardizing*), так что каждая из переменных характеризуется нулевым средним значением и единичным стандартным отклонением. Здесь (a) зеленые точки обозначают набор данных в двумерном евклидовом пространстве. Начальные значения центров $\boldsymbol{\mu}_1$ и $\boldsymbol{\mu}_2$ показаны красным и синим крестиками соответственно. (b) На начальном шаге Е каждая точка данных относится либо к красному, либо к синему кластеру в зависимости от того, к какому центру кластера она ближе. Это эквивалентно классификации точек в зависимости от того, по какую сторону перпендикулярной биссектрисы двух центров кластеров, показанной пурпурной линией, они лежат. (c) На последующем шаге М каждый кластерный центр пересчитывается как среднее значение точек, отнесенных к соответствующему кластеру. На фрагментах (d)–(i) показаны последовательные шаги Е и М до окончательной сходимости алгоритма.

В этом примере выбрано $K = 2$, поэтому отнесение каждой точки данных к ближайшему центру кластера эквивалентно классификации точек данных в зависимости от того, по какую сторону от перпендикулярной биссектрисы двух центров кластера они лежат. График функции стоимости J , заданной в (15.1) для примера Олд-Фейтфул, показан на рис. 15.2. Обратите внимание,

что для центров кластеров специально выбраны неудачные начальные значения, так что алгоритму требуется выполнить несколько шагов до сходимости. На практике более эффективной процедурой инициализации является выбор центров кластеров μ_k для случайного подмножества из K точек данных. Также следует отметить, что алгоритм K -средних зачастую используется для инициализации параметров в модели гауссовой смеси перед началом работы алгоритма EM (см. раздел 15.2.2).

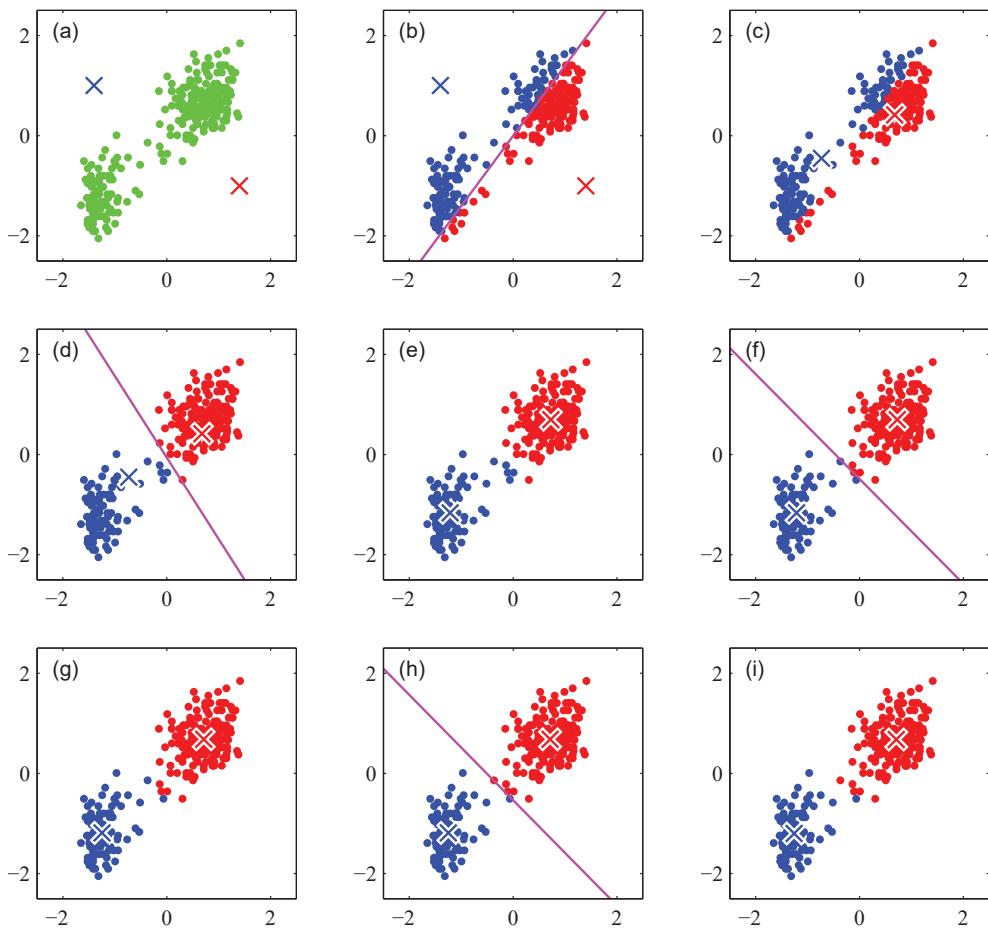


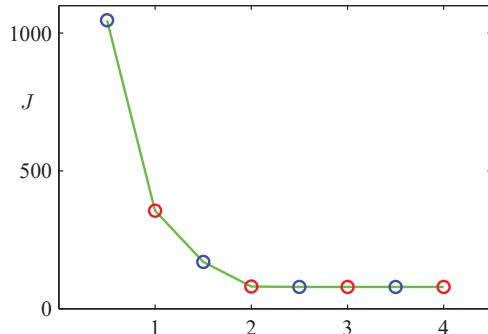
РИС. 15.1 Иллюстрация алгоритма K -средних с использованием измененного масштаба набора данных гейзера Олд-Фейтфул

До сих пор рассматривалась пакетная версия алгоритма K -средних, где для обновления векторов прототипов используется весь набор данных. Также можно вывести последовательное обновление, при котором для каждой точки данных x_n по очереди обновляется ближайший прототип μ_k (см. упражнение 15.2), при этом

$$\boldsymbol{\mu}_k^{\text{new}} = \boldsymbol{\mu}_k^{\text{old}} + \frac{1}{N_k} (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{old}}), \quad (15.5)$$

где N_k – это количество точек данных, которые до этого времени использовались для обновления $\boldsymbol{\mu}_k$. Это позволяет использовать каждую точку данных один раз, а затем отбрасывать ее до появления следующей точки данных.

РИС. 15.2 График функции стоимости J , заданной в (15.1), после каждого шага E (синие точки) и шага M (красные точки) алгоритма K -средних для примера на рис. 15.1. Алгоритм сходится после третьего шага M , и последний цикл EM не приводит к изменениям ни в назначениях, ни в векторах-прототипах



Примечательной особенностью алгоритма K -средних является то, что на каждой итерации каждая точка данных относится к одному и только одному кластеру. Хотя некоторые точки данных будут гораздо ближе к определенному центру $\boldsymbol{\mu}_k$, чем к любому другому центру, могут быть и другие точки данных, которые лежат примерно посередине между центрами кластеров. В последнем случае не совсем очевидно, что жесткое отнесение к ближайшему кластеру является наиболее подходящим (см. раздел 15.2). В дальнейшем будет показано, что при использовании вероятностного подхода можно получить «мягкое» распределение точек данных по кластерам, отражающее уровень неопределенности относительно наиболее подходящего распределения. Такая вероятностная формулировка обеспечивает множество преимуществ.

15.1.1. Сегментация изображений

В качестве иллюстрации возможностей применения алгоритма K -средних рассмотрим близкие задачи сегментации и сжатия изображений. Целью сегментации является разделение изображения на области так, чтобы каждая из них имела достаточно однородное визуальное отображение, соответствовала объектам или частям объектов (Forsyth and Ponce, 2003). Каждый пиксель в изображении – это точка в трехмерном пространстве, состоящем из значений яркости красного, синего и зеленого каналов, и алгоритм сегментации просто рассматривает каждый пиксель в изображении как отдельную точку данных. Обратите внимание, что, строго говоря, это пространство не является евклидовым, поскольку значения уровней яркости каналов ограничены интервалом $[0, 1]$. Тем не менее здесь алгоритм K -средних вполне применим.

Результат работы K -средних до сходимости для любого конкретного значения K можно проиллюстрировать перерисовкой изображения, где каждый вектор пикселя заменяется на тройку значений яркости $\{R, G, B\}$, заданную центром μ_k , к которому был отнесен пиксель. Результаты для различных значений K показаны на рис. 15.3. При заданном значении K алгоритм отображает изображение при помощи палитры, состоящей только из K цветов. Это также иллюстрирует использование векторного квантования для сжатия данных, при котором меньшие значения K дают более высокую степень сжатия за счет более низкого качества изображения. Следует подчеркнуть, что такое использование K -средних не является особенно сложным подходом к сегментации изображений, не в последнюю очередь потому, что в нем не учитывается пространственная близость различных пикселей. Сегментация изображений в целом чрезвычайно сложна, остается предметом активных исследований и представлена здесь лишь для иллюстрации возможностей алгоритма K -средних.



РИС. 15.3 Пример применения алгоритма кластеризации K -средних для сегментации изображений, где показано исходное изображение и их сегменты K -средних, полученные при различных значениях K

Кроме того, алгоритм кластеризации может также применяться для сжатия данных. Важно различать *сжатие данных без потерь* (*lossless data compression*), при котором целью является точное восстановление исходных данных из сжатого представления, и *сжатие данных с потерями* (*lossy data compression*), при котором допускаются некоторые ошибки при восстановлении в обмен на более высокую степень сжатия, чем в случае сжатия без потерь. Алгоритм K -средних можно применить к задаче сжатия данных с потерями следующим образом. Для каждой из N точек данных хранится только идентификатор k кластера, к которому она отнесена. Кроме того, сохраняются значения K кластерных центров $\{\mu_k\}$, что обычно требует значительно меньше объема данных при условии выбора $K \ll N$. Затем каждая точка данных аппроксимируется ближайшим к ней центром μ_k . Новые точки данных можно аналогично сжать, сначала найдя ближайший μ_k и затем сохранив метку k вместо исходного вектора данных. Такая схема часто называется *векторным*

квантованием (vector quantization), а векторы $\{\boldsymbol{\mu}_k\}$ – *векторами кодовой книги* (*codebook vectors*).

Рассмотренная выше задача сегментации изображения также является иллюстрацией использования кластеризации для сжатия данных. Предположим, исходное изображение имеет N пикселей, состоящих из значений $\{R, G, B\}$, каждое из которых записывается с точностью до 8 бит. Прямая передача всего изображения потребует $24N$ бит. Теперь предположим, что сначала выполняется анализ данных изображения методом K -средних, а затем вместо передачи исходных векторов яркости пикселей передается идентификатор ближайшего вектора $\boldsymbol{\mu}_k$. Поскольку таких векторов K , это потребует $\log_2 K$ бит на пиксель. Кроме того, необходимо передать K векторов кодовой книги $\{\boldsymbol{\mu}_k\}$, что потребует $24K$ бит, поэтому общее количество бит, необходимых для передачи изображения, равно $24K + N \log_2 K$ (округление до ближайшего целого числа). Исходное изображение, показанное на рис. 15.3, имеет размер $240 \times 180 = 43\,200$ пикселей, поэтому для его прямой передачи потребуется $24 \times 43\,200 = 1\,036\,800$ бит. Для сравнения: скатые изображения потребуют для передачи 43 248 бит ($K = 2$), 86 472 бита ($K = 3$) и 173 040 бит ($K = 10$) соответственно. Коэффициенты сжатия по сравнению с исходным изображением составляют 4,2, 8,3 и 16,7 % соответственно. Здесь наблюдается компромисс между степенью сжатия и качеством изображения. Обратите внимание, что целью этого примера является иллюстрация алгоритма K -средних. Если бы задачей было создание хорошего механизма сжатия изображений, то более целесообразно было бы рассматривать небольшие блоки соседних пикселей, например 5×5 , и, таким образом, использовать корреляции между близлежащими пикселями, характерные для естественных изображений.

15.2. Гауссовые смеси

Ранее модель гауссовой смеси была представлена как простая линейная суперпозиция гауссовых компонент, призванная обеспечить более обширный класс моделей плотности, нежели модель с одним гауссовым распределением (см. раздел 3.2.9). Теперь рассмотрим определение гауссовых смесей в контексте дискретных латентных переменных. Это позволит более глубоко проанализировать это важное распределение, а также послужит объяснением алгоритма максимизации ожиданий.

Вспомним из (3.111), что распределение гауссовой смеси можно записать в виде линейной суперпозиции гауссовых распределений в форме

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \quad (15.6)$$

Введем K -мерную двоичную случайную величину \mathbf{z} , имеющую представление «1 из K », где один из элементов равен 1, а все остальные элементы

равны 0. Получается, что значения z_k удовлетворяют $z_k \in \{0, 1\}$ и $\sum_k z_k = 1$, и получается, что для вектора \mathbf{z} существует K возможных состояний, в зависимости от того, какой из элементов является ненулевым. Определим совместное распределение $p(\mathbf{x}, \mathbf{z})$ в виде маргинального распределения $p(\mathbf{z})$ и условного распределения $p(\mathbf{x} | \mathbf{z})$. Маргинальное распределение по \mathbf{z} задается в виде коэффициентов смещивания π_k таких, что

$$p(z_k = 1) = \pi_k,$$

где параметры $\{\pi_k\}$ должны удовлетворять

$$0 \leq \pi_k \leq 1 \quad (15.7)$$

вместе с

$$\sum_{k=1}^K \pi_k = 1, \quad (15.8)$$

если они должны являться действительными вероятностями. Поскольку \mathbf{z} использует представление «1 из K », это распределение можно записать в виде

$$p(\mathbf{z}) = \sum_{k=1}^K \pi_k^{z_k}. \quad (15.9)$$

Точно так же условное распределение \mathbf{x} при определенном значении \mathbf{z} является гауссовым:

$$p(\mathbf{x} | z_k = 1) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

и его также можно записать в виде

$$p(\mathbf{x} | \mathbf{z}) = \sum_{k=1}^K \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \quad (15.10)$$

Совместное распределение задается $p(\mathbf{z})p(\mathbf{x} | \mathbf{z})$ и описывается графовой моделью на рис. 15.4. Маргинальное распределение \mathbf{x} получается суммированием совместного распределения (см. упражнение 15.3) по всем возможным состояниям \mathbf{z} , что дает

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x} | \mathbf{z}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad (15.11)$$

где использованы (15.9) и (15.10). Таким образом, маргинальное распределение \mathbf{x} – это гауссова смесь вида (15.6). При наличии нескольких наблюдений $\mathbf{x}_1, \dots, \mathbf{x}_N$ с учетом представленного маргинального распределения в виде $p(\mathbf{x}) = \sum_z p(\mathbf{x}, \mathbf{z})$ следует, что для каждой наблюдаемой точки данных \mathbf{x}_n существует соответствующая латентная переменная \mathbf{z}_n .

РИС. 15.4 Представление модели смеси в виде графа, где совместное распределение выражено в форме $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$



Итак, это – эквивалентная формулировка гауссовой смеси с латентными переменными в явной форме. Может показаться, что ничего особенного это не принесло. Однако теперь вместо маргинального распределения $p(\mathbf{x})$ можно работать с совместным распределением $p(\mathbf{x}, \mathbf{z})$, а это позволит значительно упростить задачу, в первую очередь за счет применения алгоритма EM.

Еще одна важная величина – это условная вероятность \mathbf{z} по \mathbf{x} . Для обозначения $p(z_k = 1 | \mathbf{x})$ будет использоваться $\gamma(z_k)$, значение которой может быть найдено с помощью теоремы Байеса:

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1 | \mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(\mathbf{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.\end{aligned}\quad (15.12)$$

В дальнейшем π_k будет рассматриваться в качестве априорной вероятности $z_k = 1$, а $\gamma(z_k)$ – в качестве соответствующей апостериорной вероятности после наблюдения \mathbf{x} . Как будет показано далее, $\gamma(z_k)$ можно также рассматривать как «ответственность», которую компонент k берет на себя за «объяснение» наблюдения \mathbf{x} .

Для получения случайных выборок, распределенных в соответствии с моделью гауссовой смеси, можно использовать выборку предков (см. раздел 14.2.5). Для этого сначала из маргинального распределения $p(\mathbf{z})$ генерируется значение для \mathbf{z} , которое обозначим $\hat{\mathbf{z}}$, а затем из условного распределения $p(\mathbf{x}|\hat{\mathbf{z}})$ генерируется значение для \mathbf{x} . Выборки из совместного распределения $p(\mathbf{x}, \mathbf{z})$ можно изобразить путем нанесения точек на соответствующие значения \mathbf{x} и раскрашивания их в зависимости от значения \mathbf{z} , т. е. в зависимости от принадлежности к той или иной гауссовой компоненте, как показано на рис. 15.5а. Аналогичным образом выборки из маргинального распределения $p(\mathbf{x})$ получаются путем взятия выборок из совместного распределения и игнорирования значений \mathbf{z} . Они показаны на рис. 15.5б путем нанесения значений \mathbf{x} без каких-либо цветных меток. На этом рисунке: (а) выборки из совместного распределения $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, в которых три состояния \mathbf{z} , соответствующие трем компонентам смеси, изображены красным, зеленым и синим цветом, и (б) соответствующие выборки из маргинального распределения $p(\mathbf{x})$, которое получается путем простого игнорирования значений \mathbf{z} и простого построения значений \mathbf{x} . Набор данных в (а) считается полным, а в (б) – неполным. (с) Те же выбор-

ки, в которых цвета представляют значение обязанностей $\gamma(z_{nk})$, связанное с точкой данных \mathbf{x}_n и полученное при построении соответствующей точки с использованием пропорций красных, синих и зеленых цветов, заданных $\gamma(z_{nk})$ для $k = 1, 2, 3$ соответственно.

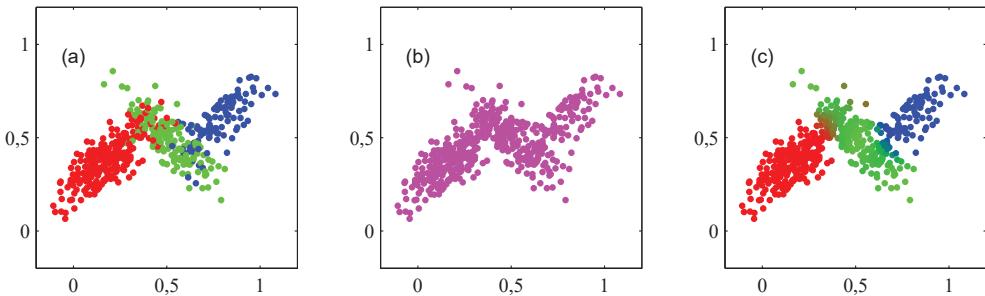


РИС. 15.5 Пример 500 точек, взятых из смеси трех гауссовых распределений из рис. 3.8

Этот синтетический набор данных можно также использовать для иллюстрации «ответственности», оценивая для каждой точки данных апостериорную вероятность для каждого компонента в распределении смеси, на основе которого был сгенерирован этот набор данных. В частности, значение ответственности $\gamma(z_{nk})$, связанное с точкой данных \mathbf{x}_n , можно представить в виде графика соответствующей точки с использованием пропорций красных, синих и зеленых цветов, заданных $\gamma(z_{nk})$ для $k = 1, 2, 3$ соответственно, как показано на рис. 15.5с. Так, например, точка данных, для которой $\gamma(z_{n1}) = 1$, будет окрашена в красный цвет, в то время как точка, для которой $\gamma(z_{n2}) = \gamma(z_{n3}) = 0,5$, будет окрашена равными пропорциями синих и зеленых цветов и поэтому будет выглядеть голубой. Это можно сравнить с рис. 15.5а, где точки данных были помечены с использованием истинной идентичности компонента, из которого они были получены.

15.2.1. Функция правдоподобия

Предположим, что имеется набор данных $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ и необходимо смоделировать эти данные с помощью смеси гауссовых распределений. Этот набор данных можно представить в виде $N \times D$ матрицы \mathbf{X} , в которой n -я строка задана \mathbf{x}_n^T . Из (15.6) следует, что логарифм функции правдоподобия имеет вид:

$$\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}. \quad (15.13)$$

Максимизация этой функции логарифмического правдоподобия (15.13) представляет собой более сложную задачу, чем для одного гауссова распределения. Сложность возникает из-за наличия суммирования по k , которое появляется внутри логарифма в (15.13), так что функция логарифма больше не действует непосредственно на гауссово распределение. Если производные

от логарифма вероятности обращается в ноль, замкнутого решения уже не получится, как будет показано далее.

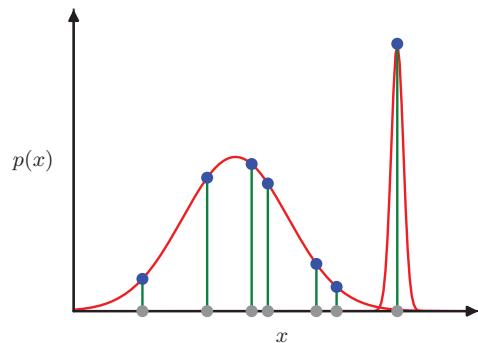
Прежде чем перейти к обсуждению вопроса максимизации этой функции, следует подчеркнуть, что при использовании модели гауссовой смеси в рамках принципа максимального правдоподобия возникает серьезная проблема, связанная с наличием сингулярностей. Для простоты рассмотрим гауссову смесь с компонентами, ковариационные матрицы которых имеют вид $\Sigma_k = \sigma_k^2 \mathbf{I}$, где \mathbf{I} – это единичная матрица, хотя выводы будут справедливы и для общих ковариационных матриц. Предположим, что один из компонентов модели смеси, скажем j -й компонент, имеет среднее значение μ_j , точно равное одной из точек данных, так что $\mu_j = \mathbf{x}_n$ для некоторого значения n . Тогда эта точка данных будет вносить в функцию правдоподобия компонент вида

$$\mathcal{N}(\mathbf{x}_n | \mathbf{x}_n, \sigma_j^2 \mathbf{I}) = \frac{1}{(2\pi)^{1/2}} \frac{1}{\sigma_j}. \quad (15.14)$$

Если рассматривать предел $\sigma_j \rightarrow 0$, то можно увидеть, что этот компонент стремится к бесконечности, а значит и функция логарифмического правдоподобия также стремится к бесконечности. Таким образом, максимизация функции логарифмического правдоподобия не является хорошо поставленной задачей, поскольку такие сингулярности будут присутствовать всегда и будут возникать всякий раз, когда одна из гауссовых компонент «обрушивается» на определенную точку данных. Вспомните, что эта проблема не возникала при единственном гауссовом распределении. Чтобы понять разницу, отметим, что если одиночное гауссово распределение попадает на точку данных, то оно вносит мультипликативные коэффициенты в функцию правдоподобия, вытекающую из других точек данных, и эти коэффициенты экспоненциально быстро стремятся к нулю, давая общее правдоподобие, которое стремится к нулю, а не к бесконечности. Однако если в смеси есть (по крайней мере) два компонента, один из них может иметь конечную дисперсию и, следовательно, задавать конечную вероятность для всех точек данных, в то время как другой компонент может сжиматься до одной конкретной точки данных и вносить все большее аддитивное значение в логарифмическое правдоподобие. Это показано на рис. 15.6. Эти сингулярности служат примером чрезмерной подгонки, которая может произойти при использовании метода максимального правдоподобия. Применяя максимальное правдоподобие к моделям гауссовых смесей, следует избегать таких нежелательных решений и вместо этого искать локальные максимумы функции правдоподобия с благоприятным поведением. Избежать сингулярностей можно с помощью подходящих эвристик, например определяя моменты обрушения гауссовой компоненты и сбрасывая ее среднее значение до случайно выбранного значения, а также сбрасывая ее ковариацию до некоторого большого значения, после чего можно продолжать оптимизацию. Сингулярностей также можно избежать за счет добавления регуляризующего члена

к логарифму правдоподобия, соответствующего априорному распределению по параметрам (см. раздел 15.4.3).

РИС. 15.6 Пример появления сингулярностей в функции правдоподобия при использовании смеси гауссовых распределений. Это можно сравнить с одиночным гауссовым распределением на рис. 2.9, где сингулярности не возникают



Еще одна проблема при поиске решений максимального правдоподобия возникает в связи с тем, что для любого заданного решения максимального правдоподобия у K -компонентной смеси будет в общей сложности $K!$ эквивалентных решений, соответствующих $K!$ способам присвоения K наборов параметров K компонентам. Другими словами, для любой заданной (невырожденной) точки в пространстве значений параметров будет существовать еще $K! - 1$ дополнительных точек, каждая из которых дает точно такое же распределение. Эта трудность известна как *отождествляемость* (*identifiability*) (Casella and Berger, 2002) и является важной проблемой при интерпретации значений параметров, найденных моделью. Отождествляемость также будет иметь место во время изучения моделей с непрерывными латентными переменными (см. главу 16). Однако для нахождения хорошей модели плотности это не имеет значения, поскольку любое из эквивалентных решений так же хорошо, как и любое другое.

15.2.2. Максимальное правдоподобие

Элегантный и мощный метод поиска решений с максимальным правдоподобием для моделей с латентными переменными называется *алгоритмом ожидания-максимизации* (*expectation–maximization algorithm*), или *EM-алгоритмом* (Dempster, Laird and Rubin, 1977; McLachlan and Krishnan, 1997). В этой главе представлены три различных вывода алгоритма EM, каждый из которых является более общим, чем предыдущий. Начнем с относительно неформального подхода в контексте модели гауссовой смеси. Однако стоит подчеркнуть, что EM имеет достаточно широкую применимость, и основные концепции будут встречаться в контексте целого ряда других моделей в этой книге.

Начнем с формулировки условий, которые должны выполняться при максимуме функции правдоподобия. Обратив в нуль производные $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ в (15.13) по средним значениям $\boldsymbol{\mu}_k$ гауссовых компонентов, получим:

$$0 = \sum_{n=1}^N \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k), \quad (15.15)$$

где используется формула (3.26) для гауссова распределения. Обратите внимание, что апостериорные вероятности, или обязанности, $\gamma(z_{nk})$, заданные в (15.12), естественным образом появляются в правой части уравнения. Умножая на $\boldsymbol{\Sigma}_k$ (который предположительно несингулярен) и делая перестановку, получаем:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n, \quad (15.16)$$

где определено

$$N_k = \sum_{n=1}^N \gamma(z_{nk}). \quad (15.17)$$

Можно интерпретировать N_k как эффективное число точек, отнесенных к кластеру k . Обратите внимание на форму этого решения. Здесь видно, что среднее значение $\boldsymbol{\mu}_k$ для k -й гауссовой компоненты получается взятием взвешенного среднего значения всех точек в наборе данных, при этом весовой коэффициент для точки данных \mathbf{x}_n задается апостериорной вероятностью $\gamma(z_{nk})$ того, что компонента k отвечает за генерацию \mathbf{x}_n .

Если определить производную $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ по $\boldsymbol{\Sigma}_k$ равной нулю и следовать аналогичной линии рассуждений с использованием результата для решения максимального правдоподобия (см. раздел 3.2.7) для ковариационной матрицы одного гауссова распределения, то получим:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k) (\mathbf{x}_n - \boldsymbol{\mu}_k)^T, \quad (15.18)$$

что аналогично соответствующему результату для одиночного гауссова распределения по набору данных, но в этом случае каждая точка данных взвешивается по соответствующей апостериорной вероятности, а знаменатель определяется эффективным числом точек, связанных с соответствующей компонентой.

Наконец, максимизируем $\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ относительно коэффициентов смещивания π_k . Здесь необходимо учесть ограничение (15.8), которое требует, чтобы коэффициенты смещивания в сумме равнялись единице. Этого можно добиться с помощью множителя Лагранжа λ (см. приложение С) и максимизации:

$$\ln p(\mathbf{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^K \pi_k - 1 \right), \quad (15.19)$$

что дает

$$0 = \sum_{n=1}^N \frac{\mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda, \quad (15.20)$$

где вновь наблюдается появление обязанностей. Если теперь умножить обе стороны на π_k и просуммировать по k с учетом ограничения (15.8), то получится $\lambda = -N$. Используя это для исключения λ и делая перестановку, получаем:

$$\pi_k = \frac{N_k}{N}. \quad (15.21)$$

Таким образом, коэффициент смещивания для k -го компонента определяется средней ответственностью, которую этот компонент принимает за объяснение точек данных.

Обратите внимание, что результаты (15.16), (15.18) и (15.21) не являются закрытым решением для параметров модели смеси, поскольку обязанности $\gamma(z_{nk})$ зависят от этих параметров комплексно через (15.12). Однако эти результаты позволяют использовать простую итерационную схему для поиска решения задачи максимального правдоподобия, которая, как будет показано далее, является вариантом ЕМ-алгоритма для конкретного случая модели гауссовой смеси. Сначала выбираются некоторые начальные величины для средних значений, ковариаций и коэффициентов смещивания. Затем поочередно выполняются два следующих обновления, которые будем называть Е-шагом и М-шагом по причинам, которые станут очевидными позже. На *этапе ожидания* (*expectation step*, или *E step*) используются актуальные значения параметров для оценки апостериорных вероятностей, или обязанностей, заданных в (15.12). Затем эти вероятности используются на *этапе максимизации* (*maximization step*, или *M step*) для переоценки средних значений, ковариаций и коэффициентов смещивания с использованием результатов (15.16), (15.18) и (15.21). Обратите внимание, что при этом вначале определяются новые средние значения с помощью (15.16), а затем эти новые значения используются для нахождения ковариаций с помощью (15.18) согласно соответствующему результату для одиночного гауссовского распределения. Далее будет показано, что каждое обновление параметров в результате шага Е, за которым следует шаг М (см. раздел 15.3), гарантированно увеличивает функцию логарифмического правдоподобия. На практике алгоритм считается сходящимся, если изменение функции логарифмического правдоподобия или – в качестве альтернативы – параметров оказывается ниже определенного порога.

На рис. 15.7 показано применение ЕМ-алгоритма для смеси двух гауссовых распределений к масштабированным данным Олд-Фейтфул. Здесь используется смесь двух гауссовых распределений, центры инициализируются теми же значениями, что и для алгоритма К-средних на рис. 15.1, а ковариационные матрицы инициализируются пропорционально единичной матрице. На

графике (а) зеленым цветом показаны точки данных, а также начальная конфигурация модели смеси, в которой контуры с одним стандартным отклонением для двух гауссовых компонентов показаны синим и красным кружками. На графике (б) показан результат начального шага Е, на котором каждая точка данных изображена с помощью доли синего цвета, равной апостериорной вероятности того, что она была сгенерирована синей компонентой, и соответствующей доли красного цвета, определяемой апостериорной вероятностью того, что она была сгенерирована красной компонентой. Таким образом, точки, вероятность принадлежности которых к одному из кластеров примерно равна, отображаются фиолетовым цветом. Ситуация после первого шага М показана на графике (с), где среднее значение синего гауссова распределения переместилось к среднему значению набора данных, взвешенному по вероятностям принадлежности каждой точки данных к синему кластеру. Другими словами, оно переместилось к центру масс синего кластера. Аналогично ковариация синей гауссовой составляющей равна ковариации синего цвета. Похожие результаты имеют место и для красного компонента. На графиках (д), (е) и (ф) показаны результаты после 2, 5 и 20 полных циклов EM соответственно. На графике (ф) алгоритм близок к сходимости.

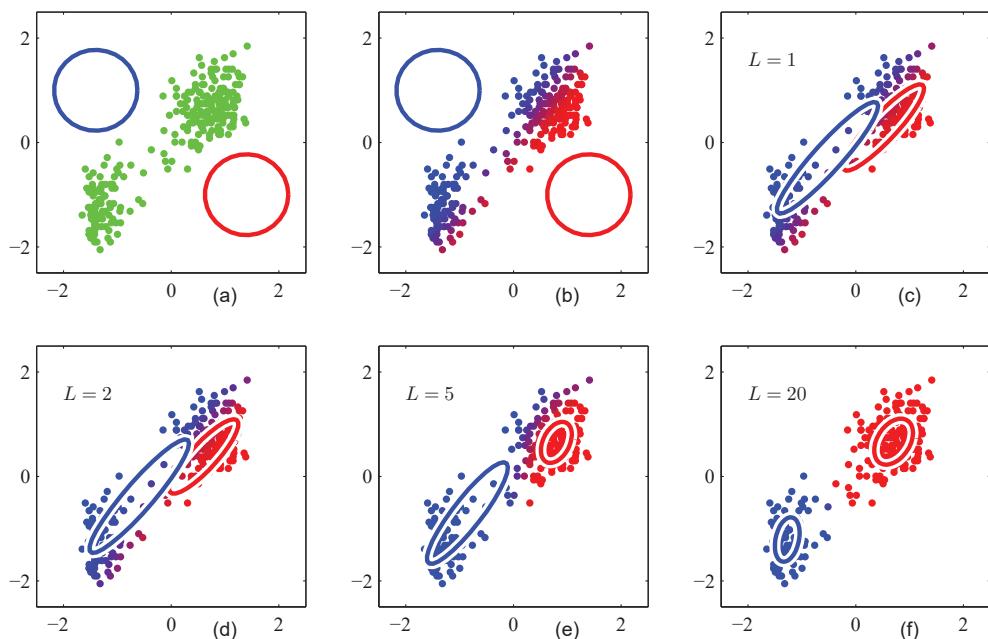


РИС. 15.7 Применение EM-алгоритма к набору данных Олд-Фейтфул, использованному для иллюстрации алгоритма К-средних на рис. 15.1

Обратите внимание, что для достижения (приблизительной) сходимости EM-алгоритму требуется гораздо больше итераций, чем алгоритму К-средних, и каждый цикл требует значительно больше вычислений. Поэтому обычно

алгоритм K -средних запускается с целью поиска подходящей инициализации для модели гауссовой смеси, которая впоследствии адаптируется с помощью ЕМ. Ковариационные матрицы удобно инициализировать выборочными ковариациями кластеров, найденными с помощью алгоритма K -средних, а коэффициенты смешивания – долями точек данных, отнесенных к соответствующим кластерам. Чтобы избежать сингулярностей функции правдоподобия, при которых гауссова компонента обрушивается на конкретную точку данных, необходимо использовать такие методы, как регуляризация параметров. Следует подчеркнуть, что обычно существует несколько локальных максимумов функции правдоподобия, и ЕМ не гарантирует нахождения наибольшего из этих максимумов. Поскольку ЕМ-алгоритм для гауссовых смесей играет такую важную роль, он кратко изложен в алгоритме 15.2.

АЛГОРИТМ 15.2 ЕМ-алгоритм для модели гауссовой смеси

Input: начальные параметры модели $\{\mu_k\}, \{\Sigma_k\}, \{\pi_k\}$

Набор данных $\{x_1, \dots, x_N\}$

Output: итоговые параметры модели $\{\mu_k\}, \{\Sigma_k\}, \{\pi_k\}$

repeat

```

// Шаг Е
for  $n \in \{1, \dots, N\}$  do
    for  $k \in \{1, \dots, K\}$  do
         $\gamma(z_{nk}) \leftarrow \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}$ 
    end for
end for
// Шаг М
for  $k \in \{1, \dots, K\}$  do
     $N_k \leftarrow \sum_{n=1}^N \gamma(z_{nk})$ 
     $\mu_k \leftarrow \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) x_n$ 
     $\Sigma_k \leftarrow \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (x_n - \mu_k)(x_n - \mu_k)^T$ 
     $\pi_k \leftarrow \frac{N_k}{N}$ 
end for
// Логарифмическое правдоподобие
 $\mathcal{L} \leftarrow \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\}$ 

```

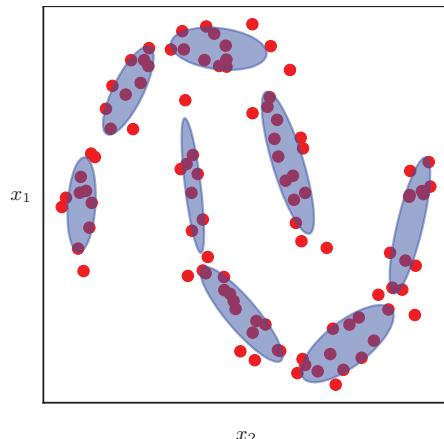
until convergence (до сходимости)

return $\{\mu_k\}, \{\Sigma_k\}, \{\pi_k\}$

Модели смесей очень гибкие и могут с высокой точностью аппроксимировать сложные распределения при достаточном количестве компонентов

при условии, что параметры модели подобраны соответствующим образом. На практике, однако, число компонентов может быть чрезвычайно велико, особенно в пространствах высокой размерности. Эта проблема проиллюстрирована для набора данных «two-moons» на рис. 15.8. Тем не менее модели смесей очень удобны для решения многих задач. Кроме того, понимание моделей смесей закладывает основы главы 16 для моделей с непрерывными латентными переменными и для генеративных моделей на основе глубоких нейронных сетей, которые гораздо лучше масштабируются к пространствам высокой размерности.

РИС. 15.8 Модель гауссовой смеси, подогнанная к набору данных «two-moons», показывает, что для точного представления сложного распределения данных может потребоваться большое количество компонентов смеси. Здесь эллипсы представляют собой контуры постоянной плотности для соответствующих компонентов смеси. При переходе к пространствам большей размерности количество компонентов, необходимых для точного моделирования распределения, может стать неприемлемо большим



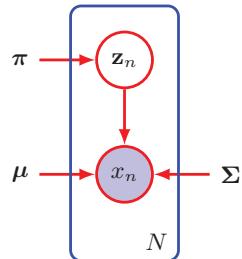
15.3. Алгоритм ожидания-максимизации

Теперь перейдем к более общему представлению ЕМ-алгоритма, где основное внимание уделяется роли латентных переменных. Как и прежде, обозначим множество всех наблюдаемых точек данных через \mathbf{X} , где n -я строка представляет собой \mathbf{x}_n^T . Аналогично соответствующие латентные переменные обозначим матрицей \mathbf{Z} размером $N \times K$ со строками \mathbf{z}_n^T . Если предположить, что точки данных берутся независимо из распределения, то можно выразить модель гауссовой смеси для этого набора i.i.d.-данных с помощью графа, показанного на рис. 15.9. Набор всех параметров модели обозначается θ , и поэтому функция логарифмического правдоподобия имеет вид:

$$\ln p(\mathbf{X} | \theta) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \theta) \right\}. \quad (15.22)$$

Обратите внимание, что эти рассуждения с тем же успехом можно применить и к непрерывным латентным переменным, просто заменив сумму по \mathbf{Z} интегралом (см. главу 16).

РИС. 15.9 Граф модели гауссовой смеси для набора из N i.i.d.-точек данных $\{\mathbf{x}_n\}$ с соответствующими латентными точками $\{\mathbf{z}_n\}$, где $n = 1, \dots, N$



Важно отметить, что суммирование по латентным переменным происходит внутри логарифма. Даже если совместное распределение $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ относится к экспоненциальному семейству, маргинальное распределение $p(\mathbf{X} | \boldsymbol{\theta})$ в результате этого суммирования обычно не является таковым. Наличие суммы не позволяет логарифму напрямую воздействовать на совместное распределение, что приводит к сложным выражениям для решения максимального правдоподобия.

Теперь предположим, что для каждого наблюдения в \mathbf{X} известно соответствующее значение латентной переменной \mathbf{Z} . Назовем $\{\mathbf{X}, \mathbf{Z}\}$ полным набором данных, а фактически наблюдаемые данные \mathbf{X} будем называть *неполными*, как показано на рис. 15.5. Функция правдоподобия для полного набора данных имеет вид $\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$, и в дальнейшем будем считать, что максимизация этой функции правдоподобия для полного набора данных является прямой.

На практике, однако, в распоряжении нет полного набора данных $\{\mathbf{X}, \mathbf{Z}\}$, а только неполные данные \mathbf{X} . Знания о значениях латентных переменных в \mathbf{Z} задаются только апостериорным распределением $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$. Поскольку использовать логарифмическое правдоподобие для полных данных невозможно, вместо этого рассматривается его ожидаемое значение при апостериорном распределении латентных переменных, что соответствует (как будет показано далее) шагу Е алгоритма EM. На последующем шаге М выполняется максимизация этого ожидания. Если обозначить актуальную оценку параметров $\boldsymbol{\theta}^{\text{old}}$, то пара последовательных шагов Е и М приводит к пересмотренной оценке $\boldsymbol{\theta}^{\text{new}}$. Инициализация алгоритма осуществляется путем выбора некоторого начального значения параметров $\boldsymbol{\theta}^0$. Хотя такое использование математического ожидания может показаться несколько произвольным, в разделе 15.4 при более глубоком рассмотрении EM причина такого выбора станет очевидна.

На шаге Е с помощью текущих значений параметров $\boldsymbol{\theta}^{\text{old}}$ определяется апостериорное распределение латентных переменных, заданное $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}})$. Затем с помощью этого апостериорного распределения можно найти ожидание логарифмического правдоподобия полных данных, которое оценивается для некоторого общего значения параметра $\boldsymbol{\theta}$. Это ожидание, обозначаемое $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$, задается как

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}). \quad (15.23)$$

На шаге М определяется пересмотренная оценка параметра $\boldsymbol{\theta}^{\text{new}}$ путем максимизации этой функции:

$$\boldsymbol{\theta}^{\text{new}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}). \quad (15.24)$$

Обратите внимание, что в определении $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ логарифм действует непосредственно на совместное распределение $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$, и поэтому соответствующая максимизация на М-шаге будет, согласно принятому предположению, выполнимой. Общий алгоритм ЕМ кратко изложен в алгоритме 15.3. Он обладает свойством, о котором будет рассказано далее, что каждый цикл ЕМ (см. раздел 15.4.1) будет увеличивать логарифмическое правдоподобие неполных данных (если только оно уже не находится в локальном максимуме).

АЛГОРИТМ 15.3. Обобщенный ЕМ-алгоритм

Input: Совместное распределение $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$

Начальные параметры $\boldsymbol{\theta}^{\text{old}}$

Набор данных $\mathbf{x}_1, \dots, \mathbf{x}_N$

Output: итоговые параметры $\boldsymbol{\theta}$

repeat

```

 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \leftarrow \sum_z p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) // \text{Шаг Е}$ 
 $\boldsymbol{\theta}^{\text{new}} \leftarrow \arg \max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) // \text{Шаг М}$ 
 $L \leftarrow p(\mathbf{X} | \boldsymbol{\theta}^{\text{new}}) // \text{Оценка логарифмического правдоподобия}$ 
 $\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} // \text{Обновление параметров}$ 

```

until convergence (до сходимости)

return $\boldsymbol{\theta}^{\text{new}}$

Также ЕМ-алгоритм можно использовать как решение для поиска *MAP* (*maximum posterior*, максимум апостериорной вероятности) (см. упражнение 15.5) для моделей, в которых для параметров задано априорное значение $p(\boldsymbol{\theta})$. В этом случае шаг Е остается таким же, как и в случае поиска максимального правдоподобия, а на шаге М величина, которую нужно максимизировать, определяется как $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) + \ln p(\boldsymbol{\theta})$. Подходящий выбор априорного значения позволит устранить сингулярности такого рода, как показано на рис. 15.6.

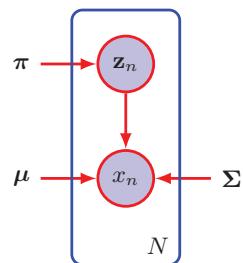
В этом разделе было рассмотрено использование ЕМ-алгоритма для максимизации функции правдоподобия при наличии дискретных латентных переменных. Однако его можно применять и в тех случаях, когда ненаблюдаемые переменные соответствуют отсутствующим значениям в наборе данных. Распределение наблюдаемых значений можно получить при помощи совместного распределения всех переменных и маргинализации по отсутствующим значениям. Затем ЕМ можно использовать для максимизации соответствующей функции правдоподобия. Эта процедура будет корректной, если значения данных *отсутствуют случайным образом*, т. е. механизм, при-

водящий к отсутствию значений, не зависит от ненаблюдаемых значений. Во многих ситуациях это не так, например если датчик не выдает значение всякий раз, когда измеряемая им величина превышает некоторый порог.

15.3.1. Гауссовые смеси

Теперь рассмотрим применение этого представления ЕМ с латентными переменными к конкретному примеру модели гауссовой смеси. Напомним, что целью исследования является максимизация функции логарифмического правдоподобия (15.13), которая вычисляется по наблюдаемому набору данных \mathbf{X} . Ранее уже отмечалось, что это сложнее, чем при использовании одного гауссова распределения из-за суммирования по k , которое происходит внутри логарифма. Предположим, что в дополнение к наблюдаемому набору данных \mathbf{X} были заданы значения соответствующих дискретных переменных \mathbf{Z} . Напомним, что на рис. 15.5a показан полный набор данных (т. е. содержащий метки с указанием того, какой компонент породил каждую точку данных), а на рис. 15.5b – соответствующий неполный набор данных. Модель графа для полных данных показана на рис. 15.10.

РИС. 15.10 Тот же график, что и на рис. 15.9, только теперь предполагается, что дискретные переменные \mathbf{z}_n наблюдаются, так же как и переменные данных \mathbf{x}_n



Теперь рассмотрим задачу максимизации правдоподобия для полного набора данных $\{\mathbf{X}, \mathbf{Z}\}$. Из (15.9) и (15.10) следует, что функция правдоподобия имеет вид:

$$p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \prod_{n=1}^N \prod_{k=1}^K \pi_k^{z_{nk}} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{nk}}, \quad (15.25)$$

где z_{nk} обозначает k -ю компоненту \mathbf{z}_n . Взяв логарифм, получаем:

$$\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \prod_{n=1}^N \prod_{k=1}^K z_{nk} \{ \ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \}. \quad (15.26)$$

Сравнение с функцией логарифмического правдоподобия (15.13) для неполных данных показывает, что суммирование по k и логарифм поменялись местами. Теперь логарифм действует непосредственно на гауссово распределение, которое само является членом экспоненциального семейства. Неудивительно, что это приводит к гораздо более простому решению задачи мак-

симального правдоподобия, как сейчас будет показано. Сначала рассмотрим максимизацию относительно средних значений и ковариаций. Поскольку \mathbf{z}_n – это K -мерный вектор, все элементы которого равны 0, за исключением одного элемента, имеющего значение 1, то функция правдоподобия полного логарифма данных – это просто сумма K независимых вкладов, по одному на каждый компонент смеси. Таким образом, максимизация относительно среднего значения или ковариации происходит точно так же, как и для одного гауссова компонента, за исключением того, что она затрагивает только подмножество точек данных, которые «приписаны» к этому компоненту. Что касается максимизации по коэффициентам смещивания, обратите внимание, что они связаны между собой для разных значений k в силу ограничения на суммирование (15.8). Опять же, как и ранее, это ограничение может быть реализовано с помощью множителя Лагранжа, что приводит к результату

$$\pi_k = \frac{1}{N} \sum_{n=1}^N z_{nk}, \quad (15.27)$$

так что коэффициенты смещивания равны долям точек данных, отнесенных к соответствующим компонентам.

Получается, что функция логарифмического правдоподобия для полных данных может быть тривиально максимизирована в замкнутой форме. Однако на практике значения латентных переменных не известны. Поэтому, как обсуждалось ранее, рассматривается ожидание полного логарифмического правдоподобия относительно апостериорного распределения латентных переменных. Используя (15.9) и (15.10) вместе с теоремой Байеса, можно увидеть, что это апостериорное распределение имеет вид:

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \propto \prod_{n=1}^N \prod_{k=1}^K [\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{nk}}. \quad (15.28)$$

Здесь показано, что факторизация по n такова, что при апостериорном распределении $\{z_n\}$ независимы (см. упражнение 15.6). В этом легко убедиться, рассмотрев направленный граф на рис. 15.9 и воспользовавшись критерием d -разделения (см. раздел 11.2). Ожидаемое значение индикаторной переменной z_{nk} при таком апостериорном распределении определяется как

$$\begin{aligned} \mathbb{E}[z_{nk}] &= \frac{\sum_{\mathbf{z}_n} z_{nk} \prod_{k'} [\pi_{k'} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})]^{z_{nk'}}}{\sum_{\mathbf{z}_n} \prod_j [\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)]^{z_{nj}}} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} = \gamma(z_{nk}), \end{aligned} \quad (15.29)$$

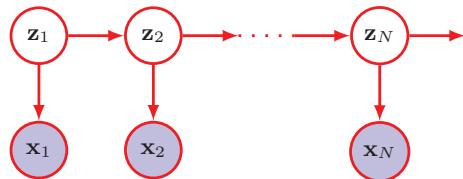
что является просто ответственностью компонента k за точку данных \mathbf{x}_n . Таким образом, ожидаемое значение функции правдоподобия для полного набора данных определяется как

$$\mathbb{E}_z[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] = \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \{\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}. \quad (15.30)$$

Теперь остается действовать следующим образом. Сначала выберем некоторые начальные значения параметров $\boldsymbol{\mu}^{\text{old}}$, $\boldsymbol{\Sigma}^{\text{old}}$ и $\boldsymbol{\pi}^{\text{old}}$ и используем их для оценки обязанностей (шаг E). Затем сохраняем обязанности фиксированными и максимизируем (15.30) относительно $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ и π_k (шаг M). Это приводит к замкнутым решениям для $\boldsymbol{\mu}^{\text{new}}$, $\boldsymbol{\Sigma}^{\text{new}}$ и $\boldsymbol{\pi}^{\text{new}}$ (см. упражнение 15.9), которые, как и прежде, задаются в (15.16), (15.18) и (15.21). Это именно тот EM-алгоритм для гауссовых смесей, который был получен ранее. Более подробно о роли ожидаемой функции логарифмического правдоподобия для полных данных будет рассказано при обсуждении сходимости EM-алгоритма в разделе 15.4.

На протяжении всей этой главы подразумевается, что данные наблюдения являются i.i.d. Для упорядоченных наблюдений, образующих последовательность, модель смеси можно расширить путем соединения латентных переменных в цепь Маркова для получения *скрытой модели Маркова*, графовая структура которой показана на рис. 15.11. Для этой более сложной модели EM-алгоритм можно расширить, при этом шаг E включает последовательное вычисление, в ходе которого сообщения передаются по цепочке латентных переменных (Bishop, 2006).

РИС. 15.11 Вероятностная графовая модель для последовательных данных, соответствующая скрытой марковской модели. Дискретные латентные переменные больше не являются независимыми, но образуют цепь Маркова



15.3.2. Сравнение с алгоритмом K-средних

Сравнение алгоритма *K*-средних с EM-алгоритмом для гауссовых смесей показывает их близкое сходство. В то время как алгоритм *K*-средних выполняет жесткое распределение точек данных по кластерам, где каждая точка данных однозначно ассоциируется с одним кластером, EM-алгоритм выполняет мягкое распределение на основе апостериорных вероятностей. В действительности алгоритм *K*-средних представляет собой частный случай EM для гауссовых смесей и может быть выведен следующим образом.

Рассмотрим модель гауссовой смеси, где ковариационные матрицы компонентов смеси заданы $\epsilon\mathbf{I}$, где ϵ – это параметр дисперсии, общий для всех компонентов, а \mathbf{I} – это матрица тождества, так что

$$p(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi\epsilon)^{D/2}} \exp\left\{-\frac{1}{2\epsilon}\|\mathbf{x} - \boldsymbol{\mu}_k\|^2\right\}. \quad (15.31)$$

Теперь рассмотрим EM-алгоритм для смеси *K* гауссовых компонентов такого вида, где вместо параметра, подлежащего переоценке, фигурирует

фиксированная константа. Из (15.12) апостериорные вероятности, или обязанности, для конкретной точки данных \mathbf{x}_n определяются как

$$\gamma(z_{nk}) = \frac{\pi_k \exp\{-\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2/2\epsilon\}}{\sum_j \pi_j \exp\{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2/2\epsilon\}}. \quad (15.32)$$

Рассмотрим предел $\epsilon \rightarrow 0$. Знаменатель состоит из суммы членов, индексируемых j , каждый из которых обращается в ноль. Определенный член, для которого $\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2$ наименьший, скажем $j = l$, будет обращаться в ноль медленнее всего и будет доминировать в этой сумме. Таким образом, ответственность $\gamma(z_{nk})$ для точки данных \mathbf{x}_n обращается в ноль, за исключением члена l , для которого ответственность $\gamma(z_{nl})$ обращается в единицу. Обратите внимание, что это происходит независимо от значений π_k , пока ни один из π_k не равен нулю. Получается, что в этом пределе получается жесткое распределение точек данных по кластерам, как в алгоритме K -средних, так что $\gamma(z_{nk}) \rightarrow r_{nk}$, где r_{nk} определяется по (15.2). Выходит, что каждая точка данных попадает в кластер с наиболее близким средним значением. Уравнение ЕМ для переоценки $\boldsymbol{\mu}_k$, заданное в (15.16), сводится к результату K -средних (15.4). Обратите внимание, что формула переоценки для коэффициентов смешивания (15.21) просто сбрасывает значение π_k , чтобы оно было равно доле точек данных, отнесенных к кластеру k , хотя эти параметры больше не играют активной роли в алгоритме.

Наконец, в пределе $\epsilon \rightarrow 0$ ожидаемое логарифмическое правдоподобие для полных данных, заданное в (15.30) (см. упражнение 15.12), становится равным

$$\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] \rightarrow -\frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 + \text{const.} \quad (15.33)$$

Таким образом, получается, что в этом пределе максимизация ожидаемого логарифмического правдоподобия для полных данных эквивалентна минимизации меры ошибки J для алгоритма K -средних, заданной в (15.1). Обратите внимание, что алгоритм K -средних не оценивает ковариации кластеров, а только кластерные средние значения.

15.3.3. Смеси распределений Бернулли

До сих пор в этой главе речь шла о распределениях непрерывных переменных, описываемых смесями гауссовых распределений. В качестве еще одного примера моделирования смесей и для иллюстрации работы ЕМ-алгоритма в другом контексте рассмотрим смесь дискретных бинарных переменных, описываемых распределениями Бернулли. Эта модель также известна как *анализ латентных классов* (*latent class analysis*) (Lazarsfeld and Henry, 1968; McLachlan and Peel, 2000).

Рассмотрим набор из D бинарных переменных \mathbf{x}_i , где $i = 1, \dots, D$, каждая из которых управляет распределением Бернулли с параметром μ_i (см. раздел 3.1.1), так что

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{i=1}^D \mu_i^{x_i} (1 - \mu_i)^{1-x_i}, \quad (15.34)$$

где $\mathbf{x} = (x_1, \dots, x_D)^\top$ и $\boldsymbol{\mu} = (\mu_1, \dots, \mu_D)^\top$. Как видно, отдельные переменные x_i независимы при данном $\boldsymbol{\mu}$. Среднее значение и ковариация этого распределения (см. упражнение 15.13), что очевидно, равны

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}, \quad (15.35)$$

$$\text{cov}[\mathbf{x}] = \text{diag}\{\mu_i(1 - \mu_i)\}. \quad (15.36)$$

Теперь рассмотрим конечную смесь этих распределений, представленную в виде

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{k=1}^K \pi_k p(\mathbf{x}|\boldsymbol{\mu}_k), \quad (15.37)$$

где $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}$, $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_K\}$, и

$$p(\mathbf{x}|\boldsymbol{\mu}_k) = \prod_{i=1}^D \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i}. \quad (15.38)$$

Коэффициенты смешивания удовлетворяют (15.7) и (15.8). Среднее значение и ковариация этого распределения смеси (см. упражнение 15.14) определяются как

$$\mathbb{E}[\mathbf{x}] = \sum_{k=1}^K \pi_k \boldsymbol{\mu}_k, \quad (15.39)$$

$$\text{cov}[\mathbf{x}] = \sum_{k=1}^K \pi_k \{\boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^\top\} - \mathbb{E}[\mathbf{x}] \mathbb{E}[\mathbf{x}]^\top, \quad (15.40)$$

где $\boldsymbol{\Sigma}_k = \text{diag}\{\mu_{ki}(1 - \mu_{ki})\}$. Поскольку ковариационная матрица $\text{cov}[\mathbf{x}]$ больше не является диагональной, распределение смеси может отражать корреляции между переменными, в отличие от одиночного распределения Бернулли.

Если дан набор данных $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, то функция логарифмического правдоподобия для этой модели имеет вид:

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k p(\mathbf{x}_n|\boldsymbol{\mu}_k) \right\}. \quad (15.41)$$

Здесь вновь наблюдается появление суммирования внутри логарифма, так что решение для максимального правдоподобия больше не имеет замкнутого вида.

Теперь выведем EM-алгоритм максимизации функции правдоподобия для смеси распределений Бернулли. Для этого сначала введем явную дискретную латентную переменную \mathbf{z} , связанную с каждым экземпляром \mathbf{x} . Как и в случае гауссовой смеси, \mathbf{z} имеет кодирование «1 из K », так что $\mathbf{z} = (z_1, \dots, z_K)^\top$ – это дво-

ичный K -мерный вектор, у которого одна компонента равна 1, а все остальные компоненты равны 0. Тогда условное распределение \mathbf{x} относительно латентной переменной можно записать как

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}) = \prod_{k=1}^K p(\mathbf{x}|\boldsymbol{\mu}_k)^{z_k}, \quad (15.42)$$

и априорное распределение для латентных переменных такое же, как и для модели гауссовой смеси, так что

$$p(\mathbf{z}|\boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_k}. \quad (15.43)$$

Если составить произведение $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu})$ и $p(\mathbf{z}|\boldsymbol{\pi})$, а затем провести маргинализацию по \mathbf{z} , то получится (15.37) (см. упражнение 15.16).

Чтобы получить ЕМ-алгоритм, сначала запишем функцию правдоподобия для полных данных, которая имеет вид:

$$\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \left\{ \ln \pi_k + \sum_{i=1}^D [x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln (1 - \mu_{ki})] \right\}, \quad (15.44)$$

где $\mathbf{X} = \{\mathbf{x}_n\}$ и $\mathbf{Z} = \{\mathbf{z}_n\}$. Далее возьмем ожидание логарифмического правдоподобия для полных данных относительно апостериорного распределения латентных переменных, чтобы получить

$$\begin{aligned} \mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\pi})] &= \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \left\{ \ln \pi_k \right. \\ &\quad \left. + \sum_{i=1}^D [x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln (1 - \mu_{ki})] \right\}, \end{aligned} \quad (15.45)$$

где $\gamma(z_{nk}) = \mathbb{E}[z_{nk}]$ – это апостериорная вероятность, или ответственность, компонента k для точки данных \mathbf{x}_n . На шаге Е эти ответственности оцениваются с помощью теоремы Байеса, которая имеет вид:

$$\begin{aligned} \gamma(z_{nk}) &= \mathbb{E}[z_{nk}] = \frac{\sum_{\mathbf{z}_n} z_{nk} \prod_{k'} [\pi_{k'} p(\mathbf{x}_n | \boldsymbol{\mu}_{k'})]^{z_{nk'}}}{\sum_{\mathbf{z}_n} \prod_j [\pi_j p(\mathbf{x}_n | \boldsymbol{\mu}_j)]^{z_{nj}}} \\ &= \frac{\pi_k p(\mathbf{x}_n | \boldsymbol{\mu}_k)}{\sum_{j=1}^K \pi_j p(\mathbf{x}_n | \boldsymbol{\mu}_j)}. \end{aligned} \quad (15.46)$$

Если рассматривать сумму по n в (15.45), то видно, что обязанности входят только через два члена, которые можно записать в виде

$$N_k = \sum_{n=1}^N \gamma(z_{nk}), \quad (15.47)$$

$$\bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n, \quad (15.48)$$

где N_k – это эффективное число точек данных, связанных с компонентом k . На шаге М производится максимизация ожидаемого логарифмического правдоподобия по полным данным с учетом параметров μ_k и π . Если производную от (15.45) по μ_k записать равной нулю и переставить члены (упражнение 15.17), то получим:

$$\mu_k = \bar{\mathbf{x}}_k. \quad (15.49)$$

В результате среднее значение компонента k становится равным средневзвешенному значению данных, а весовые коэффициенты определяются обязанностями, которые компонент k принимает для каждой из точек данных. Для максимизации относительно π_k необходимо ввести множитель Лагранжа для ограничения $\sum_k \pi_k = 1$. Выполнив действия, аналогичные шагам для гауссовой смеси (см. упражнение 15.18), получаем

$$\pi_k = \frac{N_k}{N}, \quad (15.50)$$

что является достаточно очевидным фактом: коэффициент смещивания для компонента k определяется эффективной долей точек в наборе данных, обусловленных этим компонентом.

Обратите внимание, что, в отличие от смеси гауссовых компонент, здесь нет сингулярностей, при которых функция правдоподобия обращается в бесконечность. В этом можно убедиться с помощью определения того, что функция правдоподобия ограничена сверху, поскольку $0 \leq p(\mathbf{x}_n | \mu_k) \leq 1$ (см. упражнение 15.19). Существуют решения, для которых функция правдоподобия равна нулю, но они не будут найдены методом ЕМ при условии, что он не инициализирован в патологической начальной точке, поскольку ЕМ-алгоритм всегда увеличивает значение функции правдоподобия вплоть до нахождения локального максимума (см. раздел 15.3).

На рис. 15.12 представлена модель смеси Бернулли, которая используется для моделирования рукописных цифр. Здесь изображения цифр были преобразованы в двоичные векторы путем установки всех элементов, значения которых превышают 0,5, в 1, а остальные элементы установлены в 0. Теперь набор данных из $N = 600$ таких цифр, включающий цифры «2», «3» и «4», подгоняется под смесь $K = 3$ распределений Бернулли путем выполнения 10 итераций ЕМ-алгоритма. Коэффициенты смещивания были инициализированы $\pi_k = 1/K$, а параметры μ_{kj} были установлены на случайные значения, равномерно выбранные в диапазоне $(0,25, 0,75)$, и затем нормализованы для выполнения ограничения, согласно которому $\sum_j \mu_{kj} = 1$. Смесь трех распределений Бернулли позволяет найти три кластера в наборе данных, соответствующих разным цифрам. Анализ смесей Бернулли (см. упражнение 15.20)

легко расширить на случай многочленных двоичных переменных с $M > 2$ состояниями с помощью дискретного распределения (3.14).

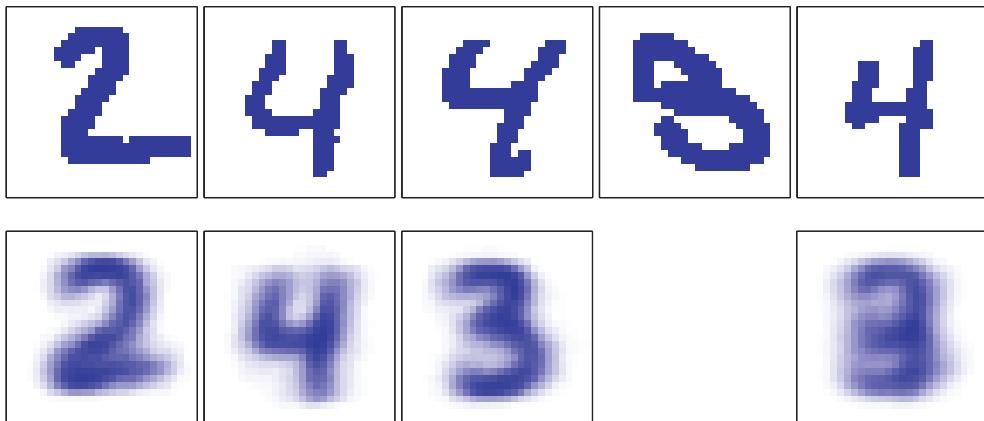


РИС. 15.12 Иллюстрация модели смеси Бернулли, где в верхнем ряду показаны примеры из набора данных по цифрам после преобразования значений пикселей из серой шкалы в двоичную с использованием порога 0,5. В нижней строке первые три изображения показывают параметры μ_{ki} для каждого из трех компонентов модели смеси. Для сравнения: тот же набор данных был рассчитан с помощью одного многомерного распределения Бернулли, опять же с использованием максимального правдоподобия. Это равносильно простому усреднению подсчетов в каждом пикселе (на самом правом изображении в нижнем ряду)

15.4. Нижняя граница доказательств

Теперь рассмотрим еще более общий взгляд на алгоритм EM, выведя нижнюю границу для функции логарифмического правдоподобия, которая известна как *нижняя граница доказательств* (*evidence lower bound*, ELBO). Иногда ее также называют *нижней вариационной границей* (*variational lower bound*). Здесь термин «доказательство» относится к функции (log) правдоподобия, которую иногда называют «доказательством модели» (*model evidence*) в байесовском подходе, поскольку она позволяет сравнивать различные модели без использования удержаных данных (Bishop, 2006). В качестве иллюстрации этого ограничения используем его для повторного вывода EM-алгоритма для гауссовых смесей с третьего подхода. Метод ELBO будет играть важную роль при работе с некоторыми глубокими генеративными моделями, о которых пойдет речь в последующих главах. Он также является примером *вариационного фреймворка*, где вводится распределение $q(\mathbf{Z})$ по латентным переменным, а затем производится оптимизация относительно этого распределения с помощью вариационного исчисления (см. приложение B).

Рассмотрим вероятностную модель, где все наблюдаемые переменные обозначим \mathbf{X} , а все скрытые переменные обозначим \mathbf{Z} . Совместное распре-

деление $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ управляет набором параметров, обозначаемых $\boldsymbol{\theta}$. Целью является максимизация функции правдоподобия:

$$p(\mathbf{X} | \boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}). \quad (15.51)$$

Здесь подразумевается, что переменные \mathbf{Z} являются дискретными, хотя ход рассуждений будет идентичным, если \mathbf{Z} будет включать непрерывные переменные или комбинации дискретных и непрерывных переменных, а суммирование при необходимости будет заменено интегрированием.

Предположим, что прямая оптимизация $p(\mathbf{X} | \boldsymbol{\theta})$ затруднительна, но оптимизация функции правдоподобия для полных данных $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ значительно проще. Далее введем распределение $q(\mathbf{Z})$ по латентным переменным и отметим, что при любом выборе $q(\mathbf{Z})$ имеет место следующее разложение:

$$\ln p(\mathbf{X} | \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \text{KL}(q \| p), \quad (15.52)$$

где определено

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}, \quad (15.53)$$

$$\text{KL}(q \| p) = - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}. \quad (15.54)$$

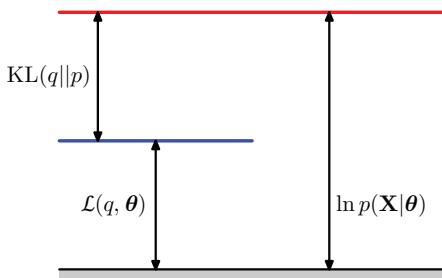
Обратите внимание, что $\mathcal{L}(q, \boldsymbol{\theta})$ – это функция распределения $q(\mathbf{Z})$ и функция параметров $\boldsymbol{\theta}$ (см. приложение B). Стоит внимательно изучить формы выражений (15.53) и (15.54), в частности, обратить внимание на то, что они отличаются по знаку, а также на то, что $\mathcal{L}(q, \boldsymbol{\theta})$ содержит совместное распределение \mathbf{X} и \mathbf{Z} , тогда как $\text{KL}(q \| p)$ содержит условное распределение \mathbf{Z} по \mathbf{X} . Для проверки разложения (15.52) сначала воспользуемся правилом произведения вероятностей, чтобы получить

$$\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) = \ln p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X} | \boldsymbol{\theta}), \quad (15.55)$$

который затем подставляем в выражение для $\mathcal{L}(q, \boldsymbol{\theta})$. Это дает два члена, один из которых отменяет $\text{KL}(q \| p)$, а другой дает требуемое логарифмическое правдоподобие $\ln p(\mathbf{X} | \boldsymbol{\theta})$ после того, как будет отмечено, что $q(\mathbf{Z})$ – это нормализованное распределение, которое в сумме равно 1.

Из (15.54) следует, что $\text{KL}(q \| p)$ – это расхождение Кульбака–Лейблера между $q(\mathbf{Z})$ и апостериорным распределением $p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$. Напомним, что расхождение Кульбака–Лейблера (см. раздел 2.5.7) удовлетворяет условию $\text{KL}(q \| p) \geq 0$ при равенстве тогда и только тогда, когда $q(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$. Поэтому из (15.52) следует, что $\mathcal{L}(q, \boldsymbol{\theta}) \leq \ln p(\mathbf{X} | \boldsymbol{\theta})$, или, другими словами, $\mathcal{L}(q, \boldsymbol{\theta})$ является нижней границей для $\ln p(\mathbf{X} | \boldsymbol{\theta})$. Разложение (15.52) проиллюстрировано на рис. 15.13.

РИС. 15.13 Иллюстрация разложения по (15.52), которое справедливо для любого выбора распределения $q(\mathbf{Z})$. Поскольку расхождение Кульбака–Лейблера удовлетворяет условию $KL(q||p) \geq 0$, получается, что величина $L(q, \boldsymbol{\theta})$ является нижней границей для функции логарифмического правдоподобия $\ln p(\mathbf{X}|\boldsymbol{\theta})$



15.4.1. Новый взгляд на ЕМ

С помощью разложения (15.52) можно вывести ЕМ-алгоритм и убедиться, что он действительно максимизирует логарифмическое правдоподобие. Предположим, что актуальное значение вектора параметров равно $\boldsymbol{\theta}^{\text{old}}$. На шаге Е нижняя граница $L(q, \boldsymbol{\theta}^{\text{old}})$ максимизируется относительно $q(\mathbf{Z})$ при фиксированном $\boldsymbol{\theta}^{\text{old}}$. Решение этой задачи максимизации можно легко определить, если обратить внимание на то, что значение $\ln p(\mathbf{X}|\boldsymbol{\theta}^{\text{old}})$ не зависит от $q(\mathbf{Z})$, и поэтому наибольшее значение $L(q, \boldsymbol{\theta}^{\text{old}})$ достигается при исчезновении расхождения Кульбака–Лейблера, или, другими словами, когда $q(\mathbf{Z})$ равно апостериорному распределению $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$. В этом случае нижняя граница будет равна логарифмическому правдоподобию, как показано на рис. 15.14.

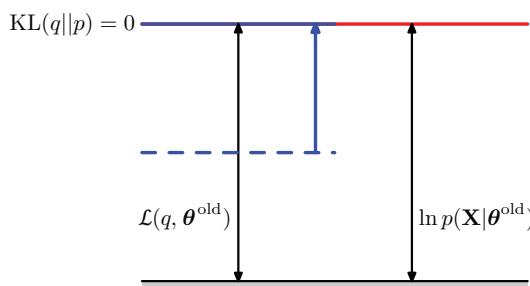


РИС. 15.14 Схема шага Е в ЕМ-алгоритме. Распределение q устанавливается равным апостериорному распределению для актуальных значений $\boldsymbol{\theta}^{\text{old}}$, в результате чего нижняя граница увеличивается до того же значения, что и функция логарифмического правдоподобия, а расхождение KL исчезает

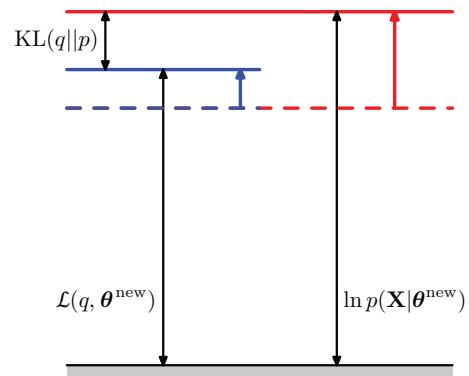
При последующем шаге М распределение $q(\mathbf{Z})$ остается фиксированным, а нижняя граница $L(q, \boldsymbol{\theta})$ максимизируется относительно $\boldsymbol{\theta}$ с получением некоторого нового значения $\boldsymbol{\theta}^{\text{new}}$. Это приведет к увеличению нижней границы L (если только она уже не достигла максимума), что обязательно приведет к увеличению соответствующей функции логарифмического правдоподобия. Поскольку распределение q определяется по старым, а не по новым значениям параметров и остается фиксированным на шаге М, оно не будет равно новому апостериорному распределению $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{new}})$, и, следовательно, будет иметь место ненулевое расхождение Кульбака–Лейблера. Поэтому увеличе-

ние функции логарифмического правдоподобия больше, чем увеличение нижней границы, как показано на рис. 15.15. Если подставить $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}})$ в (15.53), то можно убедиться, что после шага Е нижняя граница принимает вид:

$$\begin{aligned}\mathcal{L}(q, \theta) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \\ &= \mathcal{Q}(\theta, \theta^{\text{old}}) + \text{const},\end{aligned}\quad (15.56)$$

где константа – это просто отрицательная энтропия распределения q , и поэтому она не зависит от θ . Здесь под $\mathcal{Q}(\theta, \theta^{\text{old}})$ подразумевается ожидаемое полное логарифмическое правдоподобие, определяемое в (15.23), и именно это значение максимизируется на шаге М (см. раздел 15.3), как это было показано ранее в распределении для гауссовых смесей. Обратите внимание, что переменная θ , по которой производится оптимизация, появляется только внутри логарифма. Если совместное распределение $p(\mathbf{Z}, \mathbf{X}|\theta)$ является членом экспоненциального семейства или произведением таких членов, то получается, что логарифм отменяет экспоненту и приводит к М-шагу, который обычно намного проще, чем максимизация соответствующей функции правдоподобия $p(\mathbf{X}|\theta)$ для неполных данных.

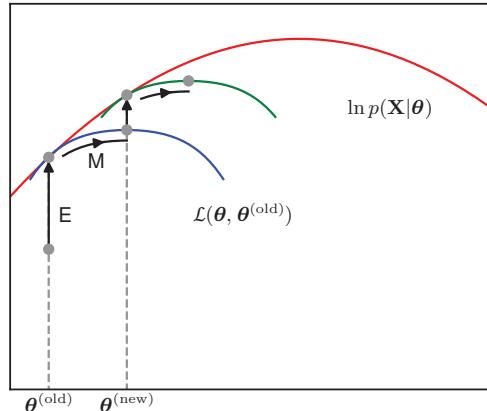
РИС. 15.15 Иллюстрация шага М в ЕМ-алгоритме. Распределение $q(\mathbf{Z})$ фиксировано, и нижняя граница $L(q, \theta)$ максимизируется по вектору параметров θ для получения пересмотренного значения θ^{new} . Поскольку расхождение Кульбака–Лейблера неотрицательно, это приводит к росту логарифмического правдоподобия $\ln p(\mathbf{X}|\theta)$ по крайней мере на такую же величину, как и нижняя граница



Работа ЕМ-алгоритма также может быть рассмотрена в пространстве параметров, как схематично показано на рис. 15.16. Здесь красная кривая изображает (неполную по данным) функцию логарифмического правдоподобия, значение которой необходимо максимизировать. Начнем с некоторого начального значения параметра θ^{old} , и на первом шаге Е оценим апостериорное распределение по латентным переменным, что дает нижнюю границу $\mathcal{L}(q, \theta)$, значение которой равно логарифмическому правдоподобию при θ^{old} , как показано синей кривой. Обратите внимание, что эта граница пересекается по касательной с логарифмическим правдоподобием при θ^{old} , так что обе кривые имеют одинаковый градиент (см. упражнение 15.22). Эта граница – выпуклая функция, имеющая единственный максимум (для компонентов смеси из экспоненциального семейства). На шаге М эта граница максимизи-

руется и дает значение $\boldsymbol{\theta}^{(\text{new})}$, которое увеличивает значение логарифмического правдоподобия по сравнению с $\boldsymbol{\theta}^{(\text{old})}$. Последующий шаг E создает границу, которая является касательной к $\boldsymbol{\theta}^{(\text{new})}$, как показано зеленой кривой.

РИС. 15.16 EM-алгоритм поочередно вычисляет нижнюю границу логарифмического правдоподобия для актуальных значений параметров, а затем максимизирует эту границу для получения новых значений параметров



Ранее уже выяснилось, что шаги E и M в EM-алгоритме увеличивают значение вполне определенной границы функции логарифмического правдоподобия, а полный EM-цикл изменяет параметры модели таким образом, что логарифмическое правдоподобие увеличивается (если только оно уже не достигло максимума, и тогда параметры остаются неизменными).

15.4.2. Независимые и одинаково распределенные данные

В частном случае i.i.d.-набора данных \mathbf{X} входят N точек данных $\{\mathbf{x}_n\}$, а в \mathbf{Z} входят N соответствующих латентных переменных $\{\mathbf{z}_n\}$, где $n = 1, \dots, N$. Из условия о независимости следует, что $p(\mathbf{X}, \mathbf{Z}) = \prod_n p(\mathbf{x}_n, \mathbf{z}_n)$, а при маргинализации по $\{\mathbf{z}_n\}$ получается $p(\mathbf{X}) = \prod_n p(\mathbf{x}_n)$. Используя правила суммы и произведения, получается, что апостериорная вероятность, которая оценивается на шаге E, имеет вид:

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) = \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{\sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})} = \frac{\prod_{n=1}^N p(\mathbf{x}_n, \mathbf{z}_n | \boldsymbol{\theta})}{\sum_{\mathbf{Z}} \prod_{n=1}^N p(\mathbf{x}_n, \mathbf{z}_n | \boldsymbol{\theta})} = \prod_{n=1}^N p(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\theta}). \quad (15.57)$$

Для модели гауссовой смеси это всего лишь означает, что ответственность каждого из компонентов смеси за конкретную точку данных \mathbf{x}_n зависит только от значения \mathbf{x}_n и параметров $\boldsymbol{\theta}$ компонентов смеси, но не от значений других точек данных.

15.4.3. Априорные параметры

Можно также использовать EM-алгоритм для максимизации апостериорного распределения $p(\boldsymbol{\theta}|\mathbf{X})$ для моделей, в которых для параметров введено

априорное значение $p(\boldsymbol{\theta})$. Чтобы убедиться в этом, отметим, что в качестве функции от $\boldsymbol{\theta}$ используется $p(\boldsymbol{\theta} | \mathbf{X}) = p(\boldsymbol{\theta}, \mathbf{X})/p(\mathbf{X})$, и поэтому

$$\ln p(\boldsymbol{\theta} | \mathbf{X}) = \ln p(\boldsymbol{\theta}, \mathbf{X}) - \ln p(\mathbf{X}). \quad (15.58)$$

Используя разложение (15.52), получаем

$$\begin{aligned} \ln p(\boldsymbol{\theta} | \mathbf{X}) &= \mathcal{L}(q, \boldsymbol{\theta}) + \text{KL}(q || p) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X}) \\ &\geq \mathcal{L}(q, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X}), \end{aligned} \quad (15.59)$$

где $\ln p(\mathbf{X})$ – это константа. Мы снова можем оптимизировать правую часть поочередно относительно q и $\boldsymbol{\theta}$. Оптимизация по q приводит к тем же уравнениям шага Е, что и для стандартного EM-алгоритма, поскольку q появляется только в $\mathcal{L}(q, \boldsymbol{\theta})$. Уравнения шага М модифицируются за счет введения предварительного члена $\ln p(\boldsymbol{\theta})$, что обычно требует лишь небольшого изменения стандартных уравнений шага М для максимального правдоподобия (см. главу 9). Дополнительный член представляет собой форму регуляризации и устраняет сингулярности функции правдоподобия для моделей гауссовой смеси.

15.4.4. Обобщенный EM

В EM-алгоритме потенциально сложная задача максимизации функции правдоподобия разбивается на два этапа – шаг Е и шаг М, каждый из которых часто оказывается более простым в реализации. Тем не менее для сложных моделей может оказаться, что либо шаг Е, либо шаг М, а то и оба остаются неразрешимыми. Это приводит к двум возможным расширениям EM-алгоритма.

Обобщенный EM-алгоритм (Generalized EM, GEM) решает проблему трудновыполнимого М-шага. Вместо стремления максимизировать $\mathcal{L}(q, \boldsymbol{\theta})$ по отношению к $\boldsymbol{\theta}$ он стремится изменить параметры таким образом, чтобы увеличить их значение. Поскольку $\mathcal{L}(q, \boldsymbol{\theta})$ – это нижняя граница функции логарифмического правдоподобия, каждый полный EM-цикл алгоритма GEM гарантированно увеличивает значение логарифмического правдоподобия (если только параметры уже не соответствуют локальному максимуму). Одним из способов применения подхода GEM является использование градиентных итерационных алгоритмов оптимизации на шаге М. Другая форма алгоритма GEM, известная как *алгоритм условной максимизации ожиданий (expectation conditional maximization algorithm)*, предполагает выполнение нескольких ограниченных оптимизаций на каждом шаге М (Meng and Rubin, 1993). Например, параметры могут быть разделены на группы, а шаг М разбит на несколько шагов, каждый из которых включает оптимизацию одной из групп с фиксированным остатком.

Аналогичным образом можно обобщить шаг Е алгоритма EM, выполнив не полную, а частичную оптимизацию $\mathcal{L}(q, \boldsymbol{\theta})$ по отношению к $q(\mathbf{Z})$ (Neal and Hinton, 1999). Как уже было замечено, для любого заданного значения $\boldsymbol{\theta}$ существует единственный максимум $\mathcal{L}(q, \boldsymbol{\theta})$ относительно $q(\mathbf{Z})$, соответствующего

апостериорному распределению $q_{\theta}(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$, и что для этого выбора $q(\mathbf{Z})$ граница $\mathcal{L}(q, \boldsymbol{\theta})$ равна функции логарифмического правдоподобия $\ln p(\mathbf{X} | \boldsymbol{\theta})$. Отсюда следует, что любой алгоритм, сходящийся к глобальному максимуму $\mathcal{L}(q, \boldsymbol{\theta})$, сможет найти значение $\boldsymbol{\theta}$, которое также является глобальным максимумом логарифмического правдоподобия $\ln p(\mathbf{X} | \boldsymbol{\theta})$. Если $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ является непрерывной функцией $\boldsymbol{\theta}$, то в силу непрерывности любой локальный максимум $\mathcal{L}(q, \boldsymbol{\theta})$ будет также локальным максимумом $\ln p(\mathbf{X} | \boldsymbol{\theta})$.

15.4.5. Последовательный ЕМ

Рассмотрим N независимых точек данных $\mathbf{x}_1, \dots, \mathbf{x}_N$ с соответствующими латентными переменными $\mathbf{z}_1, \dots, \mathbf{z}_N$. Совместное распределение $p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$ факторизуется по точкам данных, и эта структура может быть использована в инкрементальной форме ЕМ, где на каждом ЕМ-цикле обрабатывается только одна точка данных за раз. На шаге Е вместо того, чтобы пересчитывать обязанности для всех точек данных, достаточно переоценить обязанности для одной точки данных. Может показаться, что последующий шаг М потребует вычислений, включающих ответственность за все точки данных. Однако если компоненты смеси являются членами экспоненциального семейства, то ответственность вводится только через простые достаточные статистики, которые можно эффективно обновлять. Рассмотрим, например, гауссову смесь и предположим, что для точки данных m производится обновление, при котором соответствующие старые и новые значения обязанностей обозначаются $\gamma^{\text{old}}(z_{mk})$ и $\gamma^{\text{new}}(z_{mk})$. На шаге М необходимые достаточные статистики могут обновляться инкрементным образом. Например, для средних значений достаточная статистика (см. упражнение 15.23) определяется с помощью (15.16) и (15.17), из которых получаем

$$\boldsymbol{\mu}_k^{\text{new}} = \boldsymbol{\mu}_k^{\text{old}} + \left(\frac{\gamma^{\text{new}}(z_{mk}) - \gamma^{\text{old}}(z_{mk})}{N_k^{\text{new}}} \right) (\mathbf{x}_m - \boldsymbol{\mu}_k^{\text{old}}) \quad (15.60)$$

вместе с

$$N_k^{\text{new}} = N_k^{\text{old}} + \gamma^{\text{new}}(z_{mk}) - \gamma^{\text{old}}(z_{mk}). \quad (15.61)$$

Соответствующие результаты для ковариаций и коэффициентов смешивания выглядят аналогично.

Таким образом, и шаг Е, и шаг М занимают фиксированное время, которое не зависит от общего количества точек данных. Поскольку параметры пересматриваются после каждой точки данных, а не после обработки всего набора данных, эта инкрементальная версия может сходиться быстрее, чем пакетная. Каждый шаг Е или М в этом инкрементальном алгоритме увеличивает значение $\mathcal{L}(q, \boldsymbol{\theta})$, и, как было показано выше, если алгоритм сходится к локальному (или глобальному) максимуму $\mathcal{L}(q, \boldsymbol{\theta})$, то это будет соответствовать локальному (или глобальному) максимуму функции логарифмического правдоподобия $\ln p(\mathbf{X} | \boldsymbol{\theta})$.

Упражнения

- 15.1** (*) Рассмотрим алгоритм K -средних, обсуждавшийся в разделе 15.1. Докажите, что вследствие того, что существует конечное число возможных назначений для набора дискретных индикаторных переменных r_{nk} и что для каждого такого назначения существует единственный оптимум для $\{\mu_k\}$, алгоритм K -средних должен сходиться после конечного числа итераций.
- 15.2** (**) В этом упражнении выводится последовательная форма для алгоритма K -средних. На каждом шаге рассматривается новая точка данных \mathbf{x}_n и обновляется только тот вектор-прототип, который ближе всего к \mathbf{x}_n . Начиная с выражения (15.4) для векторов-прототипов в пакетной постановке, выделите вклад от конечной точки данных \mathbf{x}_n . Переставив формулу, докажите, что это обновление имеет вид (15.5). Заметим, что, поскольку в этом выводе не делается никаких приближений, результатирующие векторы-прототипы будут обладать тем свойством, что каждый из них будет равен среднему значению всех векторов данных, которые были им присвоены.
- 15.3** (*) Рассмотрим модель гауссовой смеси, в которой маргинальное распределение $p(\mathbf{z})$ для латентной переменной задано в (15.9), а условное распределение $p(\mathbf{x}|\mathbf{z})$ для наблюдаемой переменной задано в (15.10). Докажите, что маргинальное распределение $p(\mathbf{x})$, полученное суммированием $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ по всем возможным значениям \mathbf{z} , представляет собой гауссову смесь вида (15.6).
- 15.4** (*) Докажите, что число эквивалентных настроек параметров, обусловленных симметриями чередования, в модели смеси с K компонентами равно $K!$.
- 15.5** (**) Предположим, необходимо использовать алгоритм ЕМ для максимизации апостериорного распределения по параметрам $p(\boldsymbol{\theta}|\mathbf{X})$ для модели, содержащей латентные переменные, где \mathbf{X} – это набор наблюдаемых данных. Докажите, что шаг Е остается таким же, как и в случае максимального правдоподобия, тогда как на шаге М величина, которую нужно максимизировать, задается как $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) + \ln p(\boldsymbol{\theta})$, где $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ определяется в (15.23).
- 15.6** (*) Рассмотрим направленный граф для модели гауссовой смеси, показанный на рис. 15.9. Используя критерий d -разделения (см. раздел 11.2), докажите, что апостериорное распределение латентных переменных факторизуется относительно различных точек данных так, что

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \prod_{n=1}^N p(\mathbf{z}_n|\mathbf{x}_n, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}). \quad (15.62)$$

- 15.7** (**) Рассмотрим частный случай модели гауссовой смеси, в которой ковариационные матрицы Σ_k компонентов ограничены общим значением Σ . Выведите ЕМ-уравнения для максимизации функции правдоподобия в такой модели.
- 15.8** (**) Докажите, что максимизация логарифмического правдоподобия (15.26) для модели гауссовой смеси с полным набором данных приводит к тому, что средние значения и ковариации каждой компоненты независимо подходят к соответствующей группе точек данных, а коэффициенты смещивания определяются долями точек в каждой группе.
- 15.9** (**) Докажите, что при максимизации (15.30) по μ_k при фиксированных обязанностях $\gamma(z_{nk})$ получится решение в замкнутой форме, заданное в (15.16).
- 15.10** (**) Докажите, что при максимизации (15.30) по Σ_k и π_k при фиксированных обязанностях $\gamma(z_{nk})$ получаются решения в замкнутой форме, представленные в (15.18) и (15.21).
- 15.11** (**) Рассмотрим модель плотности, заданную распределением смеси

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k p(\mathbf{x}|k), \quad (15.63)$$

и предположим, что вектор \mathbf{x} разбит на две части так, что $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$. Докажите, что условная плотность $p(\mathbf{x}_b | \mathbf{x}_a)$ сама является распределением смеси, и найдите выражения для коэффициентов смещивания и плотностей компонентов.

- 15.12** (*) В разделе 15.3.2 была определена связь между K -средними и ЕМ для гауссовых смесей на примере модели смеси, где все компоненты имеют ковариацию $\epsilon \mathbf{I}$. Докажите, что в пределе $\epsilon \rightarrow 0$ максимизация ожидаемого логарифмического правдоподобия для этой модели, заданной в (15.30), эквивалентна минимизации меры ошибки J для алгоритма K -средних, заданной в (15.1).
- 15.13** (**) Подтвердите результаты (15.35) и (15.36) для среднего значения и ковариации распределения Бернулли.
- 15.14** (**) Рассмотрим смешанное распределение вида

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k p(\mathbf{x}|k), \quad (15.63)$$

где элементы \mathbf{x} могут быть дискретными, непрерывными или их комбинацией. Обозначим среднее значение и ковариацию $p(\mathbf{x}|k)$ через μ_k и Σ_k соответственно. Используя результаты упражнения 15.13, докажите, что среднее значение и ковариация распределения смеси определяются по (15.39) и (15.40).

- 15.15** (**) Используя уравнения переоценки для ЕМ-алгоритма, докажите, что смесь распределений Бернулли, параметры которой установлены на значения по максимуму функции правдоподобия, обладает свойством:

$$\mathbb{E}[\mathbf{x}] = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \equiv \bar{\mathbf{x}}. \quad (15.65)$$

Отсюда следует, что если параметры этой модели инициализированы таким образом, что все компоненты имеют одинаковое среднее значение $\boldsymbol{\mu}_k = \hat{\boldsymbol{\mu}}$ для $k = 1, \dots, K$, то ЕМ-алгоритм сходится после одной итерации при любом выборе начальных коэффициентов смешивания, и это решение обладает свойством $\boldsymbol{\mu}_k = \bar{\mathbf{x}}$. Обратите внимание, что это вырожденный случай модели смеси, где все компоненты одинаковы, и на практике таких решений рекомендуется избегать с помощью соответствующей инициализации.

- 15.16** (*) Рассмотрим совместное распределение латентных и наблюдаемых переменных для распределения Бернулли, полученного путем произведения $p(\mathbf{x} | \mathbf{z}, \boldsymbol{\mu})$, заданного в (15.42), и $p(\mathbf{z} | \boldsymbol{\pi})$, заданного в (15.43). Докажите, что в случае маргинализации этого совместного распределения по \mathbf{z} получается (15.37).
- 15.17** (*) Докажите, что в случае максимизации ожидаемой функции правдоподобия (15.45) для смеси распределений Бернулли с полными данными относительно $\boldsymbol{\mu}_k$ получится уравнение шага М (15.49).
- 15.18** (*) Докажите, что в случае максимизации ожидаемой функции правдоподобия (15.45) для смеси распределений Бернулли по коэффициентам смешивания π_k и использования множителя Лагранжа для обеспечения ограничения на суммирование получается уравнение шага М (15.50).
- 15.19** (**) Докажите, что в результате ограничения $0 \leq p(\mathbf{x}_n | \boldsymbol{\mu}_k) \leq 1$ для дискретной переменной \mathbf{x}_n функция правдоподобия для смеси распределений Бернулли с неполными данными ограничена сверху и, следовательно, не существует сингулярностей, для которых правдоподобие уходит в бесконечность.
- 15.20** (***) Рассмотрим D -мерную переменную \mathbf{x} , каждая из компонентов которой i сама является мультиномиальной переменной в степени M , так что \mathbf{x} – это бинарный вектор с компонентами x_{ij} , где $i = 1, \dots, D$ и $j = 1, \dots, M$, при условии, что $\sum_j x_{ij} = 1$ для всех i . Предположим, что распределение этих переменных описывается смесью дискретных мультиномиальных распределений (см. раздел 3.1.3), так что

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k p(\mathbf{x} | \boldsymbol{\mu}_k), \quad (15.66)$$

где

$$p(\mathbf{x}|\boldsymbol{\mu}_k) = \prod_{i=1}^D \prod_{j=1}^M \mu_{kij}^{x_{ij}}. \quad (15.67)$$

Параметры μ_{kij} представляют собой вероятности $p(x_{ij} = 1 | \boldsymbol{\mu}_k)$ и должны удовлетворять $0 \leq \mu_{kij} \leq 1$ вместе с ограничением $\sum_j \mu_{kij} = 1$ для всех значений k и i . Учитывая наблюдаемый набор данных $\{\mathbf{x}_n\}$, где $n = 1, \dots, N$, выведите уравнения шага Е и шага М работы EM-алгоритма для оптимизации коэффициентов смешивания π_k и параметров компонент μ_{kij} этого распределения методом максимального правдоподобия.

- 15.21** (*) Убедитесь в справедливости соотношения (15.52), где $\mathcal{L}(q, \boldsymbol{\theta})$ и $\text{KL}(q || p)$ определяются соответственно в (15.53) и (15.54).
- 15.22** (*) Докажите, что нижняя граница $\mathcal{L}(q, \boldsymbol{\theta})$, заданная в (15.53), при $q(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{(\text{old})})$ имеет тот же градиент относительно $\boldsymbol{\theta}$, что и функция логарифмического правдоподобия $\ln p(\mathbf{X} | \boldsymbol{\theta})$ в точке $\boldsymbol{\theta} = \boldsymbol{\theta}^{(\text{old})}$.
- 15.23** (**) Рассмотрим инкрементную форму EM-алгоритма для гауссовой смеси, в которой обязанности пересчитываются только для конкретной точки данных \mathbf{x}_m . Начиная с формул для шага М (15.16) и (15.17), выведите результаты (15.60) и (15.61) для обновления значений компонент средних значений.
- 15.24** (**) Выполните формулы шага М для обновления ковариационных матриц и коэффициентов смешивания в модели гауссовой смеси, когда обязанности обновляются инкрементным способом, по аналогии с результатом (15.60) для обновления средних значений.

Глава 16

Непрерывные латентные переменные

Темой предыдущей главы было обсуждение вероятностных моделей с дискретными латентными переменными, таких как гауссовые смеси. Теперь перейдем к обсуждению моделей, в которых часть или все латентные переменные являются непрерывными. Важным аргументом в пользу создания таких моделей (см. раздел 6.1.3) является свойство многих наборов данных, когда точки данных лежат вблизи многообразия с гораздо меньшей размерностью, чем размерность исходного пространства данных. В качестве примера можно привести искусственный набор данных, созданный путем извлечения рукописной цифры из набора данных MNIST (LeCun et al., 1998), представленной в виде полутонового изображения размером 64×64 пикселя, и встраивания ее в большее изображение размером 100×100 путем добавления пикселей с нулевым значением (соответствующих белым пикселям), где расположение и ориентация цифры меняются случайным образом, как показано на рис. 16.1. Каждое из полученных изображений представлено точкой в $100 \times 100 = 10\,000$ -мерном пространстве данных. Однако в наборе данных таких изображений есть только *три степени свободы* для изменения, соответствующие вертикальному и горизонтальному перемещениям, а также вращениям. Поэтому точки данных будут находиться в подпространстве пространства данных, *собственная размерность* которого равна трем. Обратите внимание, что это многообразие будет нелинейным, поскольку, например, если переместить цифру мимо определенного пикселя, значение этого пикселя перейдет от нуля (белого) к единице (черному) и снова к нулю, что, очевидно, является нелинейной функцией положения цифры. В этом примере параметры перемещения и поворота являются латентными переменными, поскольку в поле зрения находятся только векторы изображений, а значения переменных перемещения и поворота не определены.

Для реальных наборов данных рукописных цифр существуют дополнительные степени свободы, обусловленные масштабированием и другими вариациями, например изменчивостью почерка человека, а также различиями в стилях письма разных людей. Тем не менее количество таких степеней свободы будет невелико по сравнению с размерностью набора данных.



РИС. 16.1 Синтетический набор данных из изображения рукописной цифры и нескольких копий, где цифра подверглась случайному перемещению и вращению в некотором большом поле изображения. Каждое из полученных изображений имеет размер $100 \times 100 = 10000$ пикселей

На практике точки данных не будут четко ограничены гладким многообразием низкой размерности, и отклонения точек данных от многообразия можно интерпретировать в качестве шума. Это естественным образом приводит к формированию генеративной концепции таких моделей, где вначале выбирается точка внутри многообразия в соответствии с некоторым распределением латентных переменных, а затем генерируется наблюдаемая точка данных путем добавления шума, взятого из некоторого условного распределения переменных данных с учетом латентных переменных.

Простейшая непрерывная модель латентных переменных предполагает гауссовые распределения для латентных и наблюдаемых переменных при использовании линейной гауссовой зависимости (см. раздел 11.1.4) наблюдаемых переменных от состояния латентных переменных. Это приводит к вероятностной формулировке хорошо известной техники анализа главных компонентов (*principal component analysis, PCA*), а также к родственной модели под названием *факторный анализ* (*factor analysis*). В этой главе (см. раздел 16.1) сначала дается стандартная, не вероятностная трактовка PCA, а затем рассказывается о том, как PCA становится естественным решением максимального правдоподобия для линейной гауссовой модели латентных переменных (см. раздел 16.2). Эта вероятностная формулировка дает множество преимуществ, таких как использование EM для оценки параметров, принципиальные расширения для смесей моделей PCA и байесовские формулировки для автоматического определения числа главных компонентов по данным (Bishop, 2006). В этой главе также заложены основы анализа нелинейных моделей с непрерывными латентными переменными, включая нормализующие потоки, вариационные автокодировщики и диффузионные модели.

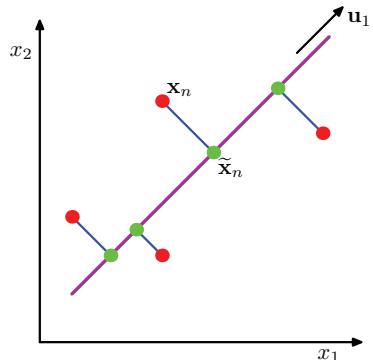
16.1. Анализ главных компонентов

Анализ главных компонентов (PCA) широко используется в таких задачах, как уменьшение размерности, сжатие данных с потерями, извлечение признаков и визуализация данных (Jolliffe, 2002). Его также называют преобразованием Косамби–Карунена–Лоэва (*Kosambi–Karhunen–Loëve transform*).

Рассмотрим ортогональную проекцию набора данных на линейное пространство меньшей размерности, называемое *главным подпространством*

(*principal subspace*), как показано на рис. 16.2. Можно назвать метод РСА линейной проекцией, которая максимизирует дисперсию проецируемых данных (Hotelling, 1933). Аналогичным образом РСА можно назвать линейной проекцией, которая минимизирует среднюю стоимость проекции, определяемую как среднее квадратичное расстояние между точками данных и их проекциями (Pearson, 1901). Рассмотрим каждое из этих определений по очереди.

РИС. 16.2 Анализ главных компонентов определяет пространство меньшей размерности (главное подпространство, пурпурная линия) таким образом, чтобы ортогональная проекция точек данных (красные точки) на это подпространство максимизировала дисперсию спроектированных точек (зеленые точки). Альтернативное определение РСА основано на минимизации суммы квадратов ошибок проецирования (синие линии)



16.1.1. Определение максимальной дисперсии

Рассмотрим набор данных наблюдений $\{\mathbf{x}_n\}$, где $n = 1, \dots, N$, а \mathbf{x}_n – это евклидова переменная с размерностью D . Задача заключается в том, чтобы спроектировать данные на пространство с размерностью $M < D$, максимизируя при этом дисперсию спроектированных данных. Пока что предположим, что значение M известно. Позже в этой главе будут рассмотрены методы определения подходящего значения M по данным.

Для начала рассмотрим проекцию на одномерное пространство ($M = 1$). Направление этого пространства можно определить с помощью D -мерного вектора \mathbf{u}_1 , который для удобства (и без потери общности) выберем единичным вектором, так что $\mathbf{u}_1^T \mathbf{u}_1 = 1$ (обратите внимание, что в данном случае интерес представляет только направление, определяемое \mathbf{u}_1 , а не величина самого \mathbf{u}_1). Затем каждая точка данных \mathbf{x}_n проецируется на скалярное значение $\mathbf{u}_1^T \mathbf{x}_n$. Среднее значение спроектированных данных равно $\mathbf{u}_1^T \bar{\mathbf{x}}$, где $\bar{\mathbf{x}}$ – это среднее значение выборочной совокупности, определяемое как

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n, \quad (16.1)$$

а дисперсия проецируемых данных определяется как

$$\frac{1}{N} \sum_{n=1}^N \{\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}}\}^2 = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1, \quad (16.2)$$

где \mathbf{S} – это ковариационная матрица данных, определяемая как

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T. \quad (16.3)$$

Теперь максимизируем прогнозируемую дисперсию $\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$ относительно \mathbf{u}_1 . Очевидно, что это должна быть ограниченная максимизация, чтобы не допустить $\|\mathbf{u}_1\| \rightarrow \infty$. Соответствующее ограничение вытекает из условия нормализации $\mathbf{u}_1^T \mathbf{u}_1 = 1$. Чтобы обеспечить это ограничение, необходимо ввести множитель Лагранжа (см. приложение C), который обозначим λ_1 , а затем выполнить максимизацию без ограничений в виде

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^T \mathbf{u}_1). \quad (16.4)$$

Установив производную по \mathbf{u}_1 равной нулю, получим, что эта величина будет иметь стационарную точку при

$$\mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1, \quad (16.5)$$

что говорит о том, что \mathbf{u}_1 должен быть собственным вектором \mathbf{S} . Если умножить левое значение на \mathbf{u}_1^T и использовать $\mathbf{u}_1^T \mathbf{u}_1 = 1$, то дисперсия будет иметь вид

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1, \quad (16.6)$$

и поэтому дисперсия будет максимальной при установке \mathbf{u}_1 равным собственному вектору с наибольшим собственным значением λ_1 . Этот собственный вектор известен как первый основной компонент.

Дополнительные основные компоненты можно определять постепенно, выбирая для каждого нового направления то, которое максимизирует дисперсию прогноза среди всех возможных направлений, ортогональных к уже рассмотренным. Если рассматривать общий случай M -мерного проекционного пространства, то оптимальная линейная проекция, для которой дисперсия проецируемых данных максимальна, теперь определяется M собственными векторами $\mathbf{u}_1, \dots, \mathbf{u}_M$ ковариационной матрицы данных \mathbf{S} , соответствующей M наибольшим собственным значениям $\lambda_1, \dots, \lambda_M$ (см. упражнение 16.1). Это легко доказать с помощью метода индукции.

Подводя итог, можно сказать, что РСА предполагает оценку среднего значения $\bar{\mathbf{x}}$ и ковариационной матрицы \mathbf{S} набора данных, а затем определение M собственных векторов \mathbf{S} , соответствующих M наибольшим собственным значениям. Алгоритмы определения собственных векторов и собственных значений, а также дополнительные теоремы, связанные с разложением по собственным векторам, можно найти в работе (Golub and Van Loan, 1996). Обратите внимание, что вычислительные затраты на полное разложение собственных векторов для матрицы размера $D \times D$ составляют $\mathcal{O}(D^3)$. Если предполагается спроектировать полученные данные на первые M основных компонентов, то потребуется найти только первые M собственных значений и собственных векторов. Это можно сделать с помощью более эффективных методов, таких как *степенной метод* (*power method*) (Golub and Van Loan,

1996), который масштабируется до $\mathcal{O}(MD^2)$, или же можно воспользоваться ЕМ-алгоритмом (см. раздел 16.3.2).

16.1.2. Определение минимальной ошибки

Теперь рассмотрим альтернативную формулировку PCA, основанную на минимизации ошибки проекции (см. приложение А). Для этого введем полный ортонормальный набор D -мерных базисных векторов $\{\mathbf{u}_i\}$, где $i = 1, \dots, D$, которые соответствуют условию

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}. \quad (16.7)$$

Поскольку этот базис является полным, каждая точка данных может быть точно представлена линейной комбинацией базисных векторов

$$\mathbf{x}_n = \sum_{i=1}^D \alpha_{ni} \mathbf{u}_i, \quad (16.8)$$

где коэффициенты α_{ni} будут разными для разных точек данных. Это всего лишь соответствует повороту системы координат до новой системы, определяемой $\{\mathbf{u}_i\}$, а исходные компоненты $D \{x_{n1}, \dots, x_{nD}\}$ заменяются эквивалентным набором $\{\alpha_{n1}, \dots, \alpha_{nD}\}$. Если взять внутреннее произведение с \mathbf{u}_j и воспользоваться свойством ортонормированности, получится $\alpha_{nj} = \mathbf{x}_n^T \mathbf{u}_j$, и поэтому можно без потери обобщенности записать:

$$\mathbf{x}_n = \sum_{i=1}^D (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i. \quad (16.9)$$

Целью исследования тем не менее является аппроксимация этой точки данных с помощью представления, включающего ограниченное число $M < D$ переменных, соответствующих проекции на подпространство меньшей размерности. Линейное подпространство размерности M без потери обобщения может быть представлено первыми из M базисных векторов, и поэтому каждую точку данных \mathbf{x}_n можно аппроксимировать следующим образом:

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^M z_{ni} \mathbf{u}_i + \sum_{i=M+1}^D b_i \mathbf{u}_i. \quad (16.10)$$

Здесь $\{z_{ni}\}$ зависит от конкретной точки данных, а $\{b_i\}$ – это константы, одинаковые для всех точек данных. Можно произвольно выбирать $\{\mathbf{u}_i\}$, $\{z_{ni}\}$ и $\{b_i\}$ так, чтобы минимизировать ошибку, вносимую уменьшением размерности. В качестве меры ошибки будет использоваться квадратичное расстояние между исходной точкой данных \mathbf{x}_n и ее аппроксимацией $\tilde{\mathbf{x}}_n$, усредненное по всему набору данных, так что задача состоит в минимизации:

$$J = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2. \quad (16.11)$$

Сначала рассмотрим минимизацию относительно величин $\{z_{nj}\}$. Подставляя $\tilde{\mathbf{x}}_n$, устанавливая производную по z_{nj} равной нулю и используя условия ортонормированности, получаем

$$z_{nj} = \mathbf{x}_n^T \mathbf{u}_j, \quad (16.12)$$

где $j = 1, \dots, M$. Аналогичным образом, обратив производную J по b_i в ноль и снова воспользовавшись соотношениями ортонормированности, получим

$$b_j = \bar{\mathbf{x}}^T \mathbf{u}_j, \quad (16.13)$$

где $j = M + 1, \dots, D$. Подставляя z_{ni} и b_i и используя общее разложение (16.9), получаем

$$\mathbf{x}_n - \tilde{\mathbf{x}}_n = \sum_{i=M+1}^D \{(\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{u}_i\} \mathbf{u}_i, \quad (16.14)$$

откуда видно, что вектор смещения от \mathbf{x}_n до $\tilde{\mathbf{x}}_n$ лежит в пространстве, ортогональном главному подпространству, поскольку является линейной комбинацией $\{\mathbf{u}_i\}$ для $i = M + 1, \dots, D$, как показано на рис. 16.2. Этого следовало ожидать, поскольку проецируемые точки $\tilde{\mathbf{x}}_n$ должны лежать в главном подпространстве, но при этом их можно свободно перемещать в пределах этого подпространства, и поэтому минимальная ошибка дается ортогональной проекцией.

Таким образом, получаем выражение для меры ошибки J как функции исключительно от $\{\mathbf{u}_i\}$ в виде

$$J = \frac{1}{N} \sum_{n=1}^N \sum_{i=M+1}^D (\mathbf{x}_n^T \mathbf{u}_i - \bar{\mathbf{x}}^T \mathbf{u}_i)^2 = \sum_{i=M+1}^D \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i. \quad (16.15)$$

Остается задача минимизации J относительно $\{\mathbf{u}_i\}$, которая должна быть минимизацией с ограничениями, иначе получится бессодержательный результат $\mathbf{u}_i = 0$. Ограничения вытекают из условий ортонормированности, и, как будет видно далее, решение будет выражено в виде разложения ковариационной матрицы по собственным векторам. Прежде чем рассматривать формальное решение, попробуем получить некоторое представление о результате, рассмотрев двумерное пространство данных $D = 2$ и одномерное главное подпространство $M = 1$. Необходимо выбрать направление \mathbf{u}_2 так, чтобы минимизировать $J = \mathbf{u}_2^T \mathbf{S} \mathbf{u}_2$ при условии ограничения нормализации $\mathbf{u}_2^T \mathbf{u}_2 = 1$. Используя множитель Лагранжа λ_2 для обеспечения ограничения, можно определить, что минимизация

$$\hat{J} = \mathbf{u}_2^T \mathbf{S} \mathbf{u}_2 + \lambda_2 (1 - \mathbf{u}_2^T \mathbf{u}_2). \quad (16.16)$$

Обратив производную по \mathbf{u}_2 в ноль, получаем $\mathbf{S} \mathbf{u}_2 = \lambda_2 \mathbf{u}_2$, так что \mathbf{u}_2 – это собственный вектор \mathbf{S} с собственным значением λ_2 . Получается, что любой собственный вектор будет определять стационарную точку меры ошибки.

Чтобы найти значение J в минимуме, подставим решение для \mathbf{u}_2 в меру ошибок и получим $J = \lambda_2$. Таким образом, для получения минимального значения J необходимо выбрать \mathbf{u}_2 как собственный вектор, соответствующий меньшему из двух собственных значений. Следовательно, необходимо выбрать главное подпространство, чтобы оно было выровнено по собственному вектору с *наибольшим* собственным значением. Этот результат согласуется с предположением, что для минимизации среднего квадратичного расстояния проекции следует выбирать подпространство главных компонент так, чтобы оно проходило через среднее значение точек данных и было выровнено с направлениями максимальной дисперсии. Если собственные значения равны, то любой выбор главного направления приведет к одному и тому же значению J .

Общее решение задачи минимизации J для произвольного D и произвольного $M < D$ (см. упражнение 16.2) получается путем выбора $\{\mathbf{u}_i\}$ в качестве собственных векторов ковариационной матрицы, заданной как

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad (16.17)$$

где $i = 1, \dots, D$, и, как обычно, собственные векторы $\{\mathbf{u}_i\}$ выбираются ортонормальными. Соответствующее значение меры ошибки определяется как

$$J = \sum_{i=M+1}^D \lambda_i, \quad (16.18)$$

которая является просто суммой собственных значений тех собственных векторов, которые ортогональны главному подпространству. Поэтому для получения минимального значения J выбираются собственные векторы, имеющие $D - M$ наименьших собственных значений, и, следовательно, собственные векторы, определяющие главное подпространство, соответствуют M наибольшим собственным значениям.

Хотя в данном случае рассматривалось условие $M < D$, анализ PCA остается в силе при $M = D$, и в этом случае не происходит уменьшения размерности, а просто осуществляется поворот координатных осей для совмещения с главными компонентами.

Наконец, обратите внимание, что существует родственный метод линейного сокращения размерности, называемый *анализом канонических корреляций* (*canonical correlation analysis*) (Hotelling, 1936; Bach and Jordan, 2002). В то время как PCA работает с одной случайной переменной, анализ канонических корреляций рассматривает две (или более) переменные и стремится найти соответствующую пару линейных подпространств, которые имеют высокую перекрестную корреляцию, так что каждая компонента в одном из подпространств коррелирует с одной компонентой из другого подпространства. Ее решение можно выразить в виде обобщенной задачи собственных векторов.

16.1.3. Сжатие данных

Одним из вариантов применения РСА является сжатие данных, и в качестве иллюстрации можно рассмотреть набор данных из изображений рукописных цифр. Поскольку каждый собственный вектор ковариационной матрицы является вектором в исходном D -мерном пространстве, собственные векторы можно представить в виде изображений того же размера, что и точки данных. Усредненное изображение и первые четыре собственных вектора, а также соответствующие им собственные значения показаны на рис. 16.3.

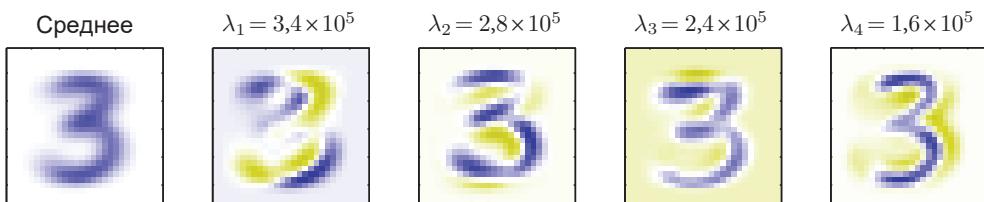


РИС. 16.3 Применение РСА к набору данных из 6000 изображений размером 28×28 , каждое содержит рукописное изображение цифры «3», где показан вектор среднего значения \bar{x} и первые четыре собственных вектора РСА $\mathbf{u}_1, \dots, \mathbf{u}_4$ вместе с их соответствующими собственными значениями

График полного спектра собственных значений, отсортированных в порядке убывания, показан на рис. 16.4а. Мера ошибки J , связанная с выбором конкретного значения M , задается суммой собственных значений от $M + 1$ до D и построена для различных значений M на рис. 16.4б.

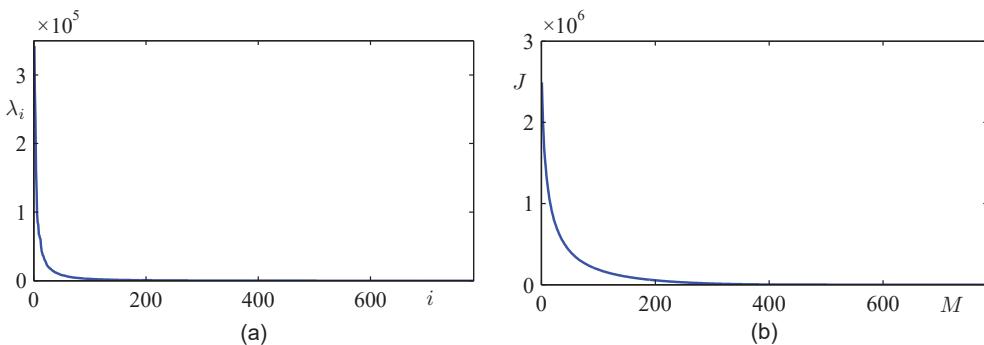


РИС. 16.4 (а) График спектра собственных значений для набора данных рукописных цифр из рис. 16.3. (б) График суммы отброшенных собственных значений, отражающей ошибку суммы квадратов J , вносимую при проецировании данных на подпространство главных компонент размерности M

Если подставить (16.12) и (16.13) в (16.10), то аппроксимацию РСА для вектора данных \mathbf{x}_n можно записать в виде

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^M (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i + \sum_{i=M+1}^D (\bar{\mathbf{x}}^T \mathbf{u}_i) \mathbf{u}_i \quad (16.19)$$

$$= \bar{\mathbf{x}} + \sum_{i=1}^M (\mathbf{x}_n^T \mathbf{u}_i - \bar{\mathbf{x}}^T \mathbf{u}_i) \mathbf{u}_i, \quad (16.20)$$

где используется соотношение

$$\bar{\mathbf{x}} = \sum_{i=1}^D (\bar{\mathbf{x}}^T \mathbf{u}_i) \mathbf{u}_i, \quad (16.21)$$

что следует из полноты $\{\mathbf{u}_i\}$. В результате происходит сжатие набора данных, поскольку для каждой точки данных D -мерный вектор \mathbf{x}_n заменяется M -мерным вектором с компонентами $(\mathbf{x}_n^T \mathbf{u}_i - \bar{\mathbf{x}}^T \mathbf{u}_i)$. Чем меньше значение M , тем больше степень сжатия. Примеры PCA-реконструкции точек данных для набора данных о цифрах показаны на рис. 16.5.

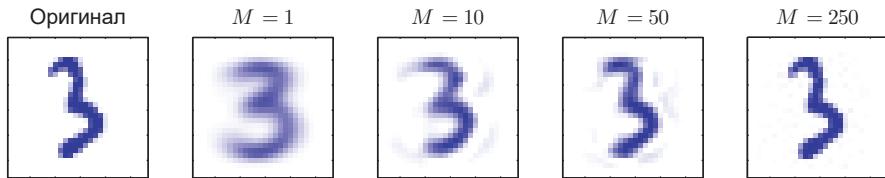


РИС. 16.5 Пример из набора данных рукописных цифр вместе с их PCA-реконструкциями при сохранении M главных компонент для различных значений M . С увеличением M реконструкция становится все более точной и станет идеальной при $M = D = 28 \times 28 = 784$

16.1.4. Отбеливание данных

Другим вариантом применения PCA является предварительная обработка данных. В этом случае целью является не уменьшение размерности, а преобразование набора данных для стандартизации некоторых его свойств. Это может быть важно для успешного применения последующих алгоритмов машинного обучения к набору данных. Обычно это делается в случаях, когда исходные переменные измеряются в разных единицах или имеют значительно отличающиеся вариации. Например, в наборе данных Олд-Фейтфул время между извержениями обычно на порядок больше, чем продолжительность извержения (см. раздел 15.1). Когда к этому набору данных был применен алгоритм K -средних, то сначала было проведено раздельное линейное перескашивание отдельных переменных таким образом, чтобы каждая из них имела нулевое среднее значение и единичную дисперсию. Такой подход называется *стандартизацией* (standardizing) данных, и ковариационная матрица для стандартизованных данных имеет компоненты:

$$\rho_{ij} = \frac{1}{N} \sum_{n=1}^N \frac{(x_{ni} - \bar{x}_i)}{\sigma_i} \frac{(x_{nj} - \bar{x}_j)}{\sigma_j}, \quad (16.22)$$

где σ_i – это стандартное отклонение x_i . Эта матрица известна как *матрица корреляции (correlation matrix)* исходных данных и обладает тем свойством, что если два компонента x_i и x_j данных полностью коррелированы, то $\rho_{ij} = 1$, а если они не коррелированы, то $\rho_{ij} = 0$.

Однако с помощью PCA возможна более существенная нормализация данных с целью придания им нулевого среднего значения и единичной ковариации, в результате чего различные переменные становятся декоррелированными. Для этого сначала запишем уравнение для собственных векторов (16.17) в виде

$$\mathbf{S}\mathbf{U} = \mathbf{U}\mathbf{L}, \quad (16.23)$$

где \mathbf{L} – это диагональная матрица $D \times D$ с элементами λ_i , а \mathbf{U} – это ортогональная матрица $D \times D$ со столбцами, заданными \mathbf{u}_i . Затем для каждой точки данных \mathbf{x}_n мы определяем преобразованное значение, представленное как

$$\mathbf{y}_n = \mathbf{L}^{-1/2} \mathbf{U}^T (\mathbf{x}_n - \bar{\mathbf{x}}), \quad (16.24)$$

где $\bar{\mathbf{x}}$ – это выборочное среднее значение, определяемое в (16.1). Очевидно, что множество $\{\mathbf{y}_n\}$ имеет нулевое среднее значение, а его ковариация задается матрицей тождества, поскольку

$$\begin{aligned} \frac{1}{N} \sum_{n=1}^N \mathbf{y}_n \mathbf{y}_n^T &= \frac{1}{N} \sum_{n=1}^N \mathbf{L}^{-1/2} \mathbf{U}^T (\mathbf{x}_n - \bar{\mathbf{x}}) (\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{U} \mathbf{L}^{-1/2} \\ &= \mathbf{L}^{-1/2} \mathbf{U}^T \mathbf{S} \mathbf{U} \mathbf{L}^{-1/2} = \mathbf{L}^{-1/2} \mathbf{L} \mathbf{L}^{-1/2} = \mathbf{I}. \end{aligned} \quad (16.25)$$

Эта операция известна как *отбеливание (whitening)* или *сферизация (spherizing)* данных. Она проиллюстрирована на рис. 16.6 для набора данных Олд-Фейтфул (см. раздел 15.1). На графике слева показаны исходные данные. На графике в центре показан результат стандартизации отдельных переменных до нулевого среднего значения и единичной дисперсии. Также показаны главные оси этого нормализованного набора данных, построенные в диапазоне $\pm \lambda_i^{1/2}$. График справа показывает результат отбеливания данных для придания им нулевого среднего значения и единичной ковариации.

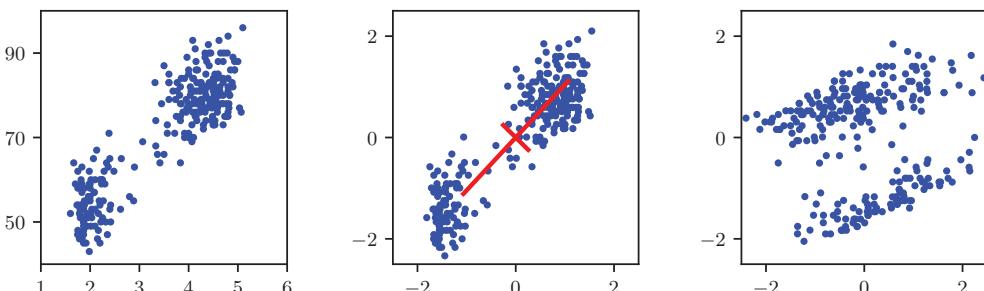


РИС. 16.6 Иллюстрация эффеkтов линейной предварительной обработки набора данных Олд-Фейтфул

16.1.5. Данные высокой размерности

В некоторых случаях применения РСА общее количество точек данных оказывается меньше, чем размерность пространства данных. Например, может понадобиться применить РСА к набору данных из нескольких сотен изображений, каждое из которых соответствует определенному вектору в пространстве с размерностью порядка нескольких миллионов (что соответствует трем значениям цвета для каждого из пикселей изображения). Обратите внимание, что в D -мерном пространстве набор из N точек, где $N < D$, определяет линейное подпространство с размерностью не более $N - 1$, и поэтому нет смысла применять РСА для значений M , которые больше $N - 1$. Действительно, если использовать РСА, то можно обнаружить, что не менее $D - N + 1$ собственных значений равны нулю. Это соответствует собственным векторам, по направлениям которых набор данных имеет нулевую дисперсию. Более того, типичные алгоритмы определения собственных векторов матрицы $D \times D$ требуют вычислительных затрат порядка $\mathcal{O}(D^3)$, поэтому для таких задач, как пример с изображениями, прямое применение РСА будет вычислительно невыполнимым.

Эту проблему можно решить следующим образом. Сначала определим \mathbf{X} как $(N \times D)$ -мерную центрированную матрицу данных, n -я строка которой задается $(\mathbf{x}_n - \bar{\mathbf{x}})^T$. Тогда ковариационная матрица (16.3) может быть записана как $\mathbf{S} = N^{-1}\mathbf{X}^T\mathbf{X}$, а соответствующее уравнение для собственных векторов приобретает вид:

$$\frac{1}{N} \mathbf{X}^T \mathbf{X} \mathbf{u}_i = \lambda_i \mathbf{u}_i. \quad (16.26)$$

Теперь предварительно умножим обе стороны на \mathbf{X} , чтобы получить

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T (\mathbf{X} \mathbf{u}_i) = \lambda_i (\mathbf{X} \mathbf{u}_i). \quad (16.27)$$

Если теперь задать $\mathbf{v}_i = \mathbf{X} \mathbf{u}_i$, то получаем

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T \mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad (16.28)$$

что является уравнением собственных векторов для $N \times N$ матрицы $N^{-1}\mathbf{X} \mathbf{X}^T$. Здесь получается, что она имеет те же $N - 1$ собственных значений, что и исходная ковариационная матрица (которая также имеет дополнительные $D - N + 1$ собственных значений, равных нулю). Таким образом, проблему собственных векторов можно решать в пространствах меньшей размерности с вычислительными затратами $\mathcal{O}(N^3)$ вместо $\mathcal{O}(D^3)$. Чтобы определить собственные векторы, умножим обе стороны (16.28) на \mathbf{X}^T и получим

$$\left(\frac{1}{N} \mathbf{X}^T \mathbf{X} \right) (\mathbf{X}^T \mathbf{v}_i) = \lambda_i (\mathbf{X}^T \mathbf{v}_i), \quad (16.29)$$

из чего следует, что $(\mathbf{X}^T \mathbf{v}_i)$ является собственным вектором \mathbf{S} с собственным значением λ_i . Тем не менее обратите внимание, что эти собственные векторы не обязательно нормализованы. Чтобы определить соответствующую нормализацию, необходимо изменить масштаб $\mathbf{u}_i \propto \mathbf{X}^T \mathbf{v}_i$ на такую константу, чтобы $\|\mathbf{u}_i\| = 1$ (см. упражнение 16.3), в результате чего, при условии, что \mathbf{v}_i нормализован на единицу длины, получается:

$$\mathbf{u}_i = \frac{1}{(N\lambda_i)^{1/2}} \mathbf{X}^T \mathbf{v}_i. \quad (16.30)$$

Получается, что для использования этого подхода следует сначала оценить $\mathbf{X}\mathbf{X}^T$, затем найти его собственные векторы и собственные значения, а затем вычислить собственные векторы в исходном пространстве с использованием (16.30).

16.2. Вероятностные латентные переменные

В предыдущем разделе выяснилось, что РСА можно определить в виде линейной проекции данных на подпространство меньшей размерности, чем исходное пространство данных. Каждая точка данных проецируется на уникальное значение величины z_{nj} , определяемой в (16.12), и эти величины могут рассматриваться как детерминированные латентные переменные. Чтобы представить и мотивировать вероятностные непрерывные латентные переменные, мы покажем, что РСА также может быть выражена в виде решения максимального правдоподобия вероятностной модели латентных переменных. Это новое определение РСА называют *вероятностным (probabilistic) РСА*, который имеет ряд преимуществ по сравнению с обычным РСА.

- Вероятностная модель РСА является ограниченной формой гауссова распределения, в которой число свободных параметров может быть ограничено, но при этом модель по-прежнему способна учитывать доминирующие корреляции в наборе данных.
- Можно вывести ЕМ-алгоритм для РСА, который будет вычислительно эффективным в ситуациях, когда необходимы только несколько ведущих собственных векторов, и который позволяет избежать промежуточного шага для оценки ковариационной матрицы данных (см. раздел 16.3.2).
- Сочетание вероятностной модели и ЕМ-алгоритма позволяет эффективно решать проблему недостающих значений в наборе данных.
- Смеси вероятностных моделей РСА могут быть сформированы и обучены с помощью ЕМ-алгоритма достаточно простым способом.
- Существование функции правдоподобия позволяет напрямую сравнивать их с другими вероятностными моделями распределения плотности. В отличие от этого обычная модель РСА присваивает низкую стоимость

восстановления точкам данных, которые находятся вблизи главного подпространства, даже если они лежат на произвольном расстоянии от обучающих данных.

- Вероятностный PCA можно использовать для моделирования плотностей в зависимости от класса и, следовательно, применять для решения задач классификации.
- Вероятностная модель PCA может использоваться как генеративная модель для получения выборок из распределения.
- Вероятностный PCA является основой для байесовской трактовки PCA, где размерность главного подпространства может быть автоматически определена на основе данных (Bishop, 2006).

Эту формулировку PCA в качестве вероятностной модели независимо друг от друга предложили в своих работах Tipping and Bishop (1997; 1999), а также Roweis (1998). Как будет показано далее, она тесно связана с *факторным анализом* (Basilevsky, 1994).

16.2.1. Генеративная модель

Вероятностный PCA (см. раздел 11.1.4) является простым примером концепции линейного гауссова фреймворка, где все маргинальные и условные распределения являются гауссовыми. Для определения вероятностного PCA сначала необходимо ввести явную M -мерную латентную переменную \mathbf{z} , соответствующую подпространству главных компонентов. Затем для латентной переменной задается гауссово априорное распределение $p(\mathbf{z})$, а для наблюдаемой переменной \mathbf{x} в D -мерном пространстве определяется гауссово условное распределение $p(\mathbf{x} | \mathbf{z})$, зависящее от значения латентной переменной. В частности, априорное распределение по \mathbf{z} задается гауссовым распределением с нулевым средним значением и единичной ковариацией:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}). \quad (16.31)$$

Точно так же условное распределение наблюдаемой переменной \mathbf{x} с учетом значения латентной переменной \mathbf{z} тоже является гауссовым:

$$p(\mathbf{x} | \mathbf{z}) = \mathcal{N}(\mathbf{x} | \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}), \quad (16.32)$$

при котором среднее значение \mathbf{x} является общей линейной функцией \mathbf{z} , управляемой матрицей \mathbf{W} размером $D \times M$ и D -мерным вектором $\boldsymbol{\mu}$. Обратите внимание, что в этом случае факторизация производится относительно элементов \mathbf{x} (см. раздел 11.2.3). Другими словами, это пример наивной модели Байеса. Как вскоре будет показано, столбцы \mathbf{W} охватывают линейное подпространство в пространстве данных, которое соответствует главному подпространству. Другим параметром в этой модели является скаляр σ^2 , определяющий дисперсию условного распределения. Следует отметить, что допущение нулевого среднего значения с единичной ковариацией гауссианы

для скрытого распределения $p(\mathbf{z})$ не приводит к потере общности, поскольку более общее гауссово распределение привело бы к эквивалентной вероятностной модели (см. упражнение 16.4).

Вероятностная модель PCA может рассматриваться как генеративная модель, где выборочное значение наблюдаемой переменной определяется путем выбора значения латентной переменной, а затем выборки наблюдаемой переменной в зависимости от этого значения латентной переменной. В частности, D -мерная наблюдаемая переменная \mathbf{x} определяется линейным преобразованием M -мерной латентной переменной \mathbf{z} плюс аддитивный гауссов шум, так что

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \quad (16.33)$$

где \mathbf{z} – это M -мерная гауссова латентная переменная, а $\boldsymbol{\epsilon}$ – это D -мерная шумовая переменная с нулевым средним гауссовым распределением и ковариацией $\sigma^2\mathbf{I}$. Этот генеративный процесс показан на рис. 16.7. Наблюдаемая точка данных \mathbf{x} генерируется сначала путем извлечения значения $\hat{\mathbf{z}}$ для латентной переменной из ее априорного распределения $p(\mathbf{z})$, а затем путем извлечения значения \mathbf{x} из изотропного гауссова распределения (показано красными кружками) со средним значением $\mathbf{w}\hat{\mathbf{z}} + \boldsymbol{\mu}$ и ковариацией $\sigma^2\mathbf{I}$. Зеленые эллипсы показывают контуры плотности для маргинального распределения $p(\mathbf{x})$. Обратите внимание, что эта схема основана на отображении из латентного пространства в пространство данных, в отличие от рассмотренного ранее более традиционного представления PCA. Ниже будет приведена обратная схема отображения из пространства данных в латентное пространство с помощью теоремы Байеса.

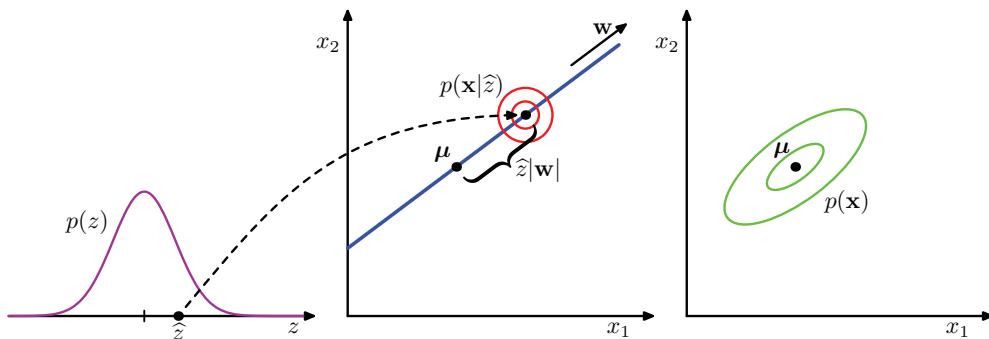


РИС. 16.7 Иллюстрация генеративного представления вероятностной модели PCA для двумерного пространства данных и одномерного латентного пространства

16.2.2. Функция правдоподобия

Предположим, необходимо определить значения параметров \mathbf{W} , $\boldsymbol{\mu}$ и σ^2 с помощью метода максимального правдоподобия. Чтобы записать функцию правдоподобия, понадобится выражение для маргинального распределения

$p(\mathbf{x})$ наблюдаемой переменной. Оно определяется из правил суммы и произведения вероятностей в виде

$$p(\mathbf{x}) = \int p(\mathbf{x} | \mathbf{z}) p(\mathbf{z}) d\mathbf{z}. \quad (16.34)$$

Поскольку это соответствует линейной гауссовой модели, маргинальное распределение тоже является гауссовым (см. упражнение 16.6) и имеет вид

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \mathbf{C}), \quad (16.35)$$

где ковариационная матрица \mathbf{C} размером $D \times D$ определяется как

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}. \quad (16.36)$$

Этот результат можно получить и более прямым путем, если отметить, что прогнозируемое распределение будет гауссовым, а затем оценить его среднее значение и ковариацию с помощью (16.33). Это дает

$$\mathbb{E}[\mathbf{x}] = \mathbb{E}[\mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}] = \boldsymbol{\mu}, \quad (16.37)$$

$$\begin{aligned} \text{cov}[\mathbf{x}] &= \mathbb{E}[(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})^T] \\ &= \mathbb{E}[\mathbf{W}\mathbf{z}\mathbf{z}^T\mathbf{W}^T] + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] \end{aligned} \quad (16.38)$$

$$= \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}, \quad (16.39)$$

где для расчета учтен тот факт, что \mathbf{z} и $\boldsymbol{\epsilon}$ являются независимыми случайными величинами и, следовательно, не имеют корреляции.

На первый взгляд, распределение $p(\mathbf{x})$ задается изотропным гауссовым «распылителем», который перемещается по главному подпространству и располяет «гауссовых чернила» с плотностью, определяемой σ^2 и взвешенной по априорному распределению. Накопленная плотность чернил приводит к распределению в форме «блинчика» с маргинальной плотностью $p(\mathbf{x})$.

Прогнозное распределение $p(\mathbf{x})$ определяется параметрами $\boldsymbol{\mu}$, \mathbf{W} и σ^2 . Однако в этой параметризации есть избыточность параметров с учетом вращения координат латентного пространства. Чтобы убедиться в этом, рассмотрим матрицу $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{R}$, где \mathbf{R} – это ортогональная матрица. Исходя из свойства ортогональности $\mathbf{R}\mathbf{R}^T = \mathbf{I}$, получается, что величина $\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T$, входящая в ковариационную матрицу \mathbf{C} , имеет вид

$$\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T = \mathbf{W}\mathbf{R}\mathbf{R}^T\mathbf{W}^T = \mathbf{W}\mathbf{W}^T \quad (16.40)$$

и, следовательно, не зависит от \mathbf{R} . Таким образом, существует целое семейство матриц $\tilde{\mathbf{W}}$, каждая из которых приводит к одному и тому же прогнозистическому распределению. Эту инвариантность можно трактовать в контексте вращений в латентном пространстве. К обсуждению количества независимых параметров в этой модели вернемся позже.

При оценке прогнозируемого распределения необходимо найти \mathbf{C}^{-1} , что предполагает инверсию матрицы $D \times D$. Необходимые для этого вычисле-

ния можно сократить за счет использования тождества инверсии матрицы (A.7), что дает

$$\mathbf{C}^{-1} = \sigma^{-2}\mathbf{I} - \sigma^{-2}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^T, \quad (16.41)$$

где матрица $M \times M$ определяется как

$$\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}. \quad (16.42)$$

Поскольку в этом случае происходит инверсия \mathbf{M} , а не непосредственная инверсия \mathbf{C} , стоимость оценки \mathbf{C}^{-1} уменьшается с $\mathcal{O}(D^3)$ до $\mathcal{O}(M^3)$.

Помимо предсказательного распределения $p(\mathbf{x})$, также потребуется апостериорное распределение $p(\mathbf{z}|\mathbf{x})$, которое вновь можно записать непосредственно по результатам (3.100) для линейных гауссовых моделей (см. упражнение 16.8), что дает

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{M}^{-1}\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu}), \sigma^2\mathbf{M}^{-1}). \quad (16.43)$$

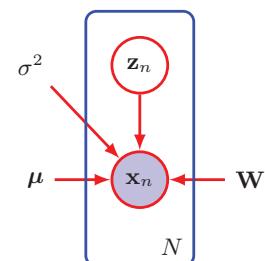
Обратите внимание, что апостериорное среднее значение зависит от \mathbf{x} , в то время как апостериорная ковариация не зависит от \mathbf{x} .

16.2.3. Максимальное правдоподобие

Далее рассмотрим определение параметров модели с помощью метода максимального правдоподобия. При наличии набора данных $\mathbf{X} = \{\mathbf{x}_n\}$ наблюдаемых точек данных вероятностная модель PCA может быть выражена в виде направленного графа, как показано на рис. 16.8. Соответствующая функция логарифмического правдоподобия определяется из (16.35) следующим образом:

$$\begin{aligned} \ln p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) &= \sum_{n=1}^N \ln p(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) \\ &= -\frac{ND}{2} \ln(2\pi) - \frac{N}{2} \ln |\mathbf{C}| - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}). \end{aligned} \quad (16.44)$$

РИС. 16.8 Вероятностная модель PCA для набора данных из N наблюдений \mathbf{x} в виде направленного графа, где каждое наблюдение \mathbf{x}_n связано со значением латентной переменной \mathbf{z}_n



Установка производной логарифмического правдоподобия по $\boldsymbol{\mu}$ равной нулю дает (см. упражнение 16.9) ожидаемый результат $\boldsymbol{\mu} = \mathbf{x}$, где \mathbf{x} – это

среднее значение данных, определенное в (16.1). Поскольку логарифмическое правдоподобие является квадратичной функцией μ , это решение представляет собой единственный максимум, что подтверждается вычислением вторых производных. Выполнив обратную подстановку, можно записать функцию логарифмического правдоподобия в виде

$$\ln p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = -\frac{N}{2}\{D \ln(2\pi) + \ln|\mathbf{C}| + \text{Tr}(\mathbf{C}^{-1}\mathbf{S})\}, \quad (16.45)$$

где \mathbf{S} – это ковариационная матрица данных, определенная в (16.3).

Максимизация относительно \mathbf{W} и σ^2 является более сложной задачей, но тем не менее имеет однозначное решение в замкнутой форме. В работе (Tipping and Bishop, 1999) было показано, что все стационарные точки функции логарифмического правдоподобия могут быть записаны в виде

$$\mathbf{W}_{\text{ML}} = \mathbf{U}_M (\mathbf{L}_M - \sigma^2 \mathbf{I})^{1/2} \mathbf{R}, \quad (16.46)$$

где \mathbf{U}_M – это матрица $D \times M$, столбцы которой задаются любым подмножеством (размера M) собственных векторов ковариационной матрицы данных \mathbf{S} . Диагональная матрица \mathbf{L}_M размером $M \times M$ имеет элементы, задаваемые соответствующими собственными значениями λ_i , а \mathbf{R} – это произвольная ортогональная матрица размером $M \times M$.

Кроме того, в работе (Tipping and Bishop, 1999) также отмечено, что максимум функции правдоподобия достигается при выборе M собственных векторов, собственные значения которых являются M -наибольшими (все остальные решения являются седловыми точками). Аналогичный результат был независимо предсказан в работе (Roweis, 1998), хотя без приведения доказательства. Снова предположим, что собственные векторы расположены в порядке убывания величин соответствующих собственных значений, так что M главных собственных векторов – это $\mathbf{u}_1, \dots, \mathbf{u}_M$. В этом случае столбцы \mathbf{W} определяют главное подпространство стандартного PCA. Соответствующее решение максимального правдоподобия для σ^2 имеет вид:

$$\sigma_{\text{ML}}^2 = \frac{1}{D - M} \sum_{i=M+1}^D \lambda_i. \quad (16.47)$$

Таким образом, σ_{ML}^2 – это средняя дисперсия, связанная с отброшенными размерностями.

Поскольку \mathbf{R} ортогональна, ее можно рассматривать как матрицу вращения в двумерном латентном пространстве. Если подставить решение для \mathbf{W} в выражение для \mathbf{C} и воспользоваться свойством ортогональности $\mathbf{R}\mathbf{R}^T = \mathbf{I}$, то получится, что \mathbf{C} не зависит от \mathbf{R} . Это просто говорит о том, что плотность предсказания не изменяется при вращении в латентном пространстве, как обсуждалось ранее. Для частного случая $\mathbf{R} = \mathbf{I}$ получается, что столбцы \mathbf{W} – это собственные векторы главных компонент, масштабированные на параметры дисперсии $\lambda_i - \sigma^2$. Интерпретация этих масштабных коэффициентов становится понятной после выяснения того, что для свертки независимых

гауссовых распределений (в данном случае распределения латентного пространства и модели шума) дисперсии являются аддитивными. Таким образом, дисперсия λ_i в направлении собственного вектора \mathbf{u}_i складывается из суммы вклада $\lambda_i - \sigma^2$ от проекции единично-вариационного распределения латентного пространства на пространство данных по соответствующему столбцу \mathbf{W} плюс изотропный вклад дисперсии σ^2 , который добавляется шумовой моделью по всем направлениям.

Стоит уделить немного времени изучению формы ковариационной матрицы, заданной в (16.36). Рассмотрим дисперсию прогнозируемого распределения вдоль некоторого направления, заданного единичным вектором \mathbf{v} , где $\mathbf{v}^T \mathbf{v} = 1$, который задается $\mathbf{v}^T \mathbf{C} \mathbf{v}$. Для начала предположим, что \mathbf{v} ортогонален главному подпространству, или, другими словами, он задается некоторой линейной комбинацией отброшенных собственных векторов. Тогда $\mathbf{v}^T \mathbf{U} = 0$ и, следовательно, $\mathbf{v}^T \mathbf{C} \mathbf{v} = \sigma^2$. Получается, что модель предсказывает дисперсию шума, ортогональную главному подпространству, которое, как следует из (16.47), является средним значением отброшенных собственных векторов. Теперь предположим, что $\mathbf{v} = \mathbf{u}_i$, где \mathbf{u}_i – это один из отброшенных собственных векторов, определяющих главное подпространство. Тогда $\mathbf{v}^T \mathbf{C} \mathbf{v} = (\lambda_i - \sigma^2) + \sigma^2 = \lambda_i$. Другими словами, эта модель корректно отражает дисперсию данных вдоль главных осей и аппроксимирует дисперсию по всем остальным направлениям одним средним значением σ^2 .

Один из способов построения модели плотности максимального правдоподобия заключается в определении собственных векторов и собственных значений ковариационной матрицы данных, а затем в оценке \mathbf{W} и σ^2 с использованием приведенных выше результатов. В этом случае для удобства можно выбрать $\mathbf{R} = \mathbf{I}$. Однако если решение с максимальным правдоподобием находится путем численной оптимизации функции правдоподобия, например с помощью алгоритма сопряженных градиентов (Fletcher, 1987; Nocedal and Wright, 1999) или ЕМ-алгоритма (см. раздел 16.3.2), то полученное значение \mathbf{R} будет, по сути, произвольным. Это означает, что столбцы \mathbf{W} не обязательно должны быть ортогональными. Если требуется ортогональный базис, тогда необходимо соответственно обработать матрицу \mathbf{W} (Golub and Van Loan, 1996). Как вариант, ЕМ-алгоритм можно модифицировать так, чтобы получить ортонормальные главные направления, отсортированные в порядке убывания соответствующих собственных значений (Ahn and Oh, 2003).

Инвариантность вращения в латентном пространстве представляет собой форму статистической неидентифицируемости, аналогичную той, что встречается в моделях смеси для дискретных латентных переменных. Здесь существует континuum параметров, любое значение которого приводит к одной и той же прогнозной плотности, в отличие от дискретной неидентифицируемости, связанной с перенаркировкой компонентов в модели смеси.

Если считать, что $M = D$, и уменьшения размерности не происходит, то $\mathbf{U}_M = \mathbf{U}$ и $\mathbf{L}_M = \mathbf{L}$. Используя свойства ортогональности $\mathbf{U} \mathbf{U}^T = \mathbf{I}$ и $\mathbf{R} \mathbf{R}^T = \mathbf{I}$, можно убедиться, что ковариация \mathbf{C} маргинального распределения для \mathbf{x} становится равной

$$\mathbf{C} = \mathbf{U}(\mathbf{L} - \sigma^2 \mathbf{I})^{1/2} \mathbf{R} \mathbf{R}^T (\mathbf{L} - \sigma^2 \mathbf{I})^{1/2} \mathbf{U}^T + \sigma^2 \mathbf{I} = \mathbf{U} \mathbf{L} \mathbf{U}^T = \mathbf{S}, \quad (16.48)$$

и, таким образом, получается стандартное решение максимального правдоподобия для гауссова распределения без ограничений, в котором ковариационная матрица задается ковариацией выборки.

Обычный РСА принято рассматривать как проекцию точек из D -мерного пространства данных на M -мерное линейное подпространство. Вероятностный РСА, в свою очередь, наиболее естественно определяется как отображение из латентного пространства в пространство данных с помощью (16.33). Для таких приложений, как визуализация и сжатие данных, это отображение можно изменить на обратное с помощью теоремы Байеса. Любая точка \mathbf{x} в пространстве данных может быть обобщена по ее апостериорному среднему значению и ковариации в латентном пространстве. Из (16.43) среднее значение определяется как

$$\mathbb{E}[\mathbf{z} | \mathbf{x}] = \mathbf{M}^{-1} \mathbf{W}_{ML}^T (\mathbf{x} - \bar{\mathbf{x}}), \quad (16.49)$$

где \mathbf{M} определяется в (16.42). Это проецируется на точку в пространстве данных, заданную

$$\mathbf{W} \mathbb{E}[\mathbf{z} | \mathbf{x}] + \boldsymbol{\mu}. \quad (16.50)$$

Обратите внимание, что это уравнение имеет ту же форму, что и уравнения для регуляризованной линейной регрессии (см. раздел 4.1.6), а также является следствием максимизации функции правдоподобия для линейной гауссовой модели. Точно так же из (16.43) следует, что апостериорная ковариация задается $\sigma^2 \mathbf{M}^{-1}$ и не зависит от \mathbf{x} .

Если взять предел $\sigma^2 \rightarrow 0$, то апостериорное среднее значение сводится к

$$(\mathbf{W}_{ML}^T \mathbf{W}_{ML})^{-1} \mathbf{W}_{ML}^T (\mathbf{x} - \bar{\mathbf{x}}), \quad (16.51)$$

что представляет собой ортогональную проекцию точки данных на латентное пространство, и, таким образом, восстанавливается стандартная модель РСА (см. упражнение 16.11). Однако в этом пределе апостериорная ковариация равна нулю и плотность становится сингулярной. При $\sigma^2 > 0$ латентная проекция (см. упражнение 16.12) смещается к началу координат относительно ортогональной проекции.

Наконец, обратите внимание, что важная роль вероятностной модели РСА заключается в определении многомерного гауссова распределения, в котором число степеней свободы (другими словами, число независимых параметров) можно контролировать, но при этом модель должна отражать доминирующие корреляции в данных. Напомним, что общее гауссово распределение имеет в своей ковариационной матрице $D(D+1)/2$ независимых параметров плюс еще D параметров в его среднем значении (см. раздел 3.2). Получается, что число параметров возрастает квадратично с увеличением D и может стать чрезмерным в пространствах большой размерности. Если ограничить ковариационную матрицу до диагональной, то у нее останется только D не-

зависимых параметров, и теперь число параметров будет расти линейно по мере увеличения размерности. Однако теперь она рассматривает переменные как независимые и, следовательно, больше не может выражать корреляции между ними. Вероятностный РСА обеспечивает элегантный компромисс, когда можно уловить M наиболее значимых корреляций и при этом гарантировать, что общее число параметров растет линейно с D . В этом можно убедиться путем оценки числа степеней свободы в модели вероятностного РСА следующим образом. Ковариационная матрица \mathbf{C} зависит от параметров \mathbf{W} , которые имеют размер $D \times M$, и σ^2 , что дает общее число параметров $DM + 1$. Однако при этом в этой параметризации присутствует некоторая избыточность, связанная с поворотами системы координат в латентном пространстве. Выражающая эти вращения ортогональная матрица \mathbf{R} имеет размер $M \times M$. В первом столбце этой матрицы $M - 1$ независимых параметров, поскольку вектор столбца должен быть нормализован на единицу длины. Во втором столбце $M - 2$ независимых параметров, потому что столбец должен быть нормализован и ортогонален предыдущему столбцу, и т. д. Если суммировать этот арифметический ряд, то получается, что \mathbf{R} имеет в общей сложности $M(M - 1)/2$ независимых параметров. Получается, что число степеней свободы в ковариационной матрице \mathbf{C} равно

$$DM + 1 - M(M - 1)/2. \quad (16.52)$$

Поэтому число независимых параметров в этой модели при фиксированном M растет линейно с D . Если принять $M = D - 1$, то получится стандартный результат для гауссова распределения с полной ковариацией (см. упражнение 16.14). В этом случае дисперсия по $D - 1$ линейно независимым направлениям определяется столбцами \mathbf{W} , а дисперсия по остальным направлениям задается σ^2 . Если $M = 0$, то модель эквивалентна случаю изотропной ковариации.

16.2.4. Факторный анализ

Факторный анализ (*factor analysis*) является линейной гауссовой моделью латентных переменных, которая тесно связана с вероятностным РСА. Его определение отличается от определения вероятностного РСА только тем, что условное распределение наблюдаемой переменной \mathbf{x} по латентной переменной \mathbf{z} имеет диагональную, а не изотропную ковариацию, так что

$$p(\mathbf{x} | \mathbf{z}) = \mathcal{N}(\mathbf{x} | \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}), \quad (16.53)$$

где $\boldsymbol{\Psi}$ – это диагональная матрица $D \times D$. Обратите внимание, что модель факторного анализа, как и вероятностный РСА, предполагает, что наблюдаемые переменные x_1, \dots, x_D являются независимыми при заданной латентной переменной \mathbf{z} . По сути, модель факторного анализа объясняет наблюдаемую ковариационную структуру данных, представляя независимую дисперсию по каждой координате в матрице $\boldsymbol{\Psi}$ и отражая ковариацию между переменными в матрице \mathbf{W} . В профильной литературе по факторному анализу столбцы \mathbf{W} ,

отражающие корреляции между наблюдаемыми переменными, называются *факторными нагрузками* (*factor loadings*), а диагональные элементы Ψ , которые представляют независимые шумовые дисперсии для каждой из переменных, называются *的独特性ами* (*uniquenesses*).

Истоки факторного анализа такие же давние, как и у PCA, и его обсуждение можно найти в работах (Everitt, 1984), (Bartholomew, 1987) и (Basilevsky, 1994). Связи между факторным анализом и PCA были исследованы в работах (Lawley, 1953) и (Anderson, 1963), которые показали, что в стационарных точках функции правдоподобия для модели факторного анализа с $\Psi = \sigma^2 \mathbf{I}$ столбцы \mathbf{W} представляют собой масштабированные собственные векторы ковариационной матрицы выборки, а σ^2 является усредненным значением отброшенных собственных значений. Позже в работе (Tipping and Bishop, 1999) было показано, что максимум функции логарифмического правдоподобия достигается в том случае, если собственные векторы, входящие в \mathbf{W} , выбраны как главные собственные векторы.

Используя (16.34), можно убедиться, что маргинальное распределение для наблюдаемой переменной задается $p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \mathbf{C})$, где теперь

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \Psi. \quad (16.54)$$

Как и в случае с вероятностным PCA (см. упражнение 16.16), эта модель инвариантна к вращению латентного пространства. Исторически факторный анализ был предметом споров при попытке интерпретации отдельных факторов (координат в \mathbf{z} -пространстве), что оказалось проблематичным из-за неидентичности факторного анализа, связанной с вращениями в этом пространстве. Однако в нашем случае факторный анализ будет рассматриваться как разновидность модели плотности латентных переменных, в которой интерес представляет форма латентного пространства, а не конкретный выбор координат, используемых для его описания. Если требуется устраниить вырождение, связанное с вращением латентного пространства, необходимо рассмотреть негауссовы распределения латентных переменных, что приведет к моделям анализа независимых компонентов (см. раздел 16.2.5).

Еще одно различие между вероятностным PCA и факторным анализом заключается в их поведении (см. упражнение 16.17) при преобразованиях набора данных. Для PCA и вероятностного PCA при вращении системы координат в пространстве данных получается точно такая же подгонка к данным, но с матрицей \mathbf{W} , преобразованной соответствующей матрицей вращения. Однако для факторного анализа аналогичное свойство заключается в том, что если провести покомпонентное изменение масштаба векторов данных, то это обернется соответствующим изменением масштаба элементов Ψ .

16.2.5. Анализ независимых компонентов

Одним из обобщений линейной гауссовой модели латентных переменных является анализ моделей, где наблюдаемые переменные линейно связаны с латентными переменными, но для которых латентное распределение не

является гауссовым. Такой тип модельных подходов, известный как *анализ независимых компонентов* (*independent component analysis, ICA*), позволяет определить распределение латентных переменных с учетом факторизации, а именно

$$p(\mathbf{z}) = \prod_{j=1}^M p(z_j). \quad (16.55)$$

Чтобы лучше уловить суть таких моделей, рассмотрим ситуацию, когда два человека разговаривают одновременно и их голоса записываются с помощью двух микрофонов. Если пренебречь такими эффектами, как временная задержка и эхо, то принимаемые микрофонами сигналы в любой момент времени будут представлять собой линейные комбинации амплитуд двух голосов. Коэффициенты этой линейной комбинации будут постоянными, и если удастся определить их значения по данным выборки, то можно инвертировать процесс смешивания (предполагается, что он несингулярный) и, таким образом, получить два чистых сигнала, каждый из которых содержит голос только одного человека. Это пример задачи под названием *слепое разделение входных сигналов* (*blind source separation*), где «слепой» означает, что нам даны только смешанные данные, и ни исходные источники, ни коэффициенты смешивания не доступны для наблюдения (Cardoso, 1998).

При решении подобных задач иногда используется подход (MacKay, 2003), при котором временная природа сигналов игнорируется, а последовательные выборки рассматриваются как i.i.d. В генеративной модели имеются две латентные переменные, соответствующие ненаблюдаемым амплитудам речевого сигнала, и две наблюдаемые переменные, задаваемые значениями сигнала от микрофонов. Латентные переменные имеют совместное распределение, которое факторизуется, как указано выше, а наблюдаемые переменные задаются линейной комбинацией латентных переменных. Нет необходимости включать распределение шума, поскольку число скрытых переменных равно числу наблюдаемых переменных, и поэтому маргинальное распределение наблюдаемых переменных в общем случае не будет сингулярным, так что наблюдаемые переменные – это просто детерминированные линейные комбинации скрытых переменных. При заданном наборе наблюдений функция правдоподобия для этой модели является функцией коэффициентов в линейной комбинации. Логарифмическое правдоподобие может быть максимизировано с помощью градиентной оптимизации, что представляет собой специфическую версию ICA.

Для успешного применения этого подхода необходимо, чтобы латентные переменные имели негауссово распределение. Чтобы убедиться в этом, вспомним, что в вероятностном РСА (и в факторном анализе) распределение в латентном пространстве задается изотропным гауссовым распределением с нулевым средним значением. Поэтому модель не может различить два разных выбора латентных переменных, если они отличаются только поворотом в латентном пространстве. Это можно проверить напрямую, обратив вни-

мание на то, что предельная плотность (16.35), а следовательно и функция правдоподобия, не изменится, если выполнить преобразование $\mathbf{W} \rightarrow \mathbf{WR}$, где \mathbf{R} – это ортогональная матрица, удовлетворяющая $\mathbf{RR}^T = \mathbf{I}$, поскольку матрица \mathbf{C} , заданная в (16.36), сама является инвариантной. Расширение модели до более общих гауссовых латентных распределений не меняет этого факта, поскольку, как уже было замечено, такая модель эквивалентна модели с нулевым средним значением изотропной гауссовой латентной переменной.

Другой способ выяснить причины, по которым в линейной модели гаусса распределения латентных переменных недостаточно для поиска независимых компонентов, заключается в том, что главные компоненты представляют собой вращение системы координат в пространстве данных с целью диагонализации ковариационной матрицы. Распределение данных в новых координатах становится некоррелированным. Хотя нулевая корреляция является необходимым условием независимости, она тем не менее не является достаточным условием (см. упражнение 2.39). На практике для латентного распределения переменных обычно выбирают формулу

$$p(z_j) = \frac{1}{\pi \cosh(z_j)} = \frac{2}{\pi(e^{z_j} + e^{-z_j})}, \quad (16.56)$$

которая имеет большие хвосты по сравнению с гауссовым распределением. Это свидетельствует о том, что многие распределения в реальном мире также обладают этим свойством.

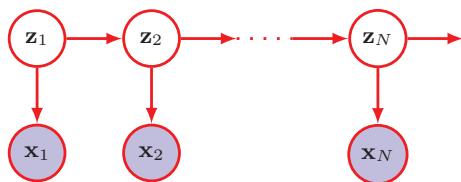
Первоначальная модель ICA (Bell and Sejnowski, 1995) была основана на оптимизации целевой функции, определяемой максимизацией информации. Одно из преимуществ вероятностной формулировки латентных переменных заключается в том, что она помогает объяснить и сформулировать обобщения базовой ICA. Например, *независимый факторный анализ (independent factor analysis)* (Attias, 1999) рассматривает модель, в которой число латентных и наблюдаемых переменных может быть разным, наблюдаемые переменные зашумлены, а отдельные латентные переменные имеют гибкие распределения и моделируются смесями гауссовых распределений. Логарифмическое правдоподобие для этой модели максимизируется с помощью EM, а восстановление латентных переменных аппроксимируется с помощью вариационного подхода. Было рассмотрено множество других типов моделей, и в настоящее время существует огромное количество специальной литературы по ICA и ее применению (Jutten and Herault, 1991; Comon, Jutten and Herault, 1991; Amari, Cichocki and Yang, 1996; Pearlmutter and Parra, 1997; Hyvärinen and Oja, 1997; Hinton et al, 2001; Miskin and MacKay, 2001; Hojen-Sorensen, Winther and Hansen, 2002; Choudrey and Roberts, 2003; Chan, Lee and Sejnowski, 2003; Stone, 2004).

16.2.6. Фильтры Калмана

До сих пор подразумевалось, что значения данных являются i.i.d. (независимыми и одинаково распределенными). Наиболее распространенной ситуаци-

ей несоблюдения этого предположения является случай, когда точки данных образуют упорядоченную последовательность. Ранее уже было отмечено, что скрытую марковскую модель можно рассматривать как расширение моделей смесей (см. раздел 15.3.1), позволяющее учитывать последовательные корреляции в данных. Аналогичным образом непрерывная модель скрытых переменных может быть расширена для работы с последовательными данными путем соединения скрытых переменных в цепь Маркова, как показано в графовой модели на рис. 16.9. Такая модель известна как *линейная динамическая система* (*linear dynamical system*) или *фильтр Калмана* (*Kalman filter*) (Zarchan and Musoff, 2005). Обратите внимание, что это та же самая графовая структура, что и скрытая марковская модель (см. раздел 15.3.1). Интересно отметить, что исторически скрытые марковские модели и линейные динамические системы разрабатывались независимо друг от друга. Однако после того, как они обе были выражены в виде графовых моделей, глубокая связь между ними сразу же стала очевидной. Фильтры Калмана широко используются во многих программах отслеживания в реальном времени, например для сопровождения самолетов по радиолокационным сигналам.

РИС. 16.9 Вероятностная графическая модель для последовательных данных, известная как линейная динамическая система, или фильтр Калмана, где скрытые переменные образуют цепь Маркова



В простейшей модели распределения $p(\mathbf{x}_n | \mathbf{z}_n)$ на рис. 16.9 представляют собой линейную гауссову модель распределения латентных переменных для данного конкретного наблюдения, подобную той, что обсуждалась ранее для i.i.d.-данных. Однако латентные переменные $\{\mathbf{z}_n\}$ больше не рассматриваются как независимые, а образуют цепь Маркова, в которой распределение $p(\mathbf{z}_n | \mathbf{z}_{n-1})$ каждой латентной переменной зависит от состояния предыдущей латентной переменной в цепи. Опять же, они могут быть выбраны линейными гауссовыми, в которых распределение \mathbf{z}_n является гауссовым со средним значением, определяемым линейной функцией \mathbf{z}_{n-1} . Обычно параметры всех распределений $p(\mathbf{x}_n | \mathbf{z}_n)$ общие, и точно так же общими являются параметры распределений $p(\mathbf{z}_n | \mathbf{z}_{n-1})$, так что общее число параметров в модели фиксировано и не зависит от длины последовательности. Эти параметры могут быть получены из данных методом максимального правдоподобия с помощью эффективных алгоритмов, предполагающих распространение сообщений по графу (Bishop, 2006). Однако в оставшейся части этой главы речь пойдет об i.i.d.-данных.

16.3. Нижняя граница доказательств

В процессе обсуждения моделей с дискретными латентными переменными (см. раздел 15.4) была получена *нижняя граница доказательств* (*evidence lower bound, ELBO*) для маргинального логарифмического правдоподобия и было продемонстрировано использование этой границы в качестве основы для выводения алгоритма оптимизации ожиданий (*expectation-maximization, EM*), включая такие его обобщения, как вариационный вывод (*variational inference*). Эта же схема применима к непрерывным латентным переменным, а также к моделям с сочетанием дискретных и непрерывных переменных. Здесь дается несколько иной вариант получения ELBO и подразумевается, что латентные переменные \mathbf{z} непрерывны.

Рассмотрим модель $p(\mathbf{x}, \mathbf{z} | \mathbf{w})$ с наблюдаемой переменной \mathbf{x} , латентной переменной \mathbf{z} и обучаемым вектором параметров \mathbf{w} . Если ввести произвольное распределение $q(\mathbf{z})$ по латентной переменной, то функцию логарифмического правдоподобия $\ln p(\mathbf{x} | \mathbf{w})$ можно записать в виде суммы двух членов (см. упражнение 16.18) в форме

$$\ln p(\mathbf{x} | \mathbf{w}) = \mathcal{L}(\mathbf{w}) + \text{KL}(q(\mathbf{z}) || p(\mathbf{z} | \mathbf{x}, \mathbf{w})), \quad (16.57)$$

где определено

$$\mathcal{L}(\mathbf{q}, \mathbf{w}) = \int q(\mathbf{z}) \ln p(\mathbf{x}, \mathbf{z} | \mathbf{w}) q(\mathbf{z}) d\mathbf{z}, \quad (16.58)$$

$$\text{KL}(q(\mathbf{z}) || p(\mathbf{z} | \mathbf{x}, \mathbf{w})) = - \int q(\mathbf{z}) \ln p(\mathbf{z} | \mathbf{x}, \mathbf{w}) q(\mathbf{z}) d\mathbf{z}. \quad (16.59)$$

Поскольку $\text{KL}(q(\mathbf{z}) || p(\mathbf{z} | \mathbf{x}, \mathbf{w}))$ является дивергенцией Кульбака–Лейблера (см. раздел 2.5.5), она удовлетворяет свойству $\text{KL}(\cdot || \cdot) \geq 0$, из которого следует, что

$$\ln p(\mathbf{x} | \mathbf{w}) > \mathcal{L}(\mathbf{w}), \quad (16.60)$$

и поэтому получается, что $\mathcal{L}(\mathbf{q}, \mathbf{w})$, заданная в (16.58), образует нижнюю границу логарифмического правдоподобия, известную как *нижняя граница доказательства*, или ELBO. Здесь видно, что $\mathcal{L}(\mathbf{q}, \mathbf{w})$ имеет ту же форму (15.53), что и для дискретного случая, но суммы заменены интегралами.

Чтобы максимизировать функцию логарифмического правдоподобия, используется двухэтапная итерационная процедура под названием *алгоритм максимизации ожиданий*, или EM-алгоритм, когда поочередно максимизируются $\mathcal{L}(\mathbf{q}, \mathbf{w})$ относительно $q(\mathbf{z})$ (шаг E) и \mathbf{w} (шаг M). Сначала задаются параметры $\mathbf{w}^{(\text{old})}$. Затем на шаге E нужно зафиксировать \mathbf{w} и максимизировать нижнюю границу относительно $q(\mathbf{z})$. Это несложно, если учесть, что наибольшее значение границы получается при минимизации расхождения Кульбака–Лейблера в (16.59) и, следовательно, достигается при $q(\mathbf{z}) = p(\mathbf{z} | \mathbf{x}, \mathbf{w}^{(\text{old})})$, для которого расхождение Кульбака–Лейблера равно нулю. На шаге M этот выбор $q(\mathbf{z})$ остается фиксированным, и $\mathcal{L}(\mathbf{q}, \mathbf{w})$ максимизируется по отношению к \mathbf{w} . При подстановке $q(\mathbf{z})$ в (16.58) получится:

$$\begin{aligned}\mathcal{L}(q, \mathbf{w}) = & \int p(\mathbf{z} | \mathbf{x}, \mathbf{w}^{(\text{old})}) \ln p(\mathbf{x}, \mathbf{z} | \mathbf{w}) d\mathbf{z} \\ & - \int p(\mathbf{z} | \mathbf{x}, \mathbf{w}^{(\text{old})}) \ln p(\mathbf{z} | \mathbf{x}, \mathbf{w}^{(\text{old})}) d\mathbf{z}.\end{aligned}\quad (16.61)$$

Теперь на шаге М это значение максимизируется по отношению к \mathbf{w} , при этом $\mathbf{w}^{(\text{old})}$ остается фиксированным. Обратите внимание, что второй член в правой части (16.61) не зависит от \mathbf{w} , и поэтому на шаге М может быть проигнорирован. Первый член в правой части – это ожидание *полного логарифмического правдоподобия данных* (*complete data log likelihood*) (см. раздел 15.3), где ожидание берется относительно апостериорного распределения \mathbf{z} , вычисленного с помощью $\mathbf{w}^{(\text{old})}$.

Если имеется набор данных $\mathbf{x}_1, \dots, \mathbf{x}_N$ из i.i.d.-наблюдений, то функция правдоподобия имеет вид:

$$\ln p(\mathbf{X} | \mathbf{w}) = \sum_{n=1}^N \ln p(\mathbf{x}_n | \mathbf{w}), \quad (16.62)$$

где матрица данных \mathbf{X} состоит из $\mathbf{x}_1, \dots, \mathbf{x}_N$, а параметры \mathbf{w} являются общими для всех точек данных. Для каждой точки данных вводится соответствующая латентная переменная \mathbf{z}_n со связанным с ней распределением $q(\mathbf{z}_n)$, и в результате шагов, аналогичных использованным при выводе (16.58) (см. упражнение 16.19), получается ELBO в виде

$$\mathcal{L}(q, \mathbf{w}) = \sum_{n=1}^N \int q(\mathbf{z}_n) \ln \left\{ \frac{p(\mathbf{x}_n, \mathbf{z}_n | \mathbf{w})}{q(\mathbf{z}_n)} \right\} d\mathbf{z}_n. \quad (16.63)$$

При обсуждении вариационных автокодировщиков (раздел 19.2) будет рассмотрена модель, для которой точное решение шага Е не представляется возможным, поэтому вместо этого выполняется частичная максимизация путем моделирования $q(\mathbf{z})$ с помощью глубокой нейронной сети и последующего использования ELBO для обучения параметров сети.

16.3.1. Максимизация ожидания

Теперь для определения параметров вероятностной модели РСА может быть использован ЕМ-алгоритм, полученный в результате итеративной максимизации нижней границы доказательства. Это может показаться довольно бессмысленным, поскольку уже получено точное решение в замкнутой форме для значений параметров максимального правдоподобия. Однако в пространствах высокой размерности использование итеративной ЕМ-процедуры может дать вычислительные преимущества по сравнению с непосредственной обработкой ковариационной матрицы выборки. Эта ЕМ-процедура (см. раздел 16.2.4) также может быть распространена на модель факторного анализа, для которой не существует решения в замкнутой форме. Наконец, она в принципе позволяет решать проблему недостающих данных.

Можно вывести ЕМ-алгоритм для вероятностного РСА в соответствии с общей концепцией ЕМ (см. раздел 15.3). Для этого нужно записать логарифми-

ческое правдоподобие для полных данных и взять его ожидание относительно апостериорного распределения латентного распределения, оцененного с помощью «старых» значений параметров. Максимизация этого ожидаемого логарифмического правдоподобия полных данных дает «новые» значения параметров. Поскольку точки данных предположительно независимы, функция логарифмического правдоподобия полных данных имеет вид:

$$\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \mathbf{W}, \sigma^2) = \sum_{n=1}^N \{\ln p(\mathbf{x}_n | \mathbf{z}_n) + \ln p(\mathbf{z}_n)\}, \quad (16.64)$$

где n -я строка матрицы \mathbf{Z} задается \mathbf{z}_n . Уже известно, что точное решение максимального правдоподобия для $\boldsymbol{\mu}$ дается выборочным средним значением \mathbf{x} , определенным в (16.1), поэтому замену $\boldsymbol{\mu}$ удобнее произвести на этом этапе. Используя выражения (16.31) и (16.32) для латентного и условного распределений соответственно и взяв ожидание относительно апостериорного распределения по латентным переменным, получаем:

$$\begin{aligned} \mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \mathbf{W}, \sigma^2)] &= -\sum_{n=1}^N \left\{ \frac{D}{2} \ln(2\pi\sigma^2) + \frac{1}{2} \text{Tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T]) \right. \\ &\quad + \frac{1}{2\sigma^2} \|\mathbf{x}_n - \boldsymbol{\mu}\|^2 - \frac{1}{\sigma^2} \mathbb{E}[\mathbf{z}_n]^T \mathbf{W}^T (\mathbf{x}_n - \boldsymbol{\mu}) \\ &\quad \left. + \frac{1}{2\sigma^2} \text{Tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \mathbf{W}^T \mathbf{W}) + \frac{M}{2} \ln(2\pi) \right\}. \end{aligned} \quad (16.65)$$

Обратите внимание, что это зависит от апостериорного распределения только через достаточную статистику гауссова распределения. Таким образом, на шаге E для оценки старых значений параметров используются

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1} \mathbf{W}^T (\mathbf{x}_n - \bar{\mathbf{x}}), \quad (16.66)$$

$$\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] = \sigma^2 \mathbf{M}^{-1} + \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^T, \quad (16.67)$$

которые следуют непосредственно из апостериорного распределения (16.43) вместе со стандартным результатом $\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] = \text{cov}[\mathbf{z}_n] + \mathbb{E}[\mathbf{z}_n] \mathbb{E}[\mathbf{z}_n]^T$. Здесь \mathbf{M} определяется в (16.42).

На шаге M выполняется максимизация по \mathbf{W} и σ^2 , при этом апостериорные статистики остаются фиксированными. Максимизация по σ^2 проста. Для максимизации по \mathbf{W} (см. уравнение 16.21) используется (A.24), что дает уравнения шага M:

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}}) \mathbb{E}[\mathbf{z}_n]^T \right] \left[\sum_{n=1}^N \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \right]^{-1}, \quad (16.68)$$

$$\begin{aligned} \sigma_{\text{new}}^2 &= \frac{1}{ND} \sum_{n=1}^N \{ \|\mathbf{x}_n - \bar{\mathbf{x}}\|^2 - 2\mathbb{E}[\mathbf{z}_n]^T \mathbf{W}_{\text{new}}^T (\mathbf{x}_n - \bar{\mathbf{x}}) \\ &\quad + \text{Tr}(\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \mathbf{W}_{\text{new}}^T \mathbf{W}_{\text{new}}) \}. \end{aligned} \quad (16.69)$$

ЕМ-алгоритм для вероятностного РСА работает следующим образом: инициализируются параметры, затем поочередно вычисляются достаточные статистики апостериорного распределения латентного пространства с помощью (16.66) и (16.67) на шаге Е и пересматриваются значения параметров с помощью (16.68) и (16.69) на шаге М.

Одним из преимуществ ЕМ-алгоритма для РСА является его вычислительная эффективность при решении крупномасштабных задач (Roweis, 1998). В отличие от обычного РСА на базе разложения по собственным векторам ковариационной матрицы выборки, ЕМ-подход является итерационным, и поэтому может показаться менее привлекательным. Однако каждый цикл ЕМ-алгоритма может оказаться более эффективным в пространствах высокой размерности с вычислительной точки зрения, нежели обычный РСА. Чтобы убедиться в этом, вспомним, что для вычисления декомпозиции собственных векторов ковариационной матрицы требуется $\mathcal{O}(D^3)$ вычислений. Зачастую интерес представляют только первые M собственных векторов и соответствующие им собственные значения, и в этом случае могут использоваться алгоритмы с затратами $\mathcal{O}(MD^2)$. Однако оценка ковариационной матрицы требует $\mathcal{O}(ND^2)$ вычислений, где N – это количество точек данных. Такие алгоритмы, как метод снимков состояния (snapshot method) (Sirovich, 1987), где предполагается, что собственные векторы являются линейными комбинациями векторов данных, позволяют избежать прямой оценки ковариационной матрицы, но имеют размерность $\mathcal{O}(N^3)$ и, следовательно, не подходят для больших наборов данных. Описанный здесь ЕМ-алгоритм также не строит ковариационную матрицу в явном виде. Однако наиболее требовательными к вычислительным ресурсам являются шаги, связанные с суммированием по набору данных, что соответствует $\mathcal{O}(NDM)$. Для больших D и $M \ll D$ это может быть значительной экономией по сравнению с $\mathcal{O}(ND^2)$ и может компенсировать итерационную специфику ЕМ-алгоритма.

Обратите внимание, что этот ЕМ-алгоритм может быть реализован в интерактивной форме, где каждая D -мерная точка данных считывается и обрабатывается, а затем отбрасывается перед рассмотрением следующей точки данных. Для понимания этого факта заметим, что величины, оцениваемые на шаге Е (M -мерный вектор и матрица $M \times M$), могут быть вычислены для каждой точки данных отдельно, а на шаге М необходимо накапливать суммы по точкам данных, что можно делать инкрементально. Такой подход может быть выгоден при больших значениях N и D .

Поскольку теперь для РСА имеется полностью вероятностная модель, можно работать с отсутствующими данными при условии, что они отсутствуют случайным образом, т. е. процесс вычисления отсутствующих данных не зависит от величин каких-либо наблюдаемых или ненаблюдаемых переменных. Такие наборы данных можно обрабатывать путем маргинализации по распределению ненаблюдаемых переменных, а полученную функцию правдоподобия можно максимизировать с помощью алгоритма ЕМ (см. упражнение 16.22).

16.3.2. ЕМ для РСА

Еще одна элегантная особенность ЕМ-подхода заключается в возможности взять предел $\sigma^2 \rightarrow 0$, соответствующий стандартному РСА, и все равно получить корректный ЕМ-совместимый алгоритм (Roweis, 1998). Из (16.67) видно, что единственная величина, которую необходимо вычислить на шаге Е, – это $\mathbb{E}[z_n]$. Кроме того, шаг М упрощается, поскольку $M = W^T W$. Чтобы подчеркнуть простоту алгоритма, определим \tilde{X} как матрицу размера $N \times D$, n -я строка которой задана вектором $x_n - \bar{x}$, и аналогично определим Ω как матрицу размера $M \times N$, n -й столбец которой задан вектором $\mathbb{E}[z_n]$. Тогда шаг Е (16.66) ЕМ-алгоритма для РСА приобретает вид:

$$\Omega = (W_{\text{old}}^T W_{\text{old}})^{-1} W_{\text{old}}^T \tilde{X}^T, \quad (16.70)$$

а шаг М (16.68) принимает вид:

$$W_{\text{new}} = \tilde{X}^T \Omega^T (\Omega \Omega^T)^{-1}. \quad (16.71)$$

И вновь они могут быть воплощены в онлайновой форме. Эти уравнения имеют следующую простую интерпретацию. Из предыдущего обсуждения следует, что шаг Е включает ортогональное проецирование точек данных на текущую оценку главного подпространства. Соответственно, шаг М представляет собой повторную оценку главного подпространства (см. упражнение 16.23) для минимизации ошибки реконструкции, при которой проекции фиксированы.

Можно привести простую физическую аналогию для этого ЕМ-алгоритма, что легко визуализировать для $D = 2$ и $M = 1$. Рассмотрим набор точек данных в двух измерениях, и пусть одномерное главное подпространство представлено сплошным стержнем. Теперь привяжем каждую точку данных к стержню с помощью пружины, подчиняющейся закону Гука (сила пропорциональна длине пружины, поэтому запасенная энергия пропорциональна квадрату длины пружины). На этапе Е стержень останется неподвижным, а точки крепления будут скользить вверх и вниз по стержню для минимизации энергии. Это заставляет каждую точку крепления (независимо) позиционироваться в ортогональной проекции соответствующей точки данных на стержень. На этапе М точки крепления остаются неподвижными, а затем стержень освобождается и перемещается в положение с минимальной энергией. Шаг Е и шаг М повторяются до тех пор, пока не будет достигнут подходящий критерий сходимости, как показано на рис. 16.10. (а) Набор точек данных, показанных зеленым цветом, вместе с истинными главными компонентами (показанными в виде собственных векторов, масштабированных квадратными корнями собственных значений). (б) Начальная конфигурация главного подпространства, определяемого W , показана красным цветом, а проекции скрытых точек Z на пространство данных, определяемые ZW^T , показаны голубым цветом. (с) После одного шага М происходит обновление W , при этом Z остаются фиксированными. (д) После очередного шага Е значения Z

обновляются и дают ортогональные проекции, при этом \mathbf{W} остается фиксированным. (e) После второго шага M. (f) Сходящееся решение.)

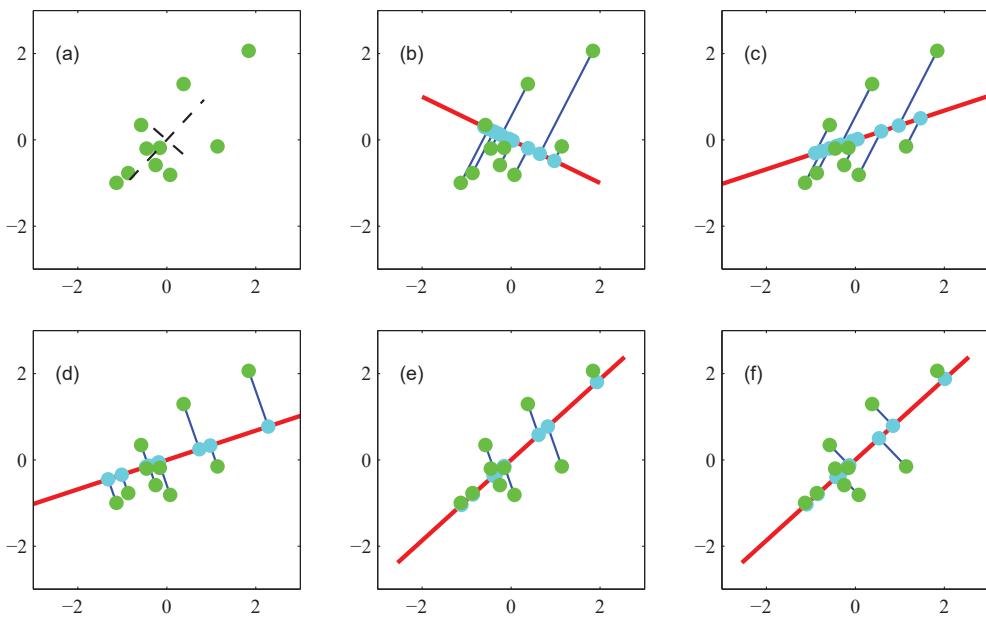


РИС. 16.10 Синтетические данные, иллюстрирующие ЕМ-алгоритм для РСА, определенный в (16.70) и (16.71)

16.3.3. ЕМ для факторного анализа

Параметры μ , \mathbf{W} и Ψ в модели факторного анализа (см. раздел 16.2.4) можно определить методом максимального правдоподобия. Решение для μ по-прежнему задается средним значением выборки. Но в отличие от вероятностного РСА, для \mathbf{W} больше не существует замкнутого решения в форме максимального правдоподобия, поэтому его необходимо определять итеративно. Поскольку факторный анализ является моделью с латентными переменными (см. упражнение 16.24), это можно сделать с помощью ЕМ-алгоритма (Rubin and Thayer, 1982), аналогичного используемому для вероятностного РСА. В частности, уравнения шага Е имеют вид:

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{G}\mathbf{W}^T\Psi^{-1}(\mathbf{x}_n - \bar{\mathbf{x}}), \quad (16.72)$$

$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] = \mathbf{G} + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^T, \quad (16.73)$$

где определено

$$\mathbf{G} = (\mathbf{I} + \mathbf{W}^T\Psi^{-1}\mathbf{W})^{-1}. \quad (16.74)$$

Обратите внимание, что это выражено в форме, которая предполагает инверсию матриц размера $M \times M$, а не $D \times D$ (за исключением диагональной

матрицы $D \times D$ у Ψ , инверсия которой тривиально вычисляется за $\mathcal{O}(D)$ шагов), что удобно, поскольку зачастую $M \ll D$. Аналогично уравнения шага М (см. упражнение 16.25) имеют вид:

$$\mathbf{W}_{\text{new}} = \left[\sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}}) \mathbb{E}[\mathbf{z}_n]^T \right] \left[\sum_{n=1}^N \mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T] \right]^{-1}, \quad (16.75)$$

$$\Psi_{\text{new}} = \text{diag} \left\{ \mathbf{S} - \mathbf{W}_{\text{new}} \frac{1}{N} \sum_{n=1}^N \mathbb{E}[\mathbf{z}_n](\mathbf{x}_n - \bar{\mathbf{x}})^T \right\}, \quad (16.76)$$

где оператор diag обращает в ноль все недиагональные элементы матрицы.

16.4. Нелинейные модели латентных переменных

До сих пор в этой главе речь шла о моделях латентных переменных на основе линейных преобразований из латентного пространства в пространство данных. Естественно задаться вопросом, можно ли использовать гибкость глубоких нейронных сетей для представления более сложных преобразований и при этом использовать способность глубоких сетей к обучению для подгонки полученного распределения к набору данных. Рассмотрим простое распределение по векторной переменной \mathbf{z} , например гауссово распределение вида

$$p_z(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}). \quad (16.77)$$

Теперь предположим, что \mathbf{z} преобразуется с помощью функции $\mathbf{x} = \mathbf{g}(\mathbf{z}, \mathbf{w})$, заданной глубокой нейронной сетью, где \mathbf{w} определяет веса и смещения. Сочетание распределения по \mathbf{z} вместе с нейронной сетью определяет распределение по \mathbf{x} . Выборка из такой модели проста, поскольку можно генерировать выборки из $p_z(\mathbf{z})$, а затем преобразовывать каждую из них с помощью функции нейронной сети для получения соответствующих выборок \mathbf{x} . Это очень эффективный процесс, поскольку он не требует итераций.

Чтобы обучить $\mathbf{g}(\mathbf{z}, \mathbf{w})$ на основе данных, рассмотрим вопрос оценки функции правдоподобия $p(\mathbf{x} | \mathbf{w})$. Распределение \mathbf{x} задается формулой замены переменных для плотностей (см. раздел 2.4):

$$p_x(\mathbf{x}) = p_z(\mathbf{z}(\mathbf{x})) |\det \mathbf{J}(\mathbf{x})|, \quad (16.78)$$

где \mathbf{J} – это матрица Якоби частных производных, элементы которой даны в виде

$$J_{ij}(\mathbf{x}) = \frac{\partial z_i}{\partial x_j}. \quad (16.79)$$

Для оценки распределения $p_z(z(x))$ в правой части (16.78) для заданного вектора данных x и для оценки матрицы Якоби в (16.79) для этого же значения x необходима инверсия $z = g^{-1}(x, w)$ функции нейронной сети. Для большинства нейронных сетей эта инверсия будет не вполне определенной. Например, сеть может представлять собой функцию «многие к одному», в которой несколько различных входных значений соответствуют одному и тому же выходному значению, и в этом случае формула замены переменных не даст четко определенной плотности. Более того, если размерность латентного пространства отличается от размерности пространства данных, то преобразование не будет инвертируемым.

В этом случае можно ограничиться инвертируемыми функциями $g(z, w)$, для чего необходимо, чтобы z и x имели одинаковую размерность. Этот подход будет рассмотрен более подробно в главе 18, когда будет представлена техника нормализации потоков.

16.4.1. Нелинейные многообразия

Требование наличия одинакового числа измерений в латентном пространстве и пространстве данных является существенным ограничением. Рассмотрим ситуацию, когда z имеет размерность M , а x – размерность D , и при этом $M < D$. В этом случае распределение по x ограничено многообразием (*manifold*), или подпространством, размерности M , как показано на рис. 16.11. Многообразия низкой размерности встречаются во многих приложениях машинного обучения, например, при моделировании распределения естественных изображений (см. раздел 6.1.4). Нелинейные модели латентных переменных могут быть очень полезны при моделировании таких данных, поскольку они выражают сильное индуктивное смещение, согласно которому данные не «заполняют» пространство данных, а ограничены многообразием, хотя форма и размерность этого многообразия, как правило, заранее неизвестны.

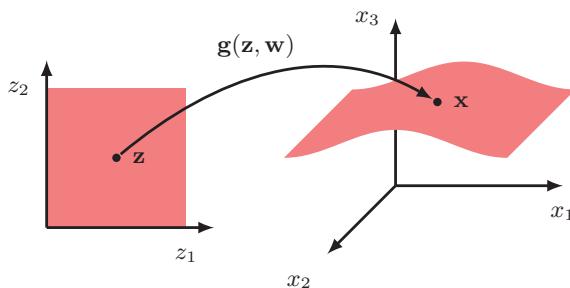


РИС. 16.11 Иллюстрация отображения двумерного латентного пространства $z = (z_1, z_2)$ в трехмерное пространство данных $x = (x_1, x_2, x_3)$ с помощью нелинейной функции $x = g(z, w)$, представленной нейронной сетью с вектором параметров w

Одна из проблем этой схемы заключается в присвоении нулевой плотности вероятности любому вектору данных, который не лежит *точно* в многообра-

зии, что является проблемой для обучения на основе градиентов, поскольку для любого реалистичного набора данных функция правдоподобия будет равна нулю в каждой из точек данных и постоянна при малых изменениях w . Для решения этой проблемы используется подход, применявшийся ранее при решении задач регрессии и классификации, а именно определение условного распределения во всем пространстве данных, параметры которого задаются выходом нейронной сети. Если, например, x представляет собой вектор непрерывных переменных, то условное распределение можно выбрать гауссовым:

$$p(x|z, w) = \mathcal{N}(x|g(z, w), \sigma^2 I), \quad (16.80)$$

где нейронная сеть $g(z, w)$ имеет линейные функции активации выходных звеньев, при этом $g \in \mathbb{R}^D$. Генеративная модель задается латентным распределением по z вместе с условным распределением по x и может быть представлена простой графовой моделью, показанной на рис. 16.12.

РИС. 16.12 Графовая модель распределения, заданного в (16.77) и (16.80), которые вместе определяют совместное распределение $p(x, z) = p(x|z)p(z)$



Обратите внимание, что получение независимых выборок из этого распределения не требует особых усилий и вычислительных затрат. Сначала стандартными методами (см. раздел 14.1.2) берется выборка из гауссова распределения (16.77). Затем это значение используется в качестве входного для нейронной сети, что дает на выходе значение $g(z, w)$. Наконец, берется выборка из гауссова распределения со средним значением $g(z, w)$ и ковариацией $\sigma^2 I$, как определено в (16.80). Этот трехэтапный процесс можно повторять для получения множества независимых выборок.

Комбинация распределения латентных переменных $p(z)$ и условного распределения $p(x|z)$ определяет маргинальное распределение в пространстве данных, которое имеет вид:

$$p(x) = \int p(z)p(x|z)dz. \quad (16.81)$$

Это можно проиллюстрировать на простом примере с одномерным латентным пространством и двумерным пространством данных на рис. 16.13. ((a) Априорное распределение в латентном пространстве задано гауссовым распределением с нулевым средним значением и единичной дисперсией. (b) Три крайних левых графика показывают примеры условного гауссова распределения $p(x|z)$ для различных значений z , а крайний правый график – маргинальное распределение $p(x)$. Нелинейная функция $g(z)$, определяющая среднее значение условного распределения, задается $g_1(z) = \sin(z)$, $g_2(z) =$

$\cos(z)$ и, следовательно, очерчивает круг в пространстве данных. Стандартное отклонение условного распределения равно $\sigma = 0,3$. [По работе (Prince, 2020) с разрешения автора].)

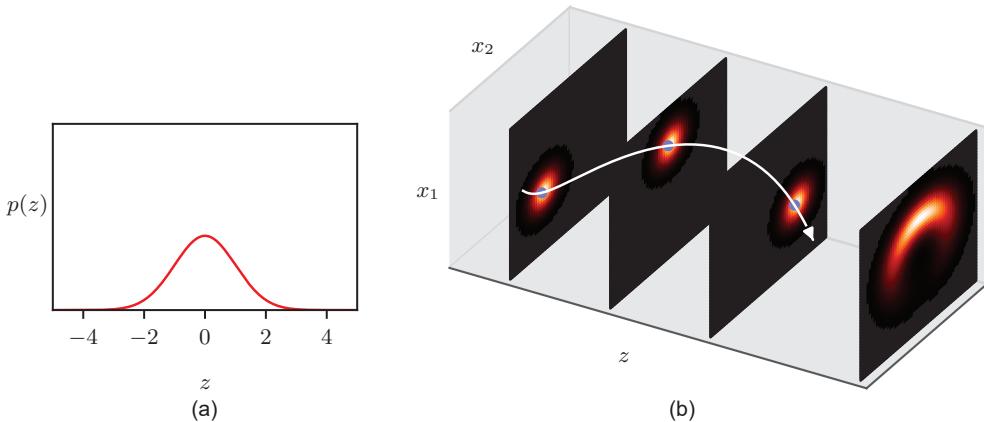


РИС. 16.13 Иллюстрация нелинейной модели латентных переменных для одномерного латентного пространства и двумерного пространства данных

16.4.2. Функция правдоподобия

Ранее уже выяснилось, что выборки из этой нелинейной модели с латентными переменными брать несложно. Теперь предположим, что необходимо подогнать модель под наблюдаемый набор данных путем максимизации функции правдоподобия. Вероятность получается из правил произведения и суммы вероятностей за счет интегрирования по \mathbf{z} :

$$\begin{aligned} p(\mathbf{x}|\mathbf{w}) &= \int p(\mathbf{x}|\mathbf{z}, \mathbf{w})p(\mathbf{z})d\mathbf{z} \\ &= \int \mathcal{N}(\mathbf{x}|\mathbf{g}(\mathbf{z}, \mathbf{w}), \sigma^2 \mathbf{I}) \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) d\mathbf{z}. \end{aligned} \quad (16.82)$$

Хотя оба распределения внутри интеграла являются гауссовыми, интеграл является не поддающимся анализу из-за сильно нелинейной функции $\mathbf{g}(\mathbf{z}, \mathbf{w})$, определяемой нейронной сетью.

Один из подходов к оценке функции правдоподобия состоит в получении выборок из распределения латентного пространства и использовании их для аппроксимации (16.82) с помощью

$$p(\mathbf{x}|\mathbf{w}) \approx \frac{1}{K} \sum_{i=1}^K p(\mathbf{x}|\mathbf{z}_i, \mathbf{w}), \quad (16.83)$$

где $\mathbf{z}_i \sim p(\mathbf{z})$. Получается, что распределение по \mathbf{z} выражается в виде смеси гауссовых распределений с фиксированными коэффициентами смешивания, равными $1/K$, и в пределе бесконечного числа выборок это дает истинную функцию правдоподобия. Однако значение K , необходимое для эффектив-

ного обучения, как правило, слишком велико, чтобы быть практическим. Для понимания причины рассмотрим три изображения рукописных цифр, показанные на рис. 16.14. Здесь: (а) – исходное изображение, (б) – поврежденное изображение с удаленной частью штриха, (с) – исходное изображение, сдвинутое на полпикселя вниз и на полпикселя вправо. Изображение (б) ближе к (а) с точки зрения правдоподобия, хотя изображение (с) гораздо ближе к (а) по внешнему виду. Предположим, что изображение (а) представляет собой вектор \mathbf{x} , для которого необходимо оценить функцию правдоподобия. Если бы обучаемая модель создала изображение (б), ее можно было бы считать плохой моделью, так как это изображение плохо представляет цифру «2», и поэтому ему следует присвоить гораздо меньшее правдоподобие. И наоборот, изображение (с), полученное путем сдвига цифры в (а) вниз и вправо на полпикселя, является хорошим примером цифры «2» и поэтому должно иметь высокую правдоподобность. Поскольку распределение (16.80) является гауссовым, функция правдоподобия пропорциональна экспоненте отрицательного квадрата расстояния между выходом сети и вектором данных \mathbf{x} . Однако квадрат расстояния между (а) и (б) равен 0,0387, тогда как квадрат расстояния между (а) и (с) составляет 0,2693. Таким образом, если параметр дисперсии σ^2 установить на достаточно малое значение, чтобы изображение (б) имело низкую вероятность, то изображение (с) будет иметь еще более низкую вероятность. Даже если модель будет хорошо генерировать цифры, потребуется рассмотреть очень большое количество выборок для \mathbf{z} , прежде чем встретится цифра, достаточно близкая к (а). Поэтому для обучения нелинейных моделей латентных переменных нужны более сложные методы, которые можно было бы использовать в практических приложениях. Прежде чем перейти к описанию таких методов, обсудим вкратце некоторые соображения по поводу дискретных пространств данных.

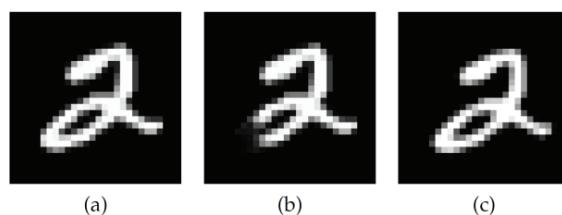


РИС. 16.14 Три примера изображений рукописных цифр, иллюстрирующие необходимость выборки из латентного пространства для оценки функции правдоподобия и большого количества выборок. [По материалам Doersch (2016) с разрешения автора]

16.4.3. Дискретные данные

Если наблюдаемый набор данных состоит из независимых бинарных переменных, тогда можно использовать условное распределение вида

$$p(\mathbf{x}|\mathbf{z}, \mathbf{w}) = \prod_{i=1}^D g_i(\mathbf{z}, \mathbf{w})^{x_i} (1 - g_i(\mathbf{z}, \mathbf{w}))^{1-x_i}, \quad (16.84)$$

где $g_i(\mathbf{z}, \mathbf{w}) = \sigma(a_i(\mathbf{z}, \mathbf{w}))$ обозначает активацию выходного элемента i , функция активации $\sigma(\cdot)$ задается логистической сигмоидной функцией, а $a_i(\mathbf{z}, \mathbf{w})$ – это предварительная активация для выходного элемента i . Точно так же для одноточечных кодированных категорийных переменных можно использовать мультиномиальное распределение:

$$p(\mathbf{x}|\mathbf{z}, \mathbf{w}) = \prod_{i=1}^D g_i(\mathbf{z}, \mathbf{w})^{x_i}, \quad (16.85)$$

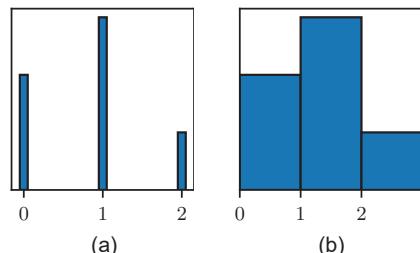
где

$$g_i(\mathbf{z}, \mathbf{w}) = \frac{\exp(a_i(\mathbf{z}, \mathbf{w}))}{\sum_j \exp(a_j(\mathbf{z}, \mathbf{w}))} \quad (16.86)$$

является функцией активации softmax. Также можно рассматривать комбинации дискретных и непрерывных переменных за счет получения произведения соответствующих условных распределений.

На практике непрерывные переменные представлены дискретными значениями, например в изображениях яркость красного, зеленого и синего каналов может быть выражена 8-битными числами, представляющими значения $\{0, \dots, 255\}$. Это может вызвать проблемы при использовании очень гибких моделей на основе глубоких нейронных сетей, так как функция правдоподобия может обратиться в ноль, если плотность свалится на одно или несколько дискретных значений. Этую проблему можно решить с помощью техники, называемой *деквантованием* (*dequantization*), которая заключается в добавлении шума к переменным, обычно взятого из равномерного распределения в области между последовательными дискретными значениями, как показано на рис. 16.15. Обучающий набор подвергается деквантованию путем замены каждого наблюдаемого значения выборкой, взятой случайным образом из соответствующего непрерывного распределения для этого дискретного значения, и это снижает риск того, что модель выявит патологическое решение.

РИС. 16.15 Схематическая иллюстрация деквантования, показывающая (a) дискретное распределение по одной переменной и (b) связанное с ним деквантованное непрерывное распределение



16.4.4. Четыре метода генеративного моделирования

На примере нелинейных моделей с латентными переменными на основе глубоких нейронных сетей было показано, что они представляют собой

очень гибкую основу для построения генеративных моделей. Благодаря универсальности нейросетевого преобразования такие модели в принципе способны с высокой точностью аппроксимировать практически любое желаемое распределение. Более того, после обучения такие модели способны генерировать выборки из распределения с помощью эффективного процесса без итераций. Однако вместе с тем были выявлены и некоторые проблемы, связанные с обучением таких моделей, что требует разработки более сложных методов, чем требуемые для линейных моделей. Было предложено множество таких методов, и каждый из них имеет свои сильные стороны и ограничения. В целом их можно разделить на четыре следующих метода.

При использовании *генеративных состязательных сетей* (*generative adversarial networks*, GAN) (см. главу 17) ослабляется требование об инвертируемости отображения сети, что обеспечивает более низкую размерность латентного пространства по сравнению с пространством данных. Кроме того, здесь отсутствует понятие функции правдоподобия, а вместо нее вводится вторая нейронная сеть, функция которой заключается в предоставлении обучающего сигнала для генеративной сети. Из-за отсутствия четко определенной функции правдоподобия процедура обучения может быть хрупкой, но после обучения генерировать выборки из модели очень просто, а результаты обучения отличаются высоким качеством.

В концепции *вариационных автокодировщиков* (*variational autoencoders*, VAE) (см. главу 19) также присутствует вторая нейронная сеть, роль которой заключается в аппроксимации апостериорного распределения по латентным переменным, что позволяет оценить приближенную функцию правдоподобия. Процесс обучения более надежен, чем при использовании GAN, а выборка из обученной модели проста, однако получить результаты высокого качества может быть сложнее.

При использовании метода *нормализующих потоков* (*normalizing flows*) (см. главу 18) устанавливается размерность латентного пространства, равная размерности пространства данных, а затем модифицируется генеративная нейронная сеть так, чтобы она стала инвертируемой. Требование инвертируемости сети ограничивает ее функциональную форму, но позволяет оценивать функцию правдоподобия без аппроксимации, а также обеспечивает эффективную выборку.

Наконец, в *диффузионных моделях* (*diffusion models*) (см. главу 20) используется сеть, которая учится преобразовывать выборку из предшествующего распределения в выборку из распределения данных с помощью последовательности шагов *сглаживания*, или снижения шума (*denoising*). Это позволяет достичь наилучшей производительности во многих приложениях, хотя стоимость выборки может быть высокой из-за многократных проходов алгоритма сглаживания по сети.

Эти подходы подробно рассматриваются в последних четырех главах этой книги.

Упражнения

- 16.1** (**) В этом упражнении с помощью доказательства по индукции нужно доказать, что линейная проекция на M -мерное подпространство, максимизирующая дисперсию проецируемых данных, определяется M собственными векторами ковариационной матрицы данных \mathbf{S} , заданной в (16.3), соответствующими M наибольшим собственным значениям. В разделе 16.1 этот результат был доказан для $M = 1$. Теперь предположим, что результат имеет место для некоторого общего значения M , и докажем, что он, как следствие, имеет место и для размерности $M + 1$. Для этого сначала нужно установить производную дисперсии проецируемых данных относительно вектора \mathbf{u}_{M+1} , задающего новое направление в пространстве данных, равной нулю. Это следует сделать с учетом ограничений на то, что \mathbf{u}_{M+1} ортогонален существующим векторам $\mathbf{u}_1, \dots, \mathbf{u}_M$, а также нормализован на единицу длины. Для выполнения этих ограничений используйте множители Лагранжа (см. приложение С). Затем воспользуйтесь свойствами ортонормированности векторов $\mathbf{u}_1, \dots, \mathbf{u}_M$, чтобы доказать, что новый вектор \mathbf{u}_{M+1} является собственным вектором \mathbf{S} . Наконец, докажите, что дисперсия максимальна, если в качестве собственного вектора выбран вектор, соответствующий собственному значению λ_{M+1} , где собственные значения упорядочены по убыванию.
- 16.2** (**) Докажите, что минимальное значение меры ошибки J в РСА, заданной в (16.15) относительно \mathbf{u}_i , при соблюдении ограничений ортонормированности (16.7) получается, когда \mathbf{u}_i являются собственными векторами ковариационной матрицы данных \mathbf{S} . Для этого введем матрицу \mathbf{H} множителей Лагранжа, по одному на каждое ограничение, так что модифицированная мера ошибки в матричной форме будет выглядеть как

$$\tilde{J} = \text{Tr}\{\hat{\mathbf{U}}^T \mathbf{S} \hat{\mathbf{U}}\} + \text{Tr}\{\mathbf{H}(\mathbf{I} - \hat{\mathbf{U}}^T \hat{\mathbf{U}})\}, \quad (16.87)$$

где $\hat{\mathbf{U}}$ – это матрица размерности $D \times (D - M)$, столбцы которой заданы \mathbf{u}_i . Теперь необходимо минимизировать \tilde{J} относительно $\hat{\mathbf{U}}$ и доказать, что решение удовлетворяет $\mathbf{S}\hat{\mathbf{U}} = \hat{\mathbf{U}}\mathbf{H}$. Очевидно, что одно из возможных решений состоит в том, что столбцы $\hat{\mathbf{U}}$ являются собственными векторами \mathbf{S} , и в этом случае \mathbf{H} – это диагональная матрица с соответствующими собственными значениями. Чтобы получить общее решение, докажите, что \mathbf{H} можно считать симметричной матрицей, и с помощью ее разложения по собственным векторам докажите, что общее решение $\mathbf{S}\hat{\mathbf{U}} = \hat{\mathbf{U}}\mathbf{H}$ дает то же значение для \tilde{J} , что и конкретное решение, в котором столбцы $\hat{\mathbf{U}}$ являются собственными векторами \mathbf{S} . Поскольку все эти решения эквивалентны, удобнее всего выбрать решение по собственным векторам.

- 16.3** (*) Докажите, что собственные векторы, определенные в (16.30), нормализованы на единицу длины, исходя из предположения, что собственные векторы \mathbf{v}_i имеют единичную длину.
- 16.4** (*) Предположим, что нулевое среднее значение с единичной ковариацией распределения в латентном пространстве (16.31) вероятностной модели РСА будет заменено на общее гауссово распределение вида $\mathcal{N}(\mathbf{z} | \mathbf{m}, \Sigma)$. Переопределив параметры модели, докажите, что это приводит к идентичной модели для маргинального распределения $p(\mathbf{x})$ по наблюдаемым переменным при любом правильном выборе \mathbf{m} и Σ .
- 16.5** (**) Пусть \mathbf{x} – это D -мерная случайная переменная с гауссовым распределением, заданным $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \Sigma)$, и пусть имеется M -мерная случайная переменная $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$, где \mathbf{A} – это матрица $M \times D$. Докажите, что \mathbf{y} также имеет гауссово распределение, и найдите выражения для ее среднего значения и ковариации. Проанализируйте форму этого гауссова распределения для $M < D$, для $M = D$ и для $M > D$.
- 16.6** (**) Используя результаты (2.122) и (2.123) для среднего значения и ковариации общего распределения, выведите результат (16.35) для маргинального распределения $p(\mathbf{x})$ в вероятностной модели РСА.
- 16.7** (*) Постройте направленный вероятностный граф для вероятностной модели РСА, описанной в разделе 16.2, где компоненты наблюдаемой переменной \mathbf{x} показаны в явном виде как отдельные узлы. Тем самым подтвердите, что вероятностная модель РСА имеет ту же независимую структуру, что и наивная модель Байеса, рассмотренная в разделе 11.2.3.
- 16.8** (**) Используя результат (3.100), докажите, что апостериорное распределение $p(\mathbf{z} | \mathbf{x})$ для вероятностной модели РСА задается в (16.43).
- 16.9** (*) Докажите, что максимизация логарифмического правдоподобия (16.44) для вероятностной модели РСА с учетом параметра $\boldsymbol{\mu}$ дает результат $\boldsymbol{\mu}_{\text{ML}} = \bar{\mathbf{x}}$, где $\bar{\mathbf{x}}$ – это среднее значение векторов данных.
- 16.10** (**) Оценивая вторые производные функции логарифмического правдоподобия (16.44) для вероятностной модели РСА по параметру $\boldsymbol{\mu}$, докажите, что стационарная точка $\boldsymbol{\mu}_{\text{ML}} = \bar{\mathbf{x}}$ представляет собой единственный максимум.
- 16.11** (**) Докажите, что в пределе $\sigma^2 \rightarrow 0$ апостериорное среднее значение для вероятностной модели РСА становится ортогональной проекцией на главное подпространство, как и в обычной РСА.
- 16.12** (**) Докажите, что в случае $\sigma^2 > 0$ апостериорное среднее значение в вероятностной модели РСА смешено к началу координат относительно ортогональной проекции.

- 16.13** (**) Докажите, что оптимальная реконструкция точки данных при вероятностном PCA в соответствии с затратами на проекцию наименьших квадратов при обычном PCA определяется как

$$\tilde{\mathbf{X}} = \mathbf{W}_{\text{ML}}(\mathbf{W}_{\text{ML}}^T \mathbf{W}_{\text{ML}})^{-1} \mathbf{M} \mathbb{E}[\mathbf{z} | \mathbf{x}]. \quad (16.88)$$

- 16.14** (*) Число независимых параметров в ковариационной матрице для

вероятностной модели PCA с M -мерным латентным пространством и D -мерным пространством данных определяется в (16.52). Подтвердите, что для $M = D - 1$ число независимых параметров такое же, как у гауссова распределения с общей ковариацией, а для $M = 0$ – как у гауссова распределения с изотропной ковариацией.

- 16.15** (*) Выведите выражение для числа независимых параметров в модели факторного анализа, описанной в разделе 16.2.4.

- 16.16** (**) Докажите, что модель факторного анализа, описанная в разделе 16.2.4, инвариантна при вращении координат латентного пространства.

- 16.17** (**) Рассмотрим линейную гауссову модель латентных переменных с распределением в латентном пространстве $p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I})$ и условным распределением для наблюдаемой переменной $p(\mathbf{x} | \mathbf{z}) = \mathcal{N}(\mathbf{x} | \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Phi})$, где $\boldsymbol{\Phi}$ – это произвольная симметричная положительно определенная ковариационная матрица шума. Теперь предположим, что выполняется несингулярное линейное преобразование переменных данных $\mathbf{x} \rightarrow \mathbf{Ax}$, где \mathbf{A} – это матрица $D \times D$. Если $\boldsymbol{\mu}_{\text{ML}}$, \mathbf{W}_{ML} и $\boldsymbol{\Phi}_{\text{ML}}$ представляют собой решение максимального правдоподобия, соответствующее исходным не преобразованным данным, докажите, что $\mathbf{A}\boldsymbol{\mu}_{\text{ML}}$, \mathbf{AW}_{ML} и $\mathbf{A}\boldsymbol{\Phi}_{\text{ML}}\mathbf{A}^T$ представляют собой соответствующее решение максимального правдоподобия для преобразованного набора данных. Наконец, докажите, что форма модели сохраняется в двух случаях. (i) \mathbf{A} – это диагональная матрица и $\boldsymbol{\Phi}$ – это диагональная матрица. Этот случай соответствует факторному анализу. Преобразованная $\boldsymbol{\Phi}$ остается диагональной, и, следовательно, факторный анализ является ковариантным при компонентном перескеллировании переменных данных. (ii) \mathbf{A} ортогональна, а $\boldsymbol{\Phi}$ пропорциональна единичной матрице, так что $\boldsymbol{\Phi} = \sigma^2 \mathbf{I}$. Это соответствует вероятностному PCA. Преобразованная матрица $\boldsymbol{\Phi}$ остается пропорциональна единичной матрице, и, следовательно, вероятностный PCA ковариантен при повороте осей пространства данных, как и в случае обычного PCA.

- 16.18** (*) Докажите, что логарифмическая функция правдоподобия для модели с непрерывными латентными переменными может быть записана в виде суммы двух членов в форме (16.57), где члены определяются через (16.58) и (16.59). Это можно сделать с помощью правила произведения вероятностей в форме

$$p(\mathbf{x}, \mathbf{z} | \mathbf{w}) = p(\mathbf{z} | \mathbf{x}, \mathbf{w})p(\mathbf{x} | \mathbf{w}), \quad (16.89)$$

а затем подставить $p(\mathbf{x}, \mathbf{z} | \mathbf{w})$ в (16.58).

- 16.19** (*) Докажите, что для набора i.i.d.-данных нижняя граница доказательства (ELBO) имеет вид (16.63).
- 16.20** (**) Постройте направленную вероятностную графовую модель, представляющую дискретную смесь вероятностных моделей РСА, где каждая модель РСА имеет свои собственные значения \mathbf{W} , $\boldsymbol{\mu}$ и σ^2 . Теперь постройте модифицированный график, в котором значения этих параметров распределены между компонентами смеси.
- 16.21** (**) Выведите уравнения шага M по (16.68) и (16.69) для вероятностной модели РСА посредством максимизации ожидаемой логарифмической функции правдоподобия для полных данных, заданной в (16.65).
- 16.22** (***) Одним из преимуществ вероятностной формулировки анализа главных компонент является возможность его применения к набору данных с некоторыми отсутствующими значениями при условии, что они отсутствуют случайным образом. Выведите ЕМ-алгоритм для максимизации функции правдоподобия для вероятностной модели РСА в такой ситуации. Обратите внимание, что $\{\mathbf{z}_n\}$, а также отсутствующие значения данных, которые являются компонентами векторов $\{\mathbf{x}_n\}$, теперь являются латентными переменными. Докажите, что в частном случае всех наблюдаемых значений данных это сводится к ЕМ-алгоритму для вероятностного РСА, полученному в разделе 16.3.2.
- 16.23** (**) Пусть \mathbf{W} – это матрица $D \times M$, столбцы которой определяют линейное подпространство размерности M , вложенное в пространство данных размерности D , и пусть $\boldsymbol{\mu}$ – это вектор размерности D . При наличии набора данных $\{\mathbf{x}_n\}$, где $n = 1, \dots, N$, можно аппроксимировать точки данных линейным отображением из набора M -мерных векторов $\{\mathbf{z}_n\}$, так что \mathbf{x}_n аппроксимируется по $\mathbf{W}\mathbf{z}_n + \boldsymbol{\mu}$. Соответствующая стоимость перестройки по сумме квадратов определяется как

$$J = \sum_{n=1}^N \|\mathbf{x}_n - \boldsymbol{\mu} - \mathbf{W}\mathbf{z}_n\|^2. \quad (16.90)$$

Сначала докажите, что минимизация J относительно $\boldsymbol{\mu}$ приводит к аналогичному выражению с заменой \mathbf{x}_n и \mathbf{z}_n на переменные с нулевым средним $\mathbf{x}_n - \bar{\mathbf{x}}$ и $\mathbf{z}_n - \bar{\mathbf{z}}$ соответственно, где $\bar{\mathbf{x}}$ и $\bar{\mathbf{z}}$ обозначают выборочные средние значения. Затем докажите, что минимизация J относительно \mathbf{z}_n , где \mathbf{W} фиксировано, приводит к шагу E в РСА (16.70), а минимизация J относительно \mathbf{W} , где $\{\mathbf{z}_n\}$ фиксировано, приводит к шагу M в РСА (16.71).

- 16.24** (**) Выведите формулы (16.72) и (16.73) для шага Е в ЕМ-алгоритме для факторного анализа. Обратите внимание, что, исходя из результата упражнения 16.26, параметр μ можно заменить на выборочное среднее \bar{x} .
- 16.25** (**) Запишите выражение для ожидаемой логарифмической функции правдоподобия для модели факторного анализа с полным набором данных и выведите соответствующие уравнения шага М по (16.75) и (16.76).
- 16.26** (**) Используя вторые производные, докажите, что единственная стационарная точка функции правдоподобия для модели факторного анализа из раздела 16.2.4 относительно параметра μ задается выборочным средним значением, определенным в (16.1). Кроме того, докажите, что эта стационарная точка является максимумом.

Глава 17

Генеративные состязательные сети

Генеративные модели используют алгоритмы машинного обучения для обучения распределению на основе набора обучающих данных, а затем генерируют новые примеры на основе этого распределения. Например, генеративная модель может быть обучена на изображениях животных, а затем использована для создания новых изображений животных. Подобную генеративную модель можно представить в виде распределения $p(\mathbf{x}|\mathbf{w})$, где \mathbf{x} – это вектор в пространстве данных, а \mathbf{w} – это обучаемые параметры модели. Во многих случаях интерес представляют условные генеративные модели вида $p(\mathbf{x}|\mathbf{c}, \mathbf{w})$, где \mathbf{c} – это вектор обуславливающих переменных. В случае использования генеративной модели для работы с изображениями животных можно указать, что сгенерированное изображение должно быть изображением определенного животного, например кошки или собаки, что определяется значением \mathbf{c} .

В реальных задачах, таких как генерация изображений, распределения чрезвычайно сложны, и поэтому внедрение глубокого обучения значительно улучшило производительность генеративных моделей. Ранее при обсуждении авторегрессионных моделей больших языков на основе трансформеров (см. главу 12) уже был рассмотрен важный класс глубоких генеративных моделей. Кроме того, были выделены четыре важных класса генеративных моделей, основанных на нелинейных моделях латентных переменных (см. раздел 16.4.4), и в этой главе рассматривается первый из них, называемый генеративными состязательными сетями. Остальные три концепции будут рассмотрены в последующих главах.

17.1. Состязательное обучение

Рассмотрим генеративную модель на основе нелинейного преобразования из латентного пространства \mathbf{z} в пространство данных \mathbf{x} . Введем латентное распределение $p(\mathbf{z})$, которое может иметь вид простого гауссова распределения

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}), \quad (17.1)$$

вместе с нелинейным преобразованием $\mathbf{x} = \mathbf{g}(\mathbf{z}, \mathbf{w})$, определяемым глубокой нейронной сетью с обучаемыми параметрами \mathbf{w} , которую называют *генератором* (*generator*). Вместе они в неявном виде определяют распределение по \mathbf{x} , и целью является подгонка этого распределения под набор данных обучающих примеров $\{\mathbf{x}_n\}$, где $n = 1, \dots, N$. Вместе с тем определение \mathbf{w} путем оптимизации функции правдоподобия невозможно, так как в общем случае она не может быть вычислена в замкнутой форме. Ключевая идея *генеративных состязательных сетей* (*generative adversarial networks, GAN*) (Goodfellow et al., 2014; Ruthotto and Haber, 2021) заключается в добавлении второй сети, *дискриминатора* (*discriminator*), которая обучается совместно с сетью-генератором и обеспечивает обучающий сигнал для обновления весов генератора. Это показано на рис. 17.1.

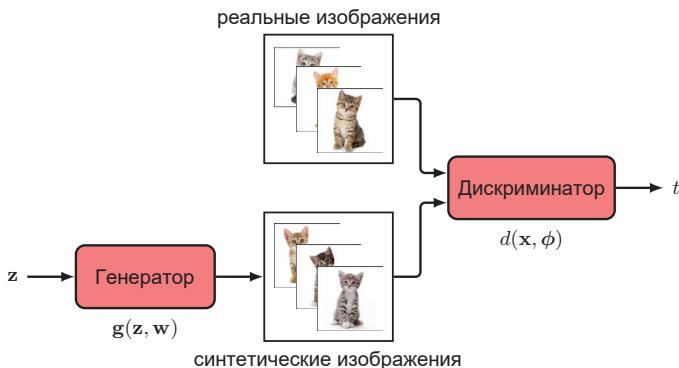


РИС. 17.1 Схематическое изображение GAN, где нейронная сеть-дискриминатор $d(\mathbf{x}, \varphi)$ обучается различать реальные выборки из обучающего набора, в данном случае изображений котят, и синтетические выборки, созданные сетью-генератором $\mathbf{g}(\mathbf{z}, \mathbf{w})$. Генератор стремится максимизировать ошибку сети дискриминатора посредством создания реалистичных изображений, в то время как сеть дискриминатора пытается минимизировать ту же ошибку, становясь лучше в определении реальных и синтетических примеров

Цель работы сети дискриминатора – определить разницу между реальными примерами из набора данных и синтетическими, или «фальшивыми», примерами, созданными сетью генератора, поэтому она обучается путем минимизации обычной функции ошибки классификации. И напротив, цель сети-генератора – максимизировать эту ошибку, синтезируя примеры из того же распределения, что и обучающий набор. Таким образом, сети генератора и дискриминатора работают друг против друга, отсюда появился термин «состязательный». Это пример *игры с нулевой суммой*, где любой выигрыш одной сети означает проигрыш другой. Это позволяет сети дискриминатора предоставлять обучающий сигнал, который может быть использован для обучения сети генератора, что превращает задачу моделирования плотности без наблюдения в форму обучения с наблюдением.

17.1.1. Функция потерь

Для уточнения определим бинарную целевую переменную:

$$t = 1, \text{ реальные данные}, \quad (17.2)$$

$$t = 0, \text{ синтетические данные}. \quad (17.3)$$

Сеть дискриминатора имеет один выходной элемент с логистической сигмоидной функцией активации, выход которой представляет собой вероятность того, что вектор данных \mathbf{x} является реальным:

$$P(t = 1) = d(\mathbf{x}, \boldsymbol{\varphi}). \quad (17.4)$$

Для обучения сети дискриминатора используется стандартная функция ошибки перекрестной энтропии в форме

$$E(\mathbf{w}, \boldsymbol{\varphi}) = -\frac{1}{N} \sum_{n=1}^N \{t_n \ln d_n + (1 - t_n) \ln(1 - d_n)\}, \quad (17.5)$$

где $d_n = d(\mathbf{x}_n, \boldsymbol{\varphi})$ – это выход сети дискриминатора для входного вектора n (см. раздел 1.2.4), нормализованный по общему количеству точек данных. Обучающее множество состоит как из реальных примеров данных, обозначаемых \mathbf{x}_n , так и из синтетических примеров, задаваемых выходом генераторной сети $\mathbf{g}(\mathbf{z}_n, \mathbf{w})$, где \mathbf{z}_n – это случайная выборка из распределения латентного пространства $p(\mathbf{z})$. Поскольку $t_n = 1$ для реальных примеров и $t_n = 0$ для синтетических примеров, функцию ошибки (17.5) можно записать в виде

$$\begin{aligned} E_{\text{GAN}}(\mathbf{w}, \boldsymbol{\varphi}) = & -\frac{1}{N_{\text{real}}} \sum_{n \in \text{real}} \ln d(\mathbf{x}_n, \boldsymbol{\varphi}) \\ & -\frac{1}{N_{\text{synth}}} \sum_{n \in \text{synth}} \ln(1 - d(\mathbf{g}(\mathbf{z}_n, \mathbf{w}), \boldsymbol{\varphi})), \end{aligned} \quad (17.6)$$

где обычно число N_{real} реальных точек данных равно числу N_{synth} синтетических точек данных. Эта комбинация сетей генератора и дискриминатора (см. главу 7) может быть обучена в сквозном режиме с помощью стохастического градиентного спуска с оценкой градиентов методом обратного распространения. Однако необычным аспектом здесь является состязательное обучение, при котором ошибка минимизируется по отношению к $\boldsymbol{\varphi}$, но максимизируется по отношению к \mathbf{w} .

Эта максимизация может быть выполнена с помощью стандартных градиентных методов с изменением знака градиента, так что обновления параметров принимают вид

$$\Delta \boldsymbol{\varphi} = -\lambda \nabla_{\boldsymbol{\varphi}} E_n(\mathbf{w}, \boldsymbol{\varphi}), \quad (17.7)$$

$$\Delta \mathbf{w} = \lambda \nabla_{\mathbf{w}} E_n(\mathbf{w}, \boldsymbol{\varphi}), \quad (17.8)$$

где $E_n(\mathbf{w}, \boldsymbol{\varphi})$ обозначает ошибку, определенную для точки данных n или, в более общем случае, для мини-батча точек данных. Обратите внимание, что два члена в (17.7) и (17.8) имеют разные знаки, поскольку дискриминатор обучается на уменьшение коэффициента ошибок, а генератор – на его увеличение. На практике обучение чередуется между обновлением параметров генеративной сети и обновлением параметров дискриминантной сети, в каждом случае делается всего один шаг градиентного спуска с использованием мини-батча, после чего генерируется новый набор синтетических выборок. Если генератору удается найти идеальное решение, то дискриминантная сеть не сможет отличить реальные данные от синтетических и, следовательно, всегда будет выдавать значение 0,5. После обучения GAN сеть дискриминатора упраздняется, а сеть генератора может быть использована для синтеза новых примеров в пространстве данных путем выборки из скрытого пространства и распространения этих выборок через обученную сеть генератора. Получается, можно доказать, что для генеративных и дискриминативных сетей, обладающих неограниченной гибкостью, полностью оптимизированная GAN будет иметь генеративное распределение с точным совпадением с распределением данных (см. упражнение 17.1). Некоторые впечатляющие примеры синтетических изображений лиц, созданных с помощью GAN, показаны на рис. 1.3.

Рассмотренная выше модель GAN генерирует выборки из безусловного распределения $p(\mathbf{x})$. Например, она может генерировать синтетические изображения собак, если ее обучить на изображениях собак. Также можно создать условные GAN (Mirza and Osindero, 2014), которые генерируют выборки из условного распределения $p(\mathbf{x}|\mathbf{c})$, где условный вектор \mathbf{c} может, например, представлять различные породы собак. Для этого и сеть генератора, и сеть дискриминатора принимают \mathbf{c} в качестве дополнительного входа, а для обучения используются маркированные примеры изображений, состоящие из пар $\{\mathbf{x}_n, \mathbf{c}_n\}$. После обучения GAN изображения нужного класса могут быть сгенерированы путем задания \mathbf{c} для соответствующего вектора класса. По сравнению с обучением отдельных GAN для каждого класса это преимущество заключается в возможности совместного обучения общих внутренних представлений для всех классов, что повышает эффективность использования данных.

17.1.2. Практическое обучение GAN

Несмотря на высокое качество результатов, успешное обучение GAN является непростой задачей (см. упражнение 17.2) в силу состязательного характера обучения. Кроме того, в отличие от стандартной минимизации функции ошибки, не существует метрики достижения цели, поскольку в процессе обучения цель может как увеличиваться, так и уменьшаться.

Одна из потенциальных проблем называется *склонением мод распределения* (*mode collapse*), когда веса генераторной сети в процессе обучения адаптируются таким образом, что все выборки латентных переменных \mathbf{z} отображаются на подмножество возможных допустимых выходов. В крайних случаях выход может соответствовать только одному или небольшому числу

выходных значений x . Тогда дискриминатор присваивает этим экземплярам значение 0,5, и обучение прекращается. Например, GAN, обученная на рукописных цифрах, может научиться генерировать только примеры цифры «3», и, даже если дискриминатор не сможет отличить их от настоящих примеров цифры «3», он не сможет определить, что генератор не способен выдавать весь ряд цифр.

Представление о сложности обучения GAN можно получить с помощью рис. 17.2, на котором показано простое одномерное пространство данных x с выборками $\{x_n\}$, взятыми из фиксированного, но неизвестного распределения данных $p_{\text{Data}}(x)$. Также показано начальное генеративное распределение $p_G(x)$ вместе с выборками, взятыми из этого распределения. Поскольку данные и генеративное распределение отличаются очень сильно, оптимальная дискриминантная функция $d(x)$ легко поддается обучению и имеет очень крутой спад с практически нулевым градиентом вблизи реальных или синтетических выборок. Рассмотрим второй член в функции ошибки GAN (17.6). Поскольку $d(g(z, w), \varphi)$ равен нулю в области, охватываемой сгенерированными выборками, небольшие изменения параметров w генеративной сети приводят к очень незначительным изменениям на выходе дискриминатора, и поэтому градиенты малы, а процесс обучения идет медленно.

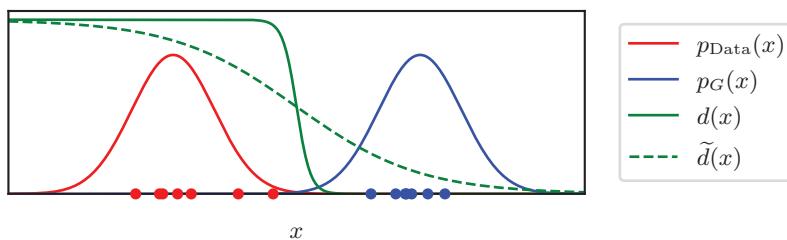


РИС. 17.2 Концептуальная иллюстрация трудностей обучения GAN на примере простого одномерного пространства данных x с фиксированным, но неизвестным распределением данных $p_{\text{Data}}(x)$ и начальным генеративным распределением $p_G(x)$. Оптимальная дискриминантная функция $d(x)$ имеет практически нулевой градиент в окрестности обучающих или синтетических точек данных, что делает обучение очень медленным. Сглаженная версия $\tilde{d}(x)$ дискриминаторной функции может привести к ускорению обучения

Эта проблема может быть решена путем использования сглаженной версии $\tilde{d}(x)$ функции дискриминатора, как показано на рис. 17.2, обеспечивая тем самым более сильный градиент для обучения сети генератора. В GAN с методом наименьших квадратов (Mao et al., 2016) сглаживание достигается путем модификации дискриминатора, чтобы он выдавал на выходе вещественное значение, а не вероятность в диапазоне (0, 1), а также замены функции ошибки перекрестной энтропии на функцию ошибки суммы квадратов. В качестве альтернативы можно использовать технику подачи шумов (*instance noise*) (Sønderby et al., 2016), которая добавляет гауссов шум как к реальным данным, так и к синтетическим выборкам, что также приводит к сглаживанию функции дискриминатора.

Для улучшения обучения было предложено множество других модификаций функции ошибки GAN и процедуры обучения (Mescheder, Geiger and Nowozin, 2018). Одно из часто используемых изменений заключается в замене члена генеративной сети в исходной функции ошибки

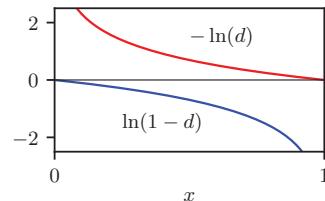
$$-\frac{1}{N_{\text{synth}}} \sum_{n \in \text{synth}} \ln(1 - d(\mathbf{g}(\mathbf{z}_n, \mathbf{w}), \boldsymbol{\varphi})) \quad (17.9)$$

на модифицированную форму

$$\frac{1}{N_{\text{synth}}} \sum_{n \in \text{synth}} \ln d(\mathbf{g}(\mathbf{z}_n, \mathbf{w}), \boldsymbol{\varphi}). \quad (17.10)$$

Несмотря на то что первая форма минимизирует вероятность нахождения поддельного изображения, вторая версия максимизирует вероятность того, что изображение настоящее. Различные свойства этих двух форм можно понять из рис. 17.3. Когда генеративное распределение $p_G(x)$ сильно отличается от истинного распределения данных $p_{\text{Data}}(x)$, величина $d(\mathbf{g}(\mathbf{z}, \mathbf{w}))$ близка к нулю, и, следовательно, первая форма имеет очень маленький градиент, тогда как вторая форма имеет большой градиент, что приводит к более быстрому обучению.

РИС. 17.3 Графики $-\ln(d)$ и $\ln(1 - d)$ показывают очень разное поведение градиентов вблизи $d = 0$ и $d = 1$



Более непосредственным способом обеспечить движение распределения генератора $p_G(x)$ к распределению данных $p_{\text{data}}(x)$ является модификация критерия ошибки с учетом степени удаленности этих двух распределений в пространстве данных. Это можно измерить с помощью *расстояния Вассерштейна* (*Wasserstein distance*), также известного как *траншейное расстояние* (*earth mover's distance*). Представим распределение $p_G(x)$ как кучу грунта, которую перемещают небольшими порциями для создания распределения $p_{\text{data}}(x)$. Метрика Вассерштейна – это общее количество перемещенного грунта, умноженное на среднее расстояние перемещения. Из множества способов перемещения кучи грунта при построении $p_{\text{data}}(x)$ для определения метрики используется тот, который дает наименьшее среднее расстояние. На практике это невозможно реализовать напрямую, поэтому для аппроксимации используется дискриминаторная сеть с вещественными выходами, а затем градиент $\nabla_x d(\mathbf{x}, \boldsymbol{\varphi})$ дискриминаторной функции по отношению к \mathbf{x} ограничивается с помощью ограничения весов, что приводит к *GAN Вассерштейна* (*Wasserstein GAN*) (Arjovsky, Chintala and Bottou, 2017). Более эффективным подходом является введение штрафа на градиент, что приводит к *градиент-*

ному штрафу GAN Вассерштейна (*gradient penalty Wasserstein GAN*) (Gulrajani et al., 2017), функция ошибки которого имеет вид:

$$E_{\text{WGAN-GP}}(\mathbf{w}, \boldsymbol{\varphi}) = -\frac{1}{N_{\text{real}}} \sum_{n \in \text{real}} \left[\ln d(\mathbf{x}_n, \boldsymbol{\varphi}) - \eta (\|\nabla_{\mathbf{x}_n} d(\mathbf{x}_n, \boldsymbol{\varphi})\|^2 - 1)^2 \right] + \frac{1}{N_{\text{synth}}} \sum_{n \in \text{synth}} \ln d(\mathbf{g}(\mathbf{z}_n, \mathbf{w}), \boldsymbol{\varphi}), \quad (17.11)$$

где η управляет относительной важностью штрафного члена.

17.2. GAN для обработки изображений

Базовая концепция GAN привела к появлению огромного количества исследовательской литературы с множеством алгоритмических разработок и многочисленных приложений. Одной из наиболее распространенных и успешных областей применения GAN является генерация изображений. Ранние модели GAN в качестве генератора и дискриминатора использовали полносвязные сети. Однако использование сверточных сетей сулит множество преимуществ, особенно для обработки изображений с высоким разрешением (см. главу 10). Сеть дискриминатора принимает на вход изображение и выдает на выходе скалярную вероятность, поэтому для нее подходит стандартная сверточная сеть. Сеть генератора должна отображать низкоразмерное латентное пространство на изображение высокого разрешения (см. раздел 10.5.3), поэтому для нее используется сеть на основе транспонирующих сверток, как показано на рис. 17.4.

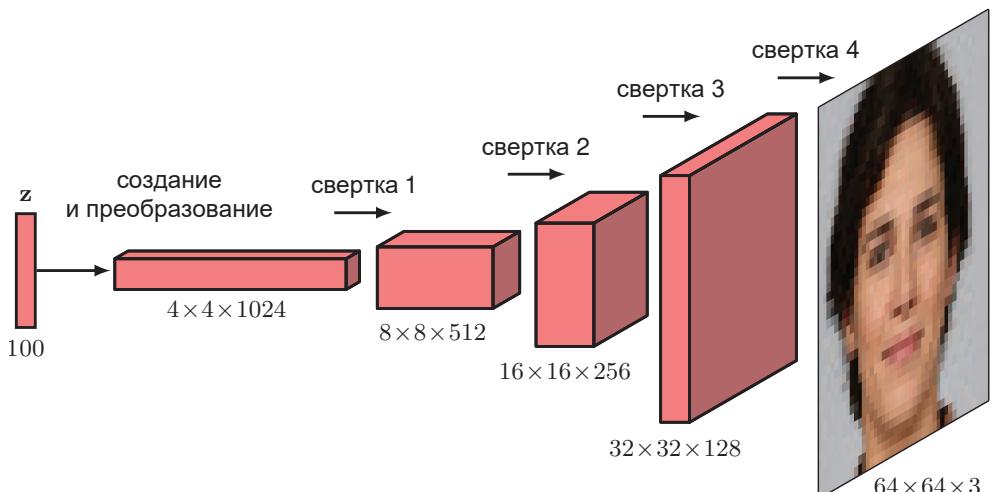


РИС. 17.4 Пример архитектуры глубокой сверточной GAN, демонстрирующей использование транспонирующих сверток для расширения размерности в последовательных элементах сети

Изображения высокого качества можно получить за счет постепенного наращивания сети генератора и сети дискриминатора, начиная с низкого разрешения, а затем последовательно добавляя новые слои, которые моделируют все более мелкие детали по мере обучения (Karras et al., 2017). Это ускоряет обучение и позволяет синтезировать изображения высокого разрешения размером 1024×1024 , отталкиваясь от изображений размером 4×4 . В качестве примера масштаба и сложности некоторых архитектур GAN рассмотрим модель GAN для генерации изображений с учетом классов под названием BigGAN, архитектура которой показана на рис. 17.5. Здесь (a) архитектура генеративной сети в модели BigGAN с более чем 70 млн параметров. (b) Детали каждого из остаточных элементов генеративной сети. Дискриминантная сеть с 88 млн параметров имеет примерно такую же структуру, за исключением того, что в ней используются слои объединения средних значений для уменьшения размерности, а не для увеличения размерности с помощью восходящей выборки.

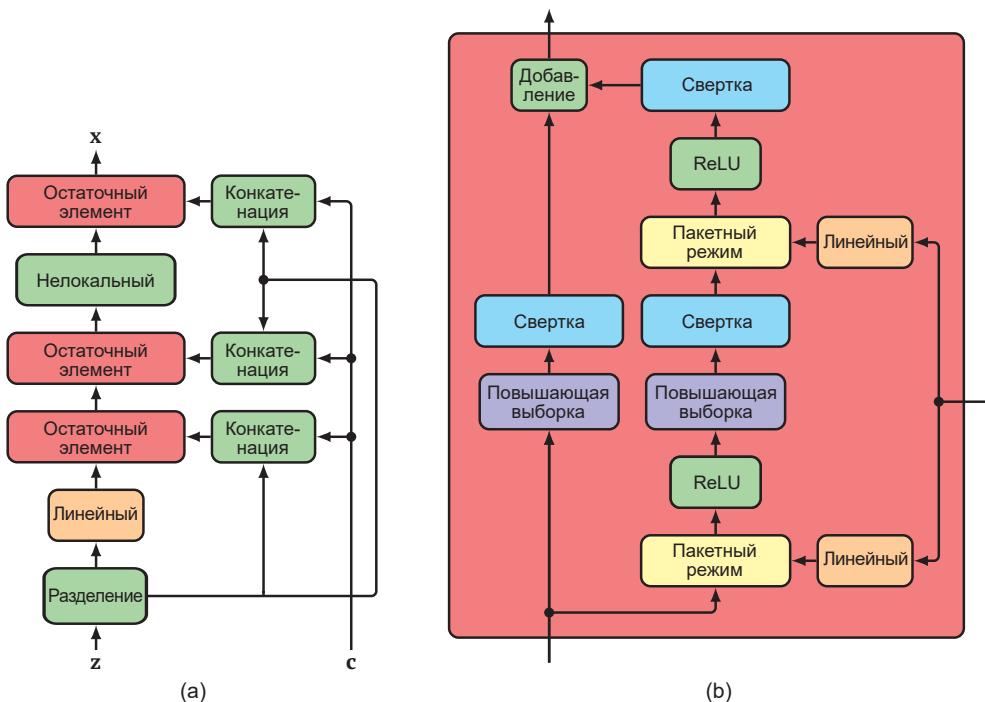


РИС. 17.5 Архитектура генеративной сети в модели BigGAN [по материалам Brock, Donahue and Simonyan (2018)]

17.2.1. CycleGAN

В качестве примера большого разнообразия GAN рассмотрим архитектуру под названием *CycleGAN* (Zhu et al., 2017). Это также иллюстрирует возможность адаптации методов глубокого обучения для решения различных типов

проблем, выходящих за рамки традиционных задач, таких как классификация и оценка плотности. Рассмотрим проблему превращения фотографии в картину Моне, изображающую ту же сцену, или наоборот. На рис. 17.6 показаны примеры пар изображений, полученных от обученной CycleGAN, которая научилась выполнять такое преобразование изображений.



Моне → фотография



Фотография → Моне

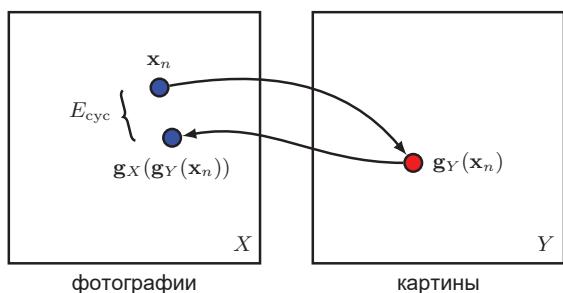
РИС. 17.6 Примеры преобразования изображений с помощью CycleGAN, демонстрирующие синтез изображения в фотографическом стиле из картины Моне (верхний ряд) и синтез изображения в стиле картины Моне из фотографии (нижний ряд). [Из (Zhu et al., 2017) с разрешения авторов]

Задача состоит в обучении двум биективным (взаимно однозначным) отображениям, одно из которых переходит из области X фотографий в область Y картин Моне, а другое выполняется в обратном направлении. Для этого CycleGAN использует два условных генератора, \mathbf{g}_x и \mathbf{g}_y , и два дискриминатора, d_x и d_y . Генератор $\mathbf{g}_x(\mathbf{y}, \mathbf{w}_x)$ принимает на вход образец картины $\mathbf{y} \in Y$ и генерирует соответствующую синтетическую фотографию, а дискриминатор $d_x(\mathbf{x}, \boldsymbol{\phi}_x)$ различает синтетические и реальные фотографии. Точно так же генератор $\mathbf{g}_y(\mathbf{x}, \mathbf{w}_y)$ принимает на вход фотографию $\mathbf{x} \in X$ и генерирует синтетическую картину \mathbf{y} , а дискриминатор $d_y(\mathbf{y}, \boldsymbol{\phi}_y)$ отличает синтетические картины от реальных. Получается, что дискриминатор d_x обучается на комбинации синтетических фотографий, сгенерированных \mathbf{g}_x , и реальных фотографий, в то время как d_y обучается на комбинации синтетических картин, сгенерированных \mathbf{g}_y , и реальных картин.

Если обучать эту архитектуру с помощью стандартной функции потерь GAN, она научится генерировать реалистичные синтетические картины Моне и реалистичные синтетические фотографии, но ничто не заставит сгени-

нерированную картину быть похожей на соответствующую фотографию или наоборот. Поэтому в функцию потерь вводится дополнительный член, называемый *ошибкой согласованности циклов* (*cycle consistency error*), куда входят два члена, структура которых показана на рис. 17.7. Здесь сначала фотография отображается в область живописи с помощью генератора \mathbf{g}_y , а затем полученный вектор отображается обратно в область фотографии с помощью генератора \mathbf{g}_x . Расхождение между полученной фотографией и оригиналом \mathbf{x}_n определяет вклад в ошибку согласованности цикла. Аналогичный процесс используется для вычисления вклада в ошибку согласованности цикла от картины \mathbf{y}_n путем отображения ее на фотографию с помощью \mathbf{g}_x , а затем обратно на картину с помощью \mathbf{g}_y .

РИС. 17.7 Диаграмма вычисления ошибки согласованности циклов для примера фотографии \mathbf{x}_n



Цель состоит в том, чтобы при переводе фотографии в картину, а затем обратно в фотографию картина была близка к оригинальной фотографии, тем самым гарантируя, что сгенерированная картина сохраняет достаточно информации о фотографии, чтобы ее можно было реконструировать. Точно так же, когда картина переводится в фотографию, а затем снова в картину, она должна быть близка к оригиналам. Применение этого принципа ко всем фотографиям и картинам в обучающем наборе дает ошибку согласованности цикла вида

$$\begin{aligned} E_{\text{cyc}}(\mathbf{w}_X, \mathbf{w}_Y) = & \frac{1}{N_X} \sum_{n \in X} \|\mathbf{g}_X(\mathbf{g}_Y(\mathbf{x}_n)) - \mathbf{x}_n\|_1 \\ & + \frac{1}{N_Y} \sum_{n \in Y} \|\mathbf{g}_Y(\mathbf{g}_X(\mathbf{y}_n)) - \mathbf{y}_n\|_1, \end{aligned} \quad (17.12)$$

где $\|\cdot\|_1$ обозначает норму L1. Ошибка согласованности циклов добавляется к обычным функциям потерь GAN, определенным в (17.6), чтобы получить функцию общей ошибки

$$E_{\text{GAN}}(\mathbf{w}_X, \boldsymbol{\varphi}_X) + E_{\text{GAN}}(\mathbf{w}_Y, \boldsymbol{\varphi}_Y) + \eta E_{\text{cyc}}(\mathbf{w}_X, \mathbf{w}_Y), \quad (17.13)$$

где коэффициент η определяет относительную важность ошибок GAN и ошибки согласованности цикла. Поток информации через CycleGAN при вычислении функции ошибки для одного изображения и одной картины показан на рис. 17.8.

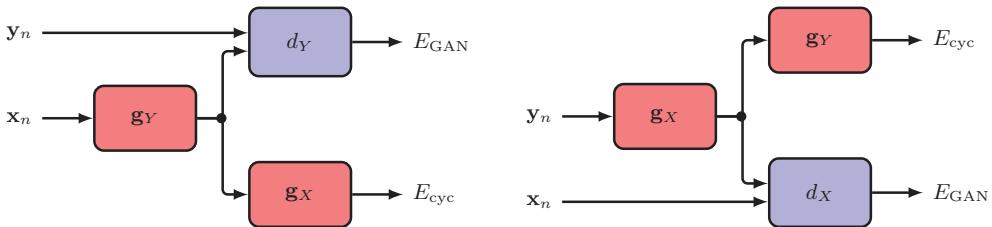


РИС. 17.8 Поток информации через CycleGAN. Общая ошибка для точек данных \mathbf{x}_n и \mathbf{y}_n является суммой четырех составляющих ошибок

Ранее уже говорилось о том, что GAN могут хорошо работать в качестве генеративных моделей, но их также можно использовать для обучения представлениям (*representation learning*) (см. раздел 6.3.3), когда богатая статистическая структура в наборе данных выявляется с помощью обучения без контроля. Когда глубокая сверточная GAN, показанная на рис. 17.4, обучается на наборе данных с изображениями спален (Radford, Metz and Chintala, 2015) и случайные выборки из латентного пространства распространяются по обученной сети, сгенерированные изображения также выглядят как спальни, как и ожидалось. Кроме того, латентное пространство становится организованным в семантически значимом смысле. Например, если следовать плавной траектории через латентное пространство и сгенерировать соответствующую серию изображений, получится плавные переходы от одного изображения к другому, как показано на рис. 17.9. Каждая строка сгенерирована путем плавного перемещения по латентному пространству между случайно сгенерированными местами. Видны плавные переходы, при этом каждое изображение правдоподобно выглядит как спальня. Например, в нижнем ряду видно, как телевизор на стене постепенно превращается в окно.

Более того, в латентном пространстве можно определить направления, которые соответствуют семантически значимым преобразованиям. Например, для лиц одно направление может соответствовать изменениям в их ориентации, в то время как другие направления могут соответствовать изменениям в освещении или тому, насколько это лицо улыбается. Это так называемые *разделенные представления* (*disentangled representations*), которые позволяют синтезировать новые изображения с заданными свойствами. На рис. 17.10 приведен пример GAN, обученной на изображениях лиц, где показано, что семантические атрибуты, такие как пол или наличие очков, соответствуют определенным направлениям в латентном пространстве. В каждом из трех столбцов векторы латентного пространства, породившие эти изображения, усредняются, а затем к полученным средним векторам применяется векторная арифметика, чтобы создать новый вектор, соответствующий центральному изображению в массиве 3×3 справа. Добавление шума к этому вектору создает еще восемь образцов изображений. Четыре изображения в нижнем ряду показывают, что та же арифметика, примененная непосредственно в пространстве данных, просто приводит к размытию изображения из-за несогласованности.

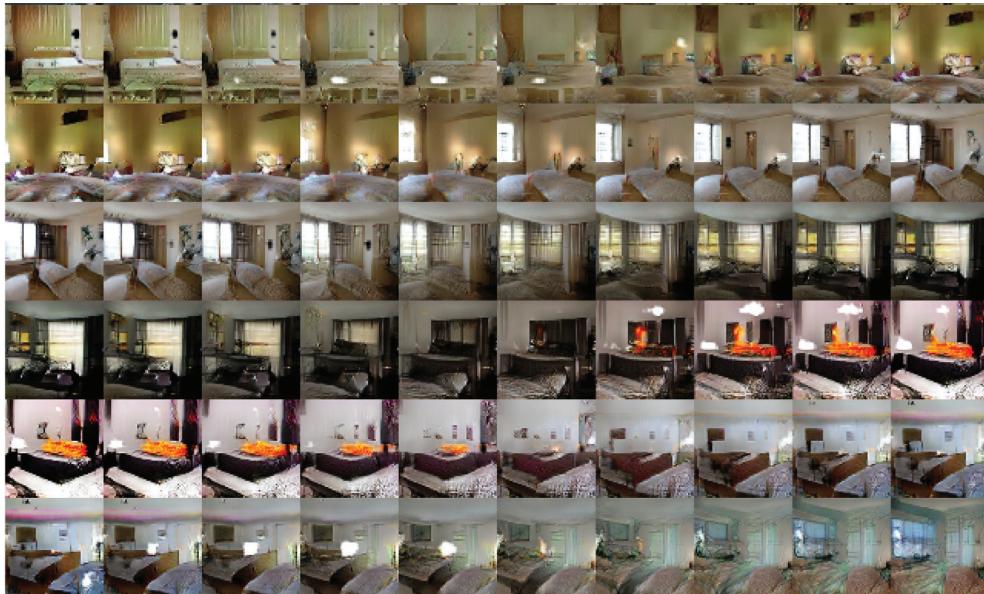


РИС. 17.9 Образцы, сгенерированные глубокой сверточной GAN, обученной на изображениях спален. [Из (Radford, Metz and Chintala, 2015) с разрешениями авторов]



РИС. 17.10 Пример векторной арифметики в латентном пространстве обученной GAN. [Из (Radford, Metz and Chintala, 2015) с разрешениями авторов]

Упражнения

- 17.1** (***) Хотелось бы, чтобы функция ошибки GAN (17.6) обладала тем свойством, что при достаточно гибких нейронных сетях стационарная точка получается при совпадении распределения генератора с истин-

ным распределением данных. В этом упражнении необходимо доказать этот результат для сетевых моделей с бесконечной гибкостью, оптимизируя по всему пространству вероятностных распределений $p_G(\mathbf{x})$ и по всему пространству функций $d(\mathbf{x})$ генеративной и дискриминантной сетей соответственно. В частности, предположим, что дискриминантная модель оптимизируется во внутреннем цикле, что приводит к эффективной функции ошибки внешнего цикла для генеративной модели. Сначала нужно доказать, что в пределе бесконечного числа выборок данных функция ошибки GAN (17.6) может быть переписана в виде

$$E(p_G, d) = - \int p_{\text{data}}(\mathbf{x}) \ln d(\mathbf{x}) d\mathbf{x} - \int p_G(\mathbf{x}) \ln(1 - d(\mathbf{x})) d\mathbf{x}, \quad (17.14)$$

где $p_{\text{data}}(\mathbf{x})$ – это фиксированное распределение реальных точек данных. Теперь рассмотрим вариационную оптимизацию (см. приложение B) по всем функциям $d(\mathbf{x})$. Докажите, что для фиксированной генеративной сети решение для дискриминатора $d(\mathbf{x})$, минимизирующее E ,дается в виде

$$d^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}. \quad (17.15)$$

Отсюда следует, что функция ошибки E может быть записана как функция генераторной сети $p_G(\mathbf{x})$ в виде

$$\begin{aligned} C(p_G) &= - \int p_{\text{data}}(\mathbf{x}) \ln \left\{ \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \right\} d\mathbf{x} \\ &\quad - \int p_G(\mathbf{x}) \ln \left\{ \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \right\} d\mathbf{x}. \end{aligned} \quad (17.16)$$

Теперь докажите, что это можно переписать в виде

$$C(p_G) = -\ln(4) + \text{KL}\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_G}{2}\right) + \text{KL}\left(p_G \middle\| \frac{p_{\text{data}} + p_G}{2}\right), \quad (17.17)$$

где расходимость Кульбака–Лейблера $\text{KL}(p \parallel q)$ определяется в (2.100). Наконец, используя свойство $\text{KL}(p \parallel q) \geq 0$ при равенстве тогда и только тогда, когда $p(\mathbf{x}) = q(\mathbf{x})$ для всех \mathbf{x} (см. раздел 2.5.5), докажите, что минимум $C(p_G)$ имеет место при $p_G(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$. Обратите внимание, что сумма двух членов расхождения Кульбака–Лейблера в (17.17) между p_{data} и p_G известна как *расхождение Дженсена–Шеннона (Jensen–Shannon divergence)*. Как и расхождение Кульбака–Лейблера, это неотрицательная величина, которая исчезает тогда и только тогда, когда два распределения равны, но, в отличие от расхождения KL, она симметрична относительно двух распределений.

- 17.2** (★★) В этом упражнении рассматриваются проблемы, которые могут возникнуть из-за состязательного характера обучения GAN. Рассмотрим функцию стоимости $E(a, b) = ab$, заданную двумя параметрами a и b , аналогичными параметрам генеративной и дискриминантной сети соответственно. Докажите, что точка $a = 0, b = 0$ является стационарной точкой функции стоимости. Рассматривая вторые производные по линиям $b = a$ и $b = -a$, докажите, что точка $a = 0, b = 0$ является седловой точкой. Теперь предположим, что эту функцию ошибки нужно оптимизировать, делая бесконечно малые шаги, так что переменные становятся функциями непрерывного времени $a(t), b(t)$, определяемыми непрерывным градиентным спуском, где параметр $a(t)$ генеративной сети обновляется с целью увеличения $E(a, b)$, тогда как параметр $b(t)$ обновляется с целью уменьшения $E(a, b)$. Докажите, что эволюция параметров подчиняется уравнениям:

$$\frac{da}{dt} = \eta \frac{\partial E}{\partial a}, \quad \frac{db}{dt} = -\eta \frac{\partial E}{\partial b}. \quad (17.18)$$

Отсюда следует, что $a(t)$ удовлетворяет дифференциальному уравнению второго порядка:

$$\frac{d^2a}{dt^2} = -\eta^2 a(t). \quad (17.19)$$

Докажите, что следующее выражение является решением (17.19):

$$a(t) = C \cos(\eta t) + D \sin(\eta t), \quad (17.20)$$

где C и D – это произвольные постоянные. Если система инициализирована при $t = 0$ значениями $a = 1, b = 0$, найдите значения C и D и докажите, что полученные значения $a(t)$ и $b(t)$ очерчивают окружность единичного радиуса в пространстве a, b с центром в начале координат, и по этой причине они никогда не сходятся к седловой точке.

- 17.3** (*) Рассмотрим GAN, в которой обучающий набор состоит из равного количества изображений кошек и собак, а сеть генератора научилась создавать высококачественные изображения собак. Докажите, что при предъявлении изображения собаки оптимальный выход сети дискриминатора (обученной генерировать вероятность того, что изображение является реальным) равен $1/3$.

Глава 18

Нормализующие потоки

Ранее уже было показано, как генеративные состязательные сети (GAN) расширяют рамки линейных моделей латентных переменных, используя глубокие нейронные сети для представления очень гибких и обучаемых нелинейных преобразований из латентного пространства в пространство данных (см. главу 17). Однако функция правдоподобия обычно либо неразрешима, поскольку функция сети не может быть инвертирована, либо вообще не может быть определена, если латентное пространство имеет меньшую размерность, чем пространство данных. Поэтому при разработке GAN была введена вторая – дискриминативная – сеть, которая облегчает процесс обучения по принципу состязательности.

В этой главе рассматривается второй из четырех методов (см. раздел 16.4.4) обучения нелинейных моделей латентных переменных, который предполагает ограничение формы нейросетевой модели таким образом, чтобы функцию правдоподобия можно было оценить без применения аппроксимации, но при этом обеспечить простоту выборки из обученной модели. Предположим, что задано распределение $p_z(z)$, иногда называемое также *базовым распределением* (*base distribution*), для латентной переменной z , а также нелинейная функция $\mathbf{x} = \mathbf{f}(\mathbf{z}, \mathbf{w})$, задаваемая глубокой нейронной сетью, которая преобразует латентное пространство в пространство данных. Если предположить, что $p_z(z)$ – это простое распределение, например гауссово, то выборка из такой модели не представляет сложности, поскольку каждая латентная выборка $\mathbf{z}^* \sim p_z(z)$ просто пропускается через нейронную сеть для генерации соответствующей выборки данных $\mathbf{x}^* = \mathbf{f}(\mathbf{z}^*, \mathbf{w})$.

Для вычисления функции правдоподобия для этой модели необходимо распределение в пространстве данных, которое зависит от обратной функции нейронной сети. Это распределение записывается как $\mathbf{z} = \mathbf{g}(\mathbf{x}, \mathbf{w})$, и оно удовлетворяет $\mathbf{z} = \mathbf{g}(\mathbf{f}(\mathbf{z}, \mathbf{w}), \mathbf{w})$. Это требует, чтобы для каждого значения \mathbf{w} функции $\mathbf{f}(\mathbf{z}, \mathbf{w})$ и $\mathbf{g}(\mathbf{x}, \mathbf{w})$ были инвертируемыми, т. е. *биективными* (*bijection*), так что каждому значению \mathbf{x} соответствует единственное значение \mathbf{z} , и наоборот. Затем можно использовать формулу замены переменных (см. раздел 2.4) для вычисления плотности данных:

$$p_x(\mathbf{x} | \mathbf{w}) = p_z(\mathbf{g}(\mathbf{x}, \mathbf{w})) |\det \mathbf{J}(\mathbf{x})|, \quad (18.1)$$

где $\mathbf{J}(\mathbf{x})$ – это матрица Якоби частных производных, элементы которой даны в виде

$$J_{ij}(\mathbf{x}) = \frac{\partial g_i(\mathbf{x}, \mathbf{w})}{\partial x_j}, \quad (18.2)$$

а $|\cdot|$ обозначает модуль или абсолютную величину. В дальнейшем также будем называть \mathbf{z} «латентной» переменной, хотя детерминированное отображение означает, что любому заданному значению данных \mathbf{x} соответствует единственное значение \mathbf{z} , которое, следовательно, больше не является непредetermined.

Функция отображения $\mathbf{f}(\mathbf{z}, \mathbf{w})$ будет определена в виде специальной формы нейронной сети, структуру которой рассмотрим в ближайшее время. Одним из следствий требования инвертируемого отображения является то, что размерность латентного пространства должна быть такой же, как и размерность пространства данных, что может привести к созданию больших моделей для данных высокой размерности, таких как изображения. Кроме того, в общем случае стоимость оценки детерминанта матрицы $D \times D$ составляет $\mathcal{O}(D^3)$, поэтому необходимо наложить некоторые дополнительные ограничения на модель, чтобы оценка определителя матрицы Якоби была более эффективной.

Если взять обучающий набор $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ независимых точек данных, то функция логарифмического правдоподобия задается из (18.1) в виде

$$\ln p(\mathcal{D} | \mathbf{w}) = \sum_{n=1}^N \ln p_x(\mathbf{x}_n | \mathbf{w}) \quad (18.3)$$

$$= \sum_{n=1}^N \{\ln p_z(\mathbf{g}(\mathbf{x}_n | \mathbf{w})) + \ln |\det \mathbf{J}(\mathbf{x}_n)|\}. \quad (18.4)$$

Цель – обучение нейронной сети с помощью функции правдоподобия. Чтобы иметь возможность моделировать широкий спектр распределений, необходимо, чтобы функция преобразования $\mathbf{x} = \mathbf{f}(\mathbf{z}, \mathbf{w})$ была очень гибкой, и поэтому в данном случае применяется архитектура глубокой нейронной сети. Обеспечить инверсию общей функции можно с помощью инвертирования каждого слоя сети (см. упражнение 18.2). Для этого рассмотрим три последовательных преобразования, каждое из которых соответствует одному слою, в виде

$$\mathbf{x} = \mathbf{f}^A(\mathbf{f}^B(\mathbf{f}^C(\mathbf{z}))). \quad (18.5)$$

Тогда обратная функция будет иметь вид

$$\mathbf{z} = \mathbf{g}^C(\mathbf{g}^B(\mathbf{g}^A(\mathbf{x}))), \quad (18.6)$$

где $\mathbf{g}^A, \mathbf{g}^B$ и \mathbf{g}^C – это обратные функции $\mathbf{f}^A, \mathbf{f}^B$ и \mathbf{f}^C соответственно. Более того, определитель Якоби для такой послойной структуры также легко оценивается в виде определителей Якоби для каждого из отдельных слоев путем использования для этого правила цепных вычислений:

$$J_{ij} = \frac{\partial z_i}{\partial x_j} = \sum_k \sum_l \frac{\partial g_i^C}{\partial g_k^B} \frac{\partial g_k^B}{\partial g_l^A} \frac{\partial g_l^A}{\partial x_j}. \quad (18.7)$$

Правая часть представляет собой произведение трех матриц, а определитель произведения – это произведение определителей (см. приложение А). По этой причине логарифмический определитель общего якобиана будет равен сумме логарифмических определителей, соответствующих каждому слою.

Такой подход к моделированию гибкого распределения называется *нормализующим потоком* (*normalizing flow*), поскольку преобразование распределения вероятностей через последовательность отображений в некотором роде аналогично течению жидкости. Кроме того, эффект обратного отображения заключается в преобразовании сложного распределения данных в нормализованную форму, как правило, гауссово или нормальное распределение. Нормализующие потоки были рассмотрены в работах (Kobyzev, Prince and Brubaker, 2019) и (Papamakarios et al., 2019). Здесь будут рассмотрены основные концепции двух основных классов нормализующих потоков, используемых на практике: *потоков сопряжения* (*coupling flows*) и *потоков авторегрессии* (*autoregressive flows*). Кроме того, будет рассмотрено использование нейронных дифференциальных уравнений для определения инвертируемых отображений, которые ведут к концепции *непрерывных потоков* (*continuous flows*).

18.1. Потоки сопряжения

Целью разработки является создание одного слоя инвертируемых функций, чтобы потом можно было скомпоновать множество таких слоев и определить крайне гибкий класс инвертируемых функций. Рассмотрим для начала линейное преобразование вида

$$\mathbf{x} = a\mathbf{z} + \mathbf{b}. \quad (18.8)$$

Его удобно инвертировать, что дает

$$\mathbf{z} = \frac{1}{a}(\mathbf{x} - \mathbf{b}). \quad (18.9)$$

Однако линейные преобразования замкнуты в композиции, т. е. последовательность линейных преобразований эквивалентна одному общему линейному преобразованию (см. упражнение 3.6). Более того, линейное преобразование гауссова распределения вновь будет гауссовым. Поэтому, даже если будет много таких «слоев» линейных преобразований, в итоге все равно получится только гауссово распределение. Вопрос в том, можно ли сохранить инвертируемость линейного преобразования и при этом обеспечить дополнительную гибкость, чтобы результирующее распределение могло быть негауссовым.

Одно из решений этой проблемы дает форма модели нормализации потока, называемая *реальной NVP* (*real NVP*) (Dinh, Krueger and Bengio, 2014; Dinh, Sohl-Dickstein and Bengio, 2016), что является сокращением от «вещественная, не сохраняющая объем» (*real-valued non-volume-preserving*). Идея состоит в разбивке вектора латентных переменных \mathbf{z} на две части $\mathbf{z} = (\mathbf{z}_A, \mathbf{z}_B)$ таким образом, что если \mathbf{z} имеет размерность D и \mathbf{z}_A имеет размерность d , тогда \mathbf{z}_B имеет размерность $D - d$. Точно так же можно разбить выходной вектор $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_B)$, где \mathbf{x}_A имеет размерность d , а \mathbf{x}_B имеет размерность $D - d$. Для первой части выходного вектора достаточно просто скопировать входной:

$$\mathbf{x}_A = \mathbf{z}_A. \quad (18.10)$$

Вторая часть вектора подвергается линейному преобразованию, но теперь коэффициенты в линейном преобразовании задаются нелинейными функциями от \mathbf{z}_A :

$$\mathbf{x}_B = \exp(\mathbf{s}(\mathbf{z}_A, \mathbf{w})) \odot \mathbf{z}_B + \mathbf{b}(\mathbf{z}_A, \mathbf{w}), \quad (18.11)$$

где $\mathbf{s}(\mathbf{z}_A, \mathbf{w})$ и $\mathbf{b}(\mathbf{z}_A, \mathbf{w})$ – это вещественные выходы нейронных сетей, а экспонента гарантирует, что мультипликативный член неотрицателен. Здесь \odot означает *произведение Адамара* (*Hadamard product*), подразумевающее поэлементное умножение двух векторов. Аналогичным образом экспонента в (18.11) берется поэлементно. Обратите внимание, что в обеих сетевых функциях показан один и тот же вектор \mathbf{w} . На практике они могут быть реализованы как отдельные сети со своими параметрами или как одна сеть с двумя наборами выходов.

Благодаря использованию нейросетевых функций значение \mathbf{x}_B может быть очень гибкой функцией от \mathbf{x}_A . Тем не менее общее преобразование легко инвертируется. Задав значение $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_B)$, сначала вычисляем

$$\mathbf{z}_A = \mathbf{x}_A, \quad (18.12)$$

затем оцениваем $\mathbf{s}(\mathbf{z}_A, \mathbf{w})$ и $\mathbf{b}(\mathbf{z}_A, \mathbf{w})$ и, наконец, вычисляем \mathbf{z}_B , используя

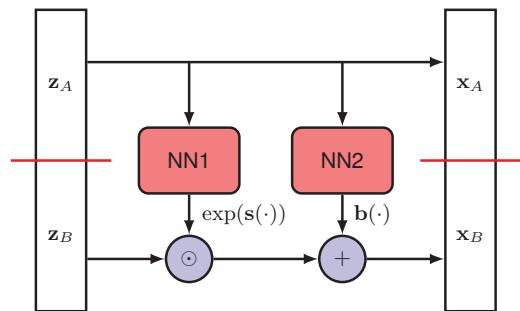
$$\mathbf{z}_B = \exp(-\mathbf{s}(\mathbf{z}_A, \mathbf{w})) \odot (\mathbf{x}_B - \mathbf{b}(\mathbf{z}_A, \mathbf{w})). \quad (18.13)$$

Общее преобразование показано на рис. 18.1. Здесь сеть NN1 вычисляет функцию $\exp(\mathbf{s}(\mathbf{z}_A, \mathbf{w}))$, а сеть NN2 вычисляет функцию $\mathbf{b}(\mathbf{z}_A, \mathbf{w})$. Выходной вектор определяется в (18.10) и (18.11). Обратите внимание, что для отдельных функций нейронной сети $\mathbf{s}(\mathbf{z}_A, \mathbf{w})$ и $\mathbf{b}(\mathbf{z}_A, \mathbf{w})$ нет требования быть инвертируемыми.

Теперь рассмотрим оценку матрицы Якоби, определяемой в (18.2), и ее определителя. Можно разбить матрицу Якоби на блоки, соответствующие разбиению \mathbf{z} и \mathbf{x} , что дает

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_d & \mathbf{0} \\ \frac{\partial \mathbf{z}_B}{\partial \mathbf{x}_A} & \text{diag}(\exp(-\mathbf{s})) \end{bmatrix}. \quad (18.14)$$

РИС. 18.1 Однослойная модель нормализующего потока реальной NVP



Левый верхний блок соответствует производным z_A по x_A и, следовательно, из (18.12) задается матрицей тождества $d \times d$. Правый верхний блок соответствует производным z_A по x_B , и эти члены исчезают, опять же исходя из (18.12). Левый нижний блок соответствует производным z_B по x_A . Из (18.13) следует, что это сложные выражения, включающие функции нейронной сети. Наконец, правый нижний блок соответствует производным z_B по x_B , которые из (18.13) задаются диагональной матрицей, а ее диагональные элементы представляют собой экспоненты отрицательных элементов $s(z_A, w)$. Таким образом, получается, что матрица Якоби (18.14) является нижней треугольной матрицей, т. е. все ее элементы над главной диагональю равны нулю. Для такой матрицы определитель – это просто произведение элементов вдоль ведущей диагонали (см. приложение А), и поэтому он не зависит от сложных выражений в левом нижнем блоке. Следовательно, определитель матрицы Якоби задается простым произведением элементов $\exp(-s(z_A, w))$.

Явное ограничение этого подхода заключается в том, что при преобразовании значение z_A остается неизменным. Это легко решается добавлением еще одного слоя, в котором роли z_A и z_B меняются местами, как показано на рис. 18.2. Эта двухслойная структура может быть использована многократно для построения очень гибкого класса генеративных моделей.

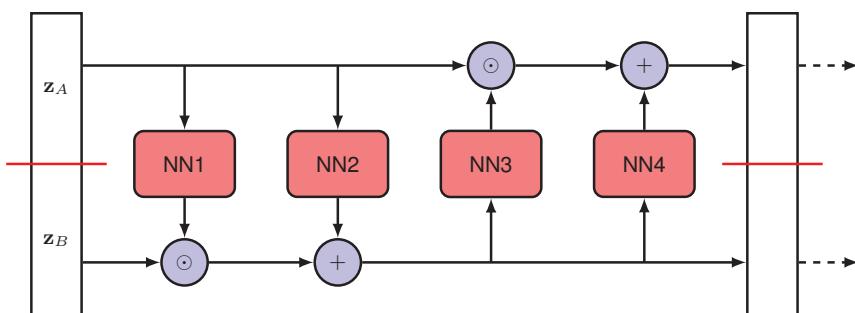


РИС. 18.2 Путем составления двух слоев (см. рис. 18.1) можно получить более гибкий, но все еще инвертируемый нелинейный слой. Каждый дополнительный слой инвертируем и имеет легко вычисляемую матрицу Якоби, поэтому общий двойной слой обладает теми же свойствами

Общая процедура обучения подразумевает создание мини-батчей точек данных, в которых вклад каждой точки данных в функцию логарифмического правдоподобия получается из (18.4). Для латентного распределения вида $\mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I})$ логарифмическая плотность равна просто $-\|\mathbf{z}\|^2/2$ до аддитивной константы. Обратное преобразование $\mathbf{z} = \mathbf{g}(\mathbf{x})$ вычисляется с помощью последовательности обратных преобразований вида (18.13). Точно так же логарифм определителя матрицы Якоби задается суммой логарифмических определителей для каждого слоя, где каждый член сам является суммой членов вида $-s_i(\mathbf{x}, \mathbf{w})$. Градиенты логарифмического правдоподобия могут быть оценены с помощью автоматического дифференцирования, а параметры сети обновляются с помощью стохастического градиентного спуска.

Модель Real NVP относится к широкому классу нормализующих потоков, называемых *потоками сопряжения* (*coupling flows*), в которых линейное преобразование (18.11) заменяется более общей формой

$$\mathbf{x}_B = \mathbf{h}(\mathbf{z}_B, \mathbf{g}(\mathbf{z}_A, \mathbf{w})), \quad (18.15)$$

где $\mathbf{h}(\mathbf{z}_B, \mathbf{g})$ – это функция \mathbf{z}_B , эффективно инвертируемая при любом заданном значении \mathbf{g} и называемая *функцией связи* (*coupling function*). Функция $\mathbf{g}(\mathbf{z}_A, \mathbf{w})$, которая обычно представлена нейронной сетью, называется *формирователем сигнала* (*conditioner*).

Поток нормализации с помощью реальной NVP можно проиллюстрировать на примере простого набора данных, иногда известного как «две луны» (*«two-moons»*), как показано на рис. 18.3. В этом примере двумерное гауссово распределение преобразуется в более сложное распределение с помощью двух последовательных слоев, каждый из которых состоит из чередующихся преобразований по каждому из двух измерений. Здесь: (a) базовое гауссово

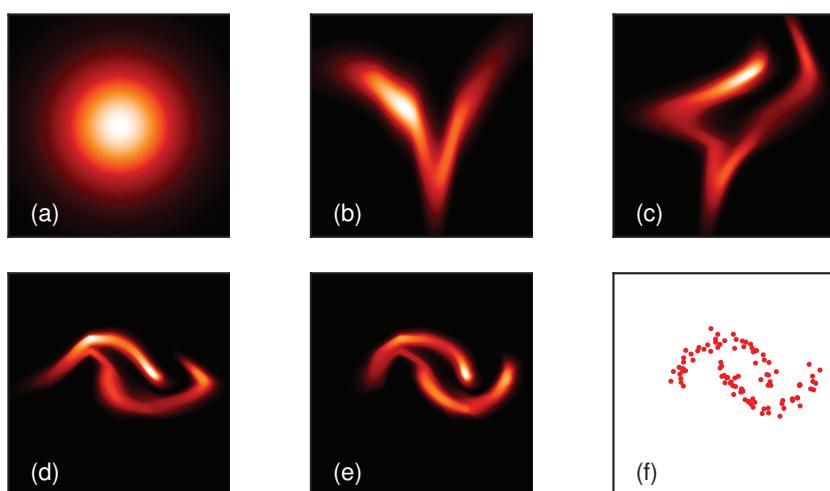


РИС. 18.3 Иллюстрация модели нормализующего потока реальной NVP, примененной к набору данных «две луны»

распределение, (b) распределение после преобразования только вертикальной оси, (c) распределение после последующего преобразования горизонтальной оси, (d) распределение после второго преобразования вертикальной оси, (e) распределение после второго преобразования горизонтальной оси и (f) набор данных, на котором обучалась модель.

18.2. Потоки авторегрессии

Подходящая формулировка для нормализующих потоков может быть основана на том, что совместное распределение по набору переменных всегда может быть записано как произведение условных распределений (см. раздел 11.1), по одному для каждой переменной. Сначала выберем упорядочение переменных в векторе \mathbf{x} , на основе которого можно записать без потери общности

$$p(x_1, \dots, x_D) = \prod_{i=1}^D p(x_i | \mathbf{x}_{1:i-1}), \quad (18.16)$$

где $\mathbf{x}_{1:i-1}$ обозначает x_1, \dots, x_{i-1} . Эта факторизация может быть использована для построения класса нормализующих потоков, называемых *маскированными авторегрессионными потоками* (*masked autoregressive flow, MAF*) (Papamakarios, Pavlakou and Murray, 2017), которые определяются как

$$x_i = h(z_i, \mathbf{g}_i(\mathbf{x}_{1:i-1}, \mathbf{w}_i)), \quad (18.17)$$

что проиллюстрировано на рис. 18.4(a). Здесь $h(z_i, \cdot)$ – это функция связи, которая выбирается таким образом, чтобы быть легко инвертируемой по отношению к z_i , а \mathbf{g}_i – это формирователь сигнала, который обычно представлен глубокой нейронной сетью. Термин «маскированный» подразумевает использование одной нейронной сети для реализации набора уравнений вида (18.17) вместе с бинарной маской (Germain et al., 2015), которая обнуляет подмножество весов сети для реализации ограничения авторегрессии (18.16).

В этом случае обратные вычисления, необходимые для оценки функции правдоподобия, выглядят как

$$z_i = h^{-1}(x_i, \mathbf{g}_i(\mathbf{x}_{1:i-1}, \mathbf{w}_i)) \quad (18.18)$$

и, следовательно, могут быть эффективно выполнены на современном оборудовании, поскольку отдельные функции в (18.18), необходимые для оценки z_1, \dots, z_D , могут быть вычислены параллельно. Матрица Якоби, соответствующая набору преобразований в (18.18), имеет элементы $\partial z_i / \partial x_j$ (см. упражнение 18.4). Они образуют верхнюю треугольную матрицу, детерминант которой задается произведением диагональных элементов, и поэтому она также может быть вычислена достаточно эффективно. Однако выборка из этой модели должна производиться путем вычисления (18.17), что по своей

сущности является последовательным процессом и, следовательно, медленным, поскольку значения x_1, \dots, x_{i-1} должны быть вычислены до определения x_i .

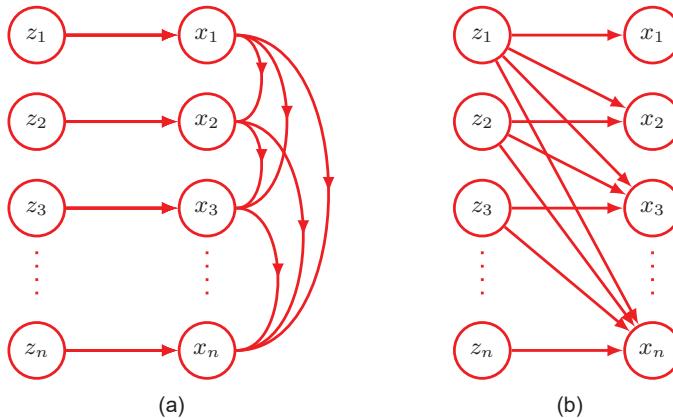


РИС. 18.4 Иллюстрация двух альтернативных структур для авторегрессионных нормализующих потоков. Маскированный поток авторегрессии, показанный на (a), позволяет эффективно оценивать функцию правдоподобия, в то время как альтернативный обратный поток авторегрессии, показанный на (b), обеспечивает эффективную выборку

Чтобы избежать этой неэффективной выборки, можно взамен определить *обратные потоки авторегрессии* (*inverse autoregressive flows*, IAF) (Kingma et al., 2016), которые представлены в виде

$$x_i = h(z_i, \tilde{\mathbf{g}}_i(\mathbf{z}_{1:i-1}, \mathbf{w}_i)), \quad (18.19)$$

как показано на рис. 18.4б. Теперь выборка эффективна, так как для заданного выбора \mathbf{z} оценку элементов x_1, \dots, x_D с помощью (18.19) можно выполнять в параллельном режиме. Однако обратная функция, необходимая для оценки правдоподобия, требует серии вычислений вида

$$z_i = h^{-1}(x_i, \tilde{\mathbf{g}}_i(\mathbf{z}_{1:i-1}, \mathbf{w}_i)), \quad (18.20)$$

которые по своей сути являются последовательными и, следовательно, медленными. Выбор масочного авторегрессионного потока или обратного авторегрессионного потока определяется спецификой конкретной задачи.

Теперь становится ясно, что потоки сопряжения и потоки авторегрессии имеют тесную связь. Хотя потоки авторегрессии обеспечивают значительную гибкость, это сопровождается вычислительными затратами, которые возрастают линейно с увеличением размерности D пространства данных из-за необходимости последовательной выборки предков. Потоки сопряжения можно рассматривать как частный случай авторегрессионных потоков, где часть этой общности приносится в жертву эффективности за счет разделения переменных на две группы вместо D групп.

18.3. Непрерывные потоки

Последний подход к нормализации потоков, который рассматривается в этой главе, предполагает использование глубоких нейронных сетей, определенных в выражениях обыкновенного дифференциального уравнения (ordinary differential equation), или ОДУ (ODE). Их можно представить в виде глубокой сети с бесконечным числом слоев. Сначала введем концепцию нейронного ОДУ, а затем рассмотрим возможности его применения к разработке модели нормализующего потока.

18.3.1. Нейронные дифференциальные уравнения

Выше уже говорилось об эффективности нейронных сетей со многими слоями обработки, и поэтому можно задаться вопросом о том, что происходит в случае использования бесконечно большого числа слоев. Рассмотрим остаточную сеть, в которой каждый слой обработки генерирует выход, заданный входным вектором с добавлением некоторой параметризованной нелинейной функции этого входного вектора, в виде

$$\mathbf{z}^{(t+1)} = \mathbf{z}^{(t)} + \mathbf{f}(\mathbf{z}^{(t)}, \mathbf{w}), \quad (18.21)$$

где $t = 1, \dots, T$ обозначают слои сети. Обратите внимание, что для каждого слоя используется одна и та же функция с общим вектором параметров \mathbf{w} , поскольку это позволяет рассматривать произвольно большое число таких слоев при сохранении ограниченного числа параметров. Предположим, что количество слоев увеличивается, но при этом изменения на каждом слое пропорционально уменьшаются. В пределе вектор активации скрытого блока становится функцией $\mathbf{z}^{(t)}$ от непрерывной переменной t , и эволюцию этого вектора в сети можно выразить в виде дифференциального уравнения (см. упражнение 18.5) вида

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{f}(\mathbf{z}(t), \mathbf{w}), \quad (18.22)$$

где t обычно означает «время». Формула (18.22) известна как *нейронное обыкновенное дифференциальное уравнение* (*neural ordinary differential equation*), или *нейронное ОДУ* (*neural ODE*) (Chen et al., 2018). Здесь «обыкновенное» подразумевает наличие единственной переменной t . Если обозначить вход сети вектором $\mathbf{z}(0)$, то выход $\mathbf{z}(T)$ получается путем интегрирования дифференциального уравнения:

$$\mathbf{z}(T) = \int_0^T \mathbf{f}(\mathbf{z}(t), \mathbf{w}) dt. \quad (18.23)$$

Этот интеграл можно вычислить с помощью стандартных пакетов численного интегрирования. Простейшим методом решения дифференциальных уравнений является метод *прямого интегрирования Эйлера* (*Euler's forward in-*

tegration), который соответствует выражению (18.21). На практике для оценки функции могут быть адаптированы более мощные алгоритмы численного интегрирования. В частности, они могут гибко выбирать значения t , которые, как правило, не являются равномерно распределенными. Количество таких оценок замещает концепцию глубины в обычной многослойной сети. Сравнение стандартной многослойной нейронной сети и нейронного дифференциального уравнения показано на рис. 18.5. На диаграмме справа показан результат численного интегрирования непрерывного нейронного ОДУ, опять же для нескольких начальных значений скалярного входа, где видно, что функция не оценивается на равномерно распределенных временных интервалах, а точки оценки выбираются аддитивно численным решением и зависят от выбора входного значения.

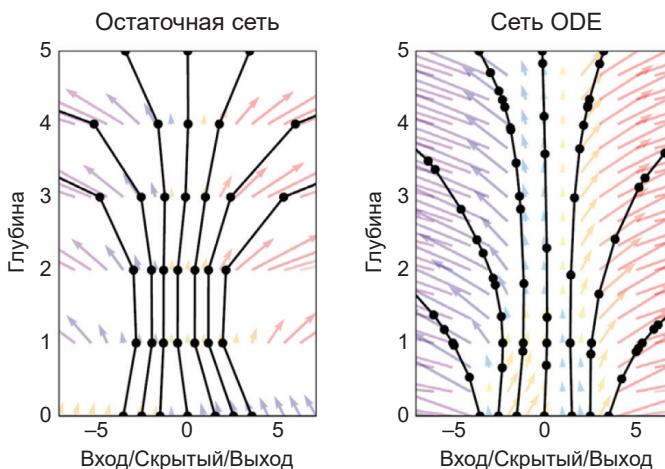


РИС. 18.5 Сравнение обычной многослойной сети и нейронного дифференциального уравнения. Диаграмма слева соответствует остаточной сети с пятью слоями и показывает траектории для нескольких начальных значений одного скалярного входа. [Из (Chen et al., 2018) с разрешения авторов]

18.3.2. Обратное распространение нейронных ОДУ

Теперь необходимо решить задачу обучения нейронного ОДУ, т. е. определить значение w путем оптимизации функции потерь. Предположим, что имеется набор данных из значений входного вектора $\mathbf{z}(0)$ и связанного с ним выходного целевого вектора, а также функция потерь $L(\cdot)$, зависящая от выходного вектора $\mathbf{z}(T)$. Один из подходов заключается в использовании автоматического дифференцирования (см. раздел 8.2) для дифференцирования по всем операциям, выполняемым решателем ОДУ во время прямого прохода. Хотя это достаточно несложно, но такой подход требует больших затрат памяти и не является оптимальным с точки зрения контроля численной погрешности. Вместо этого в работе (Chen et al., 2018) решатель ОДУ рассматривается как «черный ящик» и используется методика под названием *метод сопряженной*

чувствительности (adjoint sensitivity method), которую можно рассматривать как непрерывный аналог явного обратного распространения. Напомним, что обратное распространение включает в себя (см. главу 8) три последовательных этапа для каждой точки данных: первый – прямое распространение для оценки векторов активации на каждом слое сети; второй – оценка производных потерь относительно активаций на каждом слое, начиная с выхода и распространяясь назад по сети с использованием правила цепного исчисления; и третий – оценка производных относительно параметров сети путем формирования произведений активаций из прямого прохождения и градиентов из обратного прохождения. Далее будут приведены аналогичные шаги при вычислении градиентов для нейронной ОДУ.

Чтобы применить обратное распространение к нейронным ОДУ, необходимо определить величину под названием *адъюнкт (adjoint)*, которая имеет вид:

$$\mathbf{a}(t) = \frac{dL}{dz(t)}. \quad (18.24)$$

Как видно, $\mathbf{a}(T)$ соответствует обычной производной потерь по выходному вектору (см. упражнение 18.6). Слагаемое удовлетворяет собственному дифференциальному уравнению, которое задается как

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \nabla_z f(\mathbf{z}(t), \mathbf{w}), \quad (18.25)$$

что является непрерывной версией цепного правила исчисления. Его можно решить интегрированием в обратном направлении, начиная с $\mathbf{a}(T)$, что опять же можно сделать с помощью ОДУ-решателя «черного ящика». Фактически для этого необходимо сохранить траекторию $\mathbf{z}(t)$, вычисленную на прямом этапе, что может быть проблематично, поскольку обратный решатель может попытаться оценить $\mathbf{z}(t)$ при других значениях t по сравнению с прямым решателем. Вместо этого можно просто позволить обратному решателю пересчитать все необходимые значения $\mathbf{z}(t)$ путем интегрирования (18.22) вместе с (18.25), начиная с выходного значения $\mathbf{z}(T)$.

Третий шаг метода обратного распространения заключается в оценке производных потерь по параметрам сети путем формирования соответствующих произведений активаций и градиентов. Когда значение параметра разделяется между несколькими соединениями в сети, общая производная формируется из суммы производных для каждого из соединений (см. упражнение 9.7). Для нейронного ОДУ, в котором один и тот же вектор параметров \mathbf{w} является общим для всей сети, это суммирование превращается в интегрирование по t (см. упражнение 18.7), которое принимает вид:

$$\nabla_w L = - \int_0^T \mathbf{a}(t)^T \nabla_w f(\mathbf{z}(t), \mathbf{w}) dt.. \quad (18.26)$$

Производные $\nabla_z f$ в (18.25) и $\nabla_w f$ в (18.26) могут быть эффективно вычислены с помощью автоматического дифференцирования (см. раздел 8.2). Обратите

внимание, что приведенные выше результаты можно применить и к более общей нейросетевой функции $\mathbf{f}(\mathbf{z}(t), t, \mathbf{w})$, которая имеет явную зависимость от t в дополнение к неявной зависимости по $\mathbf{z}(t)$.

Одно из преимуществ нейронных ОДУ, обученных с помощью метода со-пряжения, по сравнению с обычными многослойными сетями, заключается в том, что нет необходимости хранить промежуточные результаты прямого распространения, и, следовательно, затраты памяти постоянны. Кроме того, нейронные ОДУ могут естественным образом работать с данными непрерывного времени, в которых наблюдения происходят в произвольные моменты времени. Если функция ошибки L зависит от значений $\mathbf{z}(t)$, отличных от выходного значения, то потребуется несколько запусков решателя обратной модели, по одному запуску для каждой последовательной пары выходов, так что единственное решение разбивается на несколько последовательных для получения доступа к промежуточным состояниям (Chen et al., 2018). Отметим, что высокий уровень точности решателя может использоваться в процессе обучения, а более низкая точность и, следовательно, меньшее количество оценок функций может использоваться в процессе вычисления в задачах с ограниченными вычислительными ресурсами.

18.3.3. Потоки нейронных ОДУ

Нейронные обыкновенные дифференциальные уравнения (ОДУ) дают возможность сформулировать альтернативный подход к построению простых в реализации моделей нормализующихся потоков. Нейронное ОДУ определяет очень гибкое преобразование от входного вектора $\mathbf{z}(0)$ к выходному вектору $\mathbf{z}(T)$ в виде дифференциального уравнения вида

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{f}(\mathbf{z}(t), \mathbf{w}). \quad (18.27)$$

Если определить базовое распределение по входному вектору $p(\mathbf{z}(0))$, то нейронное ОДУ распространяет его вперед по времени, чтобы дать распределение $p(\mathbf{z}(t))$ для каждого значения t , и в результате получается распределение по выходному вектору $p(\mathbf{z}(T))$. В работе (Chen et al., 2018) показано, что для нейронных ОДУ преобразование плотности можно оценить путем интегрирования дифференциального уравнения (см. упражнение 18.8), заданного как

$$\frac{d \ln p(\mathbf{z}(t))}{dt} = -\text{Tr}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{z}(t)}\right), \quad (18.28)$$

где $\partial \mathbf{f} / \partial \mathbf{z}$ представляет собой матрицу Якоби с элементами $\partial f_i / \partial z_j$. Это интегрирование может быть выполнено с помощью стандартных решателей ОДУ. Точно так же выборки из этой плотности могут быть получены путем выборки из базовой плотности $p(\mathbf{z}(0))$, которая задается как простое распределение, например гауссово, и распространения значений на выход путем

интегрирования в (18.27) вновь с помощью решателя ОДУ. Полученная схема известна как *непрерывный нормализующий поток* (*continuous normalizing flow*) (см. упражнение 18.9), а иллюстрация этой схемы приведена на рис. 18.6. Непрерывные нормализующие потоки (см. раздел 18.3.1) можно обучать с использованием метода *сопряженной чувствительности* (*adjoint sensitivity*), применяемого для нейронных ОДУ. Этот метод можно рассматривать в качестве непрерывного временного эквивалента обратного распространения.

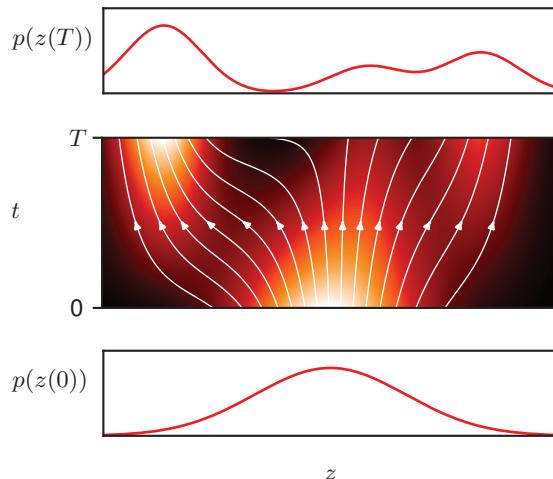


РИС. 18.6 Иллюстрация непрерывного нормализующего потока, где простое гауссово распределение при $t = 0$ непрерывно преобразуется в мультимодальное распределение при $t = T$. Линии потока показывают, как точки вдоль оси z изменяются в зависимости от t . Плотность уменьшается на участках расходления линий потока и увеличивается на участках, где они сходятся

Поскольку в (18.28) используется след якобиана, а не собственно определитель, который появляется при дискретной нормализации потоков, может показаться, что это более эффективно в вычислительном плане. В общем случае вычисление определителя матрицы $D \times D$ требует $\mathcal{O}(D^3)$ операций, тогда как при вычислении следа требуется $\mathcal{O}(D)$ операций. Однако если определитель имеет нижнюю диагональ, как во многих формах нормализующего потока, то он является произведением диагональных членов и, следовательно, также требует $\mathcal{O}(D)$ операций. Поскольку для оценки отдельных элементов матрицы Якоби нужно отдельное прямое распространение, которое само по себе требует $\mathcal{O}(D)$ операций, оценка следа или определителя (для нижней треугольной матрицы) в целом занимает $\mathcal{O}(D^2)$ операций. Однако стоимость оценки следа может быть уменьшена до $\mathcal{O}(D)$ с помощью *оценщика трассировки Хатчinsona* (*Hutchinson's trace estimator*) (Grathwohl et al., 2018), который для матрицы A имеет вид

$$\text{Tr}(A) = \mathbb{E}_\epsilon[\epsilon^T A \epsilon], \quad (18.29)$$

где ϵ – это случайный вектор, распределение которого имеет нулевое среднее значение и единичную ковариацию, например гауссово $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Для конкретного ϵ матрично-векторное произведение $\mathbf{A}\epsilon$ может быть эффективно вычислено за один проход с помощью автоматического дифференцирования в обратном режиме. Затем след можно аппроксимировать с помощью конечного числа выборок в виде

$$\text{Tr}(\mathbf{A}) \simeq \frac{1}{M} \sum_{m=1}^M \epsilon_m^\top \mathbf{A} \epsilon_m. \quad (18.30)$$

На практике можно задать $M = 1$ и использовать только одну выборку, которая обновляется для каждой новой точки данных. Хотя это зашумленная оценка, это может оказаться не слишком существенным, поскольку она является частью зашумленной стохастической процедуры градиентного спуска. Что важно, она является несмешенной (см. упражнение 18.11), что означает, что ожидание оценщика равно истинному значению.

Значительное повышение эффективности обучения для непрерывных нормализующих потоков может быть достигнуто с помощью техники под названием *согласование потоков* (*flow matching*) (Lipman et al., 2022). Эта методика сближает нормализующие потоки (глава 20) с диффузионными моделями и избавляет от необходимости обратного распространения через интегратор, значительно снижая требования к памяти и обеспечивая более быстрый вывод с более стабильным обучением.

Упражнения

18.1 (***) Рассмотрим преобразование $\mathbf{x} = \mathbf{f}(\mathbf{z})$ и его обратное преобразование $\mathbf{z} = \mathbf{g}(\mathbf{x})$. Дифференцируя $\mathbf{x} = \mathbf{f}(\mathbf{g}(\mathbf{x}))$, докажите, что

$$\mathbf{J}\mathbf{K} = \mathbf{I}, \quad (18.31)$$

где \mathbf{I} – это матрица тождества, а \mathbf{J} и \mathbf{K} – это матрицы с элементами

$$J_{ij} = \frac{\partial g_i}{\partial x_j}, \quad K_{ij} = \frac{\partial f_i}{\partial z_j}. \quad (18.32)$$

Используя результат, в котором определитель произведения матриц равен произведению их определителей, докажите, что

$$\det(\mathbf{J}) = \frac{1}{\det(\mathbf{K})}. \quad (18.33)$$

Отсюда следует, что формула (18.1) для преобразования плотности при смене переменных может быть переписана в виде

$$p_x(\mathbf{x}) = p_z(\mathbf{g}(\mathbf{x})) |\det \mathbf{K}|^{-1}, \quad (18.34)$$

где \mathbf{K} оценивается при $\mathbf{z} = \mathbf{g}(\mathbf{x})$.

- 18.2** (*) Рассмотрим последовательность инвертируемых преобразований вида

$$\mathbf{x} = \mathbf{f}_1(\mathbf{f}_2(\cdots \mathbf{f}_{M-1}(\mathbf{f}_M(\mathbf{z})) \cdots)). \quad (18.35)$$

Докажите, что обратная функция имеет вид:

$$\mathbf{z} = \mathbf{f}_M^{-1}(\mathbf{f}_{M-1}^{-1}(\cdots \mathbf{f}_2^{-1}(\mathbf{f}_1^{-1}(\mathbf{x})) \cdots)). \quad (18.36)$$

- 18.3** (*) Рассмотрим линейное преобразование переменных вида

$$\mathbf{x} = \mathbf{z} + \mathbf{b}. \quad (18.37)$$

Докажите, что якобиан этого преобразования является матрицей тождества. Интерпретируйте этот результат, сравнив объем небольшой области пространства \mathbf{z} с объемом соответствующей области пространства \mathbf{x} .

- 18.4** (**) Докажите, что якобиан авторегрессионного нормализующего преобразования потока, заданного в (18.18), является нижней треугольной матрицей. Определитель такой матрицы дается произведением членов на главной диагонали и поэтому может быть легко вычислен.
- 18.5** (*) Рассмотрим уравнение прямого распространения для остаточной сети, приведенное в (18.21), где рассматривается небольшое приращение «временной» переменной t :

$$\mathbf{z}^{(t+\epsilon)} = \mathbf{z}^{(t)} + \epsilon \mathbf{f}(\mathbf{z}^{(t)}, \mathbf{w}). \quad (18.38)$$

Здесь аддитивный вклад от нейронной сети масштабируется с помощью ϵ . Обратите внимание, что (18.21) соответствует случаю $\epsilon = 1$. Взяв предел $\epsilon \rightarrow 0$, выведите дифференциальное уравнение прямого распространения, заданное в (18.22).

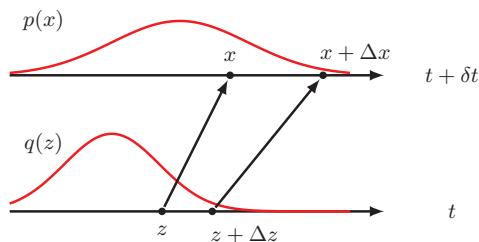
- 18.6** (**) В этом и следующем упражнениях дается неформальный вывод уравнений обратного распространения и оценки градиента для нейронного ОДУ. Более формальный вывод этих результатов можно найти в работе (Chen et al., 2018). Запишите уравнение обратного распространения, соответствующее прямому уравнению (18.38). Взяв предел $\epsilon \rightarrow 0$, выведите уравнение обратного распространения (18.25), где $\mathbf{a}(t)$ определяется по (18.24).
- 18.7** (**) Используя результат (8.10), запишите выражение для градиента функции потерь $L(\mathbf{z}(T))$ для многослойной остаточной сети, заданной в (18.38), где все слои имеют один и тот же вектор параметров \mathbf{w} . Взяв предел $\epsilon \rightarrow 0$, выведите уравнение (18.26) для производной функции потерь.

- 18.8** (★★) В этом упражнении представлен неформальный вывод (18.28) для одномерных распределений. Рассмотрим распределение $q(z)$ в момент времени t , которое преобразуется в новое распределение $p(x)$ в момент времени $t + \delta t$ в результате преобразования из z в x . Также рассмотрим близкие значения z и $z + \Delta z$ вместе с соответствующими значениями x и $x + \Delta x$, как показано на рис. 18.7. Во-первых, запишите уравнение для выражения того, что масса вероятности в интервале Δz такая же, как и в интервале Δx . Во-вторых, запишите уравнение, показывающее изменение плотности вероятности при переходе от t к $t + \delta t$, выраженное через производную $dq(t)/dt$. В-третьих, запишите уравнение для Δx в выражениях Δz , введя функцию $f(z) = dz/dt$. Наконец, объединив эти три уравнения и взяв предел $\delta t \rightarrow 0$, покажите, что

$$\frac{d}{dt} \ln q(z) = -f'(z), \quad (18.39)$$

что является одномерной версией (18.28).

РИС. 18.7 Схематическая иллюстрация преобразования плотностей вероятности, используемого при выводе уравнения для непрерывных нормализующих потоков в одном измерении



- 18.9** (★★) Линии потока на рис. 18.6 построены с помощью набора равномерно распределенных значений и использования обратной кумулятивной функции распределения для каждого значения t при построении соответствующих точек в z -пространстве. Докажите, что это эквивалентно использованию дифференциального уравнения (18.27) для вычисления линий потока, где f определяется в (18.28).
- 18.10** (★★) Используя дифференциальное уравнение (18.27), запишите выражение для базовой плотности непрерывного нормализующего потока в виде выходной плотности, выраженной в виде интеграла по t . Отсюда, воспользовавшись тем, что изменение знака определенного интеграла эквивалентно смене границ этого интеграла, докажите, что вычислительные затраты на инвертирование непрерывного нормализующего потока такие же, как и на оценку прямого потока.
- 18.11** (*) Докажите, что ожидание правой части в оценщике трассировки Хатчинсона (18.30) равно $\text{Tr}(A)$ для любого значения M . Это доказывает, что оценщик не имеет смещения.

Глава 19

Автокодировщики

Главная цель глубокого обучения заключается в определении представлений данных, полезных для последующего использования в одной или нескольких прикладных областях. Один из хорошо известных подходов к обучению внутренним представлениям называется *автоассоциативной нейронной сетью* (*auto-associative neural network*), или *автокодировщиком* (*autoencoder*). Он состоит из нейронной сети с тем же количеством выходов, что и входов, и эта сеть обучается генерировать выход y , близкий к входу x . После обучения внутренний слой нейронной сети выдает представление $z(x)$ для каждого нового входа. Такую сеть можно рассматривать как систему из двух частей. Первая – это кодировщик, который преобразует входной сигнал x в скрытое представление $z(x)$, а вторая – это декодировщик, который преобразует скрытое представление в выходной сигнал $y(z)$.

Для того чтобы автокодировщик находил нетривиальные решения, необходимо ввести некое ограничение, иначе сеть может просто скопировать входные значения в выходные. Это требование можно реализовать, например, ограничив размерность z относительно размерности x или установив условие разреженного представления z . В качестве альтернативы можно заставить сеть находить нетривиальные решения за счет изменения процесса обучения таким образом, чтобы она учились устранивать искажения входных векторов, такие как аддитивный шум или пропущенные значения. Подобные ограничения побуждают сеть обнаруживать в данных интересующие ее структуры для достижения хороших результатов обучения.

В этой главе для начала будут рассмотрены детерминированные автокодировщики, а затем и стохастические модели, которые обучаются распределению кодировщика $p(z|x)$ вместе с распределением декодировщика $p(y|z)$. Эти вероятностные модели известны как *вариационные автокодировщики* (*variational autoencoders*). Они представляют собой третий из четырех подходов к обучению нелинейных моделей латентных переменных (см. раздел 16.4.4).

19.1. Детерминированные автокодировщики

Впервые простая форма автокодировщика была рассмотрена в этой книге в ходе изучения анализа главных компонентов, PCA (см. раздел 16.1). Такая модель обеспечивает линейное преобразование входного вектора на мно-

гообразие меньшей размерности, а полученную проекцию можно приблизенно восстановить в исходном пространстве данных опять же с помощью линейного преобразования. Нелинейность нейронных сетей можно использовать для определения формы нелинейного РСА, где латентное многообразие уже не является линейным подпространством пространства данных. Этого можно достичь путем использования сети с одинаковым количеством выходов и входов, а также путем оптимизации весов таким образом, чтобы минимизировать некоторую меру ошибки реконструкции между входами и выходами по отношению к набору обучающих данных.

Простые автокодировщики редко используются напрямую для современного глубокого обучения, поскольку они не дают семантически значимых представлений в латентном пространстве и не способны напрямую генерировать новые примеры из распределения данных. Однако они служат важной концептуальной основой для некоторых более мощных глубоких генеративных моделей, таких как вариативные автокодировщики (см. раздел 19.2).

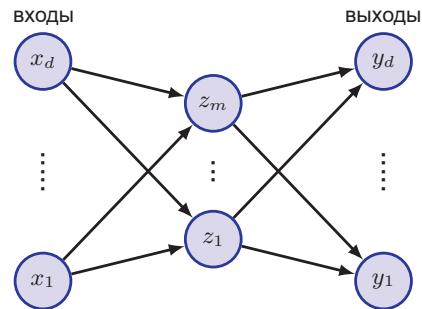
19.1.1. Линейные автокодировщики

Для начала рассмотрим многослойный перцептрон, показанный на рис. 19.1, который содержит D входов, D выходов и M скрытых элементов, при этом $M < D$. Цели, используемые для обучения сети, представляют собой непосредственно входные векторы, так что сеть пытается отобразить каждый входной вектор на себя. О такой сети говорят, что она формирует *автоассоциативное отображение* (*autoassociative mapping*). Поскольку количество скрытых элементов меньше количества входов, идеальная реконструкция всех входных векторов в общем случае невозможна. Поэтому для определения параметров сети w необходимо минимизировать функцию ошибки, которая отражает степень несоответствия между входными векторами и их реконструкциями. В частности, в качестве ошибки можно выбрать сумму квадратов в виде

$$E(w) = \frac{1}{2} \sum_{n=1}^N \|y(x_n, w) - x_n\|^2. \quad (19.1)$$

Если скрытые элементы имеют линейные функции активации, то можно доказать, что функция ошибки имеет единственный глобальный минимум и что в этом минимуме сеть выполняет проекцию на M -мерное подпространство, охватываемое первыми M главными компонентами данных (Bourlard and Kamp, 1988; Baldi and Hornik, 1989). Получается, что векторы весов, которые приводят к скрытым элементам на рис. 19.1, образуют базисный набор, охватывающий главное подпространство. Обратите внимание, что эти векторы не обязательно должны быть ортогональными или нормализованными. Этот результат неудивителен, поскольку и РСА, и нейронные сети основаны на линейном сокращении размерности и минимизируют одну и ту же функцию ошибки по сумме квадратов.

РИС. 19.1 Нейронная сеть автокодировщика с двумя слоями весов. Такая сеть обучается отображать входные векторы на себя путем минимизации ошибки суммы квадратов. Даже при наличии нелинейных блоков в скрытом слое такая сеть эквивалентна линейному анализу главных компонентов. Ссылки, обозначающие параметры смещения, для ясности опущены



Можно было бы предположить, что ограничения линейного многообразия можно преодолеть за счет использования нелинейных функций активации для скрытых элементов сети на рис. 19.1. Однако даже при использовании нелинейных скрытых элементов решение с минимальной ошибкой опять определяется проекцией на подпространство главных компонентов (Bourlard and Kamp, 1988). Поэтому использование двухслойных нейронных сетей для снижения размерности не приносит никаких преимуществ. Стандартные методы PCA, основанные на разложении по сингулярным значениям (singular-value decomposition, SVD), позволяют гарантированно получить правильное решение за определенное время, а также генерируют упорядоченный набор собственных значений с соответствующими ортонормальными собственными векторами.

19.1.2. Глубокие автокодировщики

Однако ситуация меняется при наличии в сети дополнительных нелинейных слоев. Рассмотрим четырехслойную автоассоциативную сеть, показанную на рис. 19.2. И вновь выходные элементы здесь линейны, а элементы M во втором слое также могут быть линейными. Однако первый и третий слои имеют нелинейные сигмоидные функции активации. Сеть по-прежнему обучается путем минимизации функции ошибки по формуле (19.1). Эту сеть можно представить в виде двух последовательных функциональных отображений F_1 и F_2 , как показано на рис. 19.2.

Первое отображение F_1 проецирует исходные D -мерные данные на M -мерное подпространство S , определяемое активациями элементов второго слоя. Поскольку первый слой состоит из нелинейных элементов, это отображение является достаточно общим и не ограничивается линейностью. Точно так же вторая половина сети определяет произвольное функциональное отображение из M -мерного скрытого пространства обратно в исходное D -мерное входное пространство. Такое отображение имеет простую геометрическую интерпретацию, как показано на рис. 19.3 для $D = 3$ и $M = 2$. Функция F_2 из латентного пространства определяет способ, которым многообразие S вкладывается в пространство данных более высокой размерности. Поскольку F_2 может быть нелинейной, вложение S может быть не планарным, как показано

на рисунке. Затем функция F_1 определяет проекцию из исходного D -мерного пространства данных в M -мерное латентное пространство.

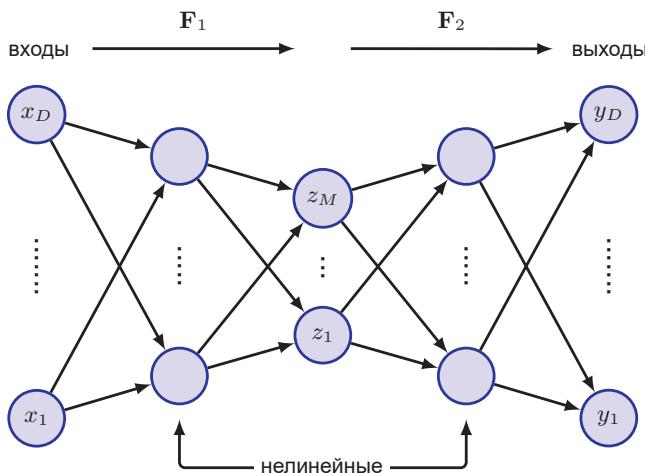


РИС. 19.2 Добавление дополнительных скрытых слоев нелинейных блоков приводит к созданию автоассоциативной сети, которая способна выполнять нелинейное сокращение размерности

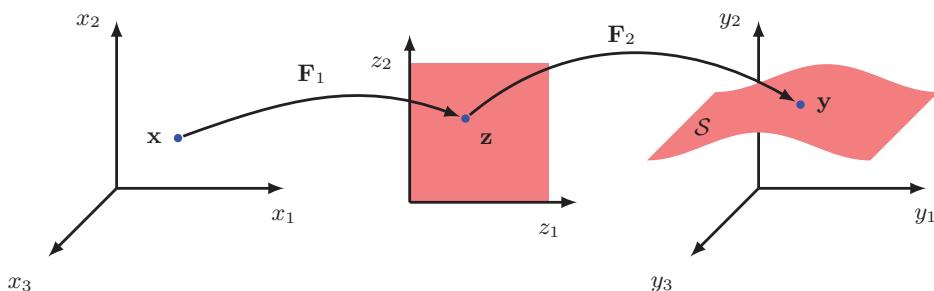


РИС. 19.3 Геометрическая интерпретация отображений, выполняемых сетью на рис. 19.2, для модели с $D = 3$ входами и $M = 2$ элементами во втором слое

Такая сеть позволяет эффективно использовать нелинейную форму РСА. Ее преимущество заключается в том, что она не ограничивается линейными преобразованиями, хотя и содержит стандартный РСА в качестве частного случая. Однако теперь обучение сети требует нелинейной оптимизации, поскольку функция ошибки в (19.1) больше не является квадратичной функцией от параметров сети. Необходимо использовать методы нелинейной оптимизации, требующие больших вычислительных затрат, и существует риск обнаружить субоптимальный локальный минимум функции ошибки. Кроме того, размерность подпространства необходимо уточнять до начала обучения сети.

19.1.3. Разреженные автокодировщики

В качестве альтернативного способа ограничения внутреннего представления вместо ограничения числа узлов в одном из скрытых слоев можно использовать регуляризатор, который будет поддерживать разреженное представление, что приведет к снижению эффективной размерности.

Простым выбором является регуляризатор L_1 (см. раздел 9.2.2), поскольку он стимулирует разреженность и при этом предоставляет регуляризованную функцию ошибки вида

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \sum_{k=1}^K |z_k|, \quad (19.2)$$

где $E(\mathbf{w})$ – это нерегуляризованная ошибка, а сумма по k берется по значениям активации всех блоков в одном из скрытых слоев. Обратите внимание, что регуляризация обычно применяется к параметрам сети, в то время как здесь она используется для активаций блоков. Производные, необходимые для обучения градиентному спуску, как обычно, можно вычислить с помощью автоматического дифференцирования.

19.1.4. Шумоподавляющие автокодировщики

На примере простого автокодировщика можно убедиться в важности ограничений на размерность слоя латентного пространства во избежание ситуации, когда модель просто обучается отображению тождественности. Альтернативный подход, который также побуждает модель выявлять важную внутреннюю структуру в данных, заключается в использовании *шумоподавляющего автокодировщика* (*denoising autoencoder*) (Vincent et al., 2008). Идея заключается в том, чтобы каждый входной вектор \mathbf{x}_n был подвержен воздействию шумовых сигналов, и полученный в результате модифицированный вектор $\tilde{\mathbf{x}}_n$ затем поступает в автокодировщик для получения выходного значения $\mathbf{y}(\tilde{\mathbf{x}}_n, \mathbf{w})$. Сеть обучается восстанавливать исходный входной вектор без шумов посредством минимизации функции ошибки, такой как сумма квадратов, которая задается

$$E(\mathbf{w}) = \sum_{n=1}^N \|\mathbf{y}(\tilde{\mathbf{x}}_n, \mathbf{w}) - \mathbf{x}_n\|^2. \quad (19.3)$$

Одна из форм шума предполагает настройку случайно выбранного подмножества входных переменных на ноль. Доля v таких входов определяет уровень шума и располагается в диапазоне $0 \leq v \leq 1$. Альтернативный подход заключается в добавлении независимого гауссова шума с нулевым средним значением к каждой входной переменной, где масштаб шума задается дисперсией гауссова распределения. В процессе обучения снижению уровня шума во входных данных сети приходится изучать различные аспекты струк-

туры этих данных. Например, если данные представляют собой изображения, то именно информация о высокой корреляции значений близлежащих пикселей позволяет корректировать зашумленные пиксели.

Если говорить более формально, процесс обучения шумоподавляющих автокодировщиков сводится к подбору оценок (Vincent, 2011), где оценка определяется как $s(\mathbf{x}) = \nabla_{\mathbf{x}} \ln p(\mathbf{x})$. Некоторое представление об этой взаимосвязи дано на рис. 19.4. Автокодировщик обучается инвертировать вектор искажений $\tilde{\mathbf{x}}_n - \mathbf{x}_n$ и, следовательно, обучается тому, что для каждой точки в пространстве данных вектор направлен в сторону многообразия и, значит, в область высокой плотности данных. Вектор оценки $\nabla \ln p(\mathbf{x})$ точно так же является вектором, указывающим на область высокой плотности данных. Более подробно взаимосвязь между согласованием оценок и подавлением шумов будет рассмотрена при обсуждении диффузионных моделей, которые также учатся удалять шум из зашумленных входных данных (см. раздел 20.3).

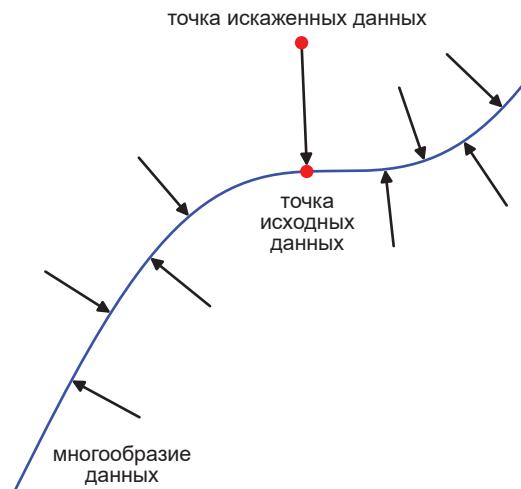


РИС. 19.4 В шумоподавляющем автокодировщике точки данных, которые, как предполагается, находятся в многообразии пространства данных низкой размерности, искажаются аддитивным шумом. Автокодировщик учится возвращать поврежденные точки данных к их исходным значениям и, следовательно, для каждой точки в пространстве данных определяет вектор, указывающий на многообразие

19.1.5. Маскированные автокодировщики

Как было отмечено ранее, модели трансформеров, такие как BERT, могут обучаться обширным внутренним представлениям естественных языков с помощью самоконтроля за счет маскирования случайных подмножеств входных данных (см. раздел 12.2), и поэтому возникает вполне закономерный вопрос: можно ли применить аналогичный подход к естественным изо-

бражениям? В *маскированном автокодировщике* (*masked autoencoder*) (He et al., 2021) глубокая сеть используется по аналогии с шумоподавляющим автокодировщиком для восстановления изображения, полученного на входе в виде его искаженной версии. Однако в данном случае формой искажения является *маскировка* (*masking*), или *выпадение* (*dropping out*), некоторой части входного изображения. Эта техника обычно используется в сочетании с архитектурой визуального трансформера (см. раздел 12.4.1), поскольку в этом случае маскировка части входного сигнала может быть легко реализована путем передачи кодировщику только подмножества случайно выбранных входных лексем. Общий алгоритм вкратце представлен на рис. 19.5. Обратите внимание, что целевой сигнал дополняет входной, так как потери применяются только к маскированным фрагментам. После обучения декодировщик отбрасывается, а кодировщик используется для преобразования изображений во внутреннее представление для использования в последующих задачах.

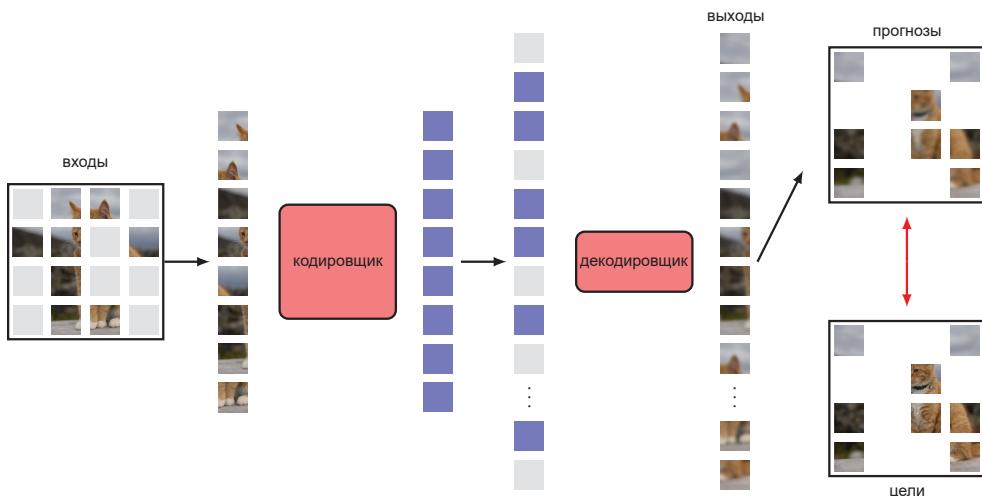


РИС. 19.5 Архитектура маскированного автокодировщика на этапе обучения

По сравнению с языковыми данными в изображениях гораздо больше избыточности и сильных локальных корреляций. Даже удаление одного слова из предложения может значительно увеличить его неоднозначность, в то время как удаление случайного участка изображения, как правило, не оказывает существенного влияния на его семантику. Неудивительно, что лучшие внутренние представления обучаются при маскировании относительно большой части входного изображения, обычно 75 % по сравнению с 15 % маскирования в BERT. При использовании BERT маскированные входы заменяются фиксированной лексемой маски, в то время как в маскированном автокодировщике маскированные участки просто пропускаются. Опуская

большую часть входных фрагментов, можно значительно сэкономить на вычислениях, особенно учитывая тот факт, что вычисления, необходимые для обучения экземпляра трансформера, слабо зависят от длины входной последовательности. Таким образом, маскированный автокодировщик может быть удачным выбором для предварительного обучения больших кодировщиков трансформеров.

Поскольку слой декодировщика также является трансформером, он должен работать в размерности исходного изображения. Так как выход трансформера имеет ту же размерность, что и вход, необходимо восстановить размерность изображения между выходом кодировщика и входом декодировщика. Это достигается путем восстановления маскированных участков, представленных фиксированным вектором лексем маски, при этом каждая лексема дополнена информацией о позиционном кодировании. Из-за более высокой размерности представления декодировщика его трансформер имеет гораздо меньше обучаемых параметров, чем кодировщик. За выходом декодировщика следует обучаемый линейный слой, который отображает выходное представление в пространство значений пикселей, а функция ошибки обучения – это просто средняя квадратичная ошибка, усредненная по отсутствующим фрагментам каждого изображения. Примеры изображений, восстановленных с помощью обученного маскированного автокодировщика, показаны на рис. 19.6. Они демонстрируют способность обученного автокодировщика генерировать семантически правдоподобные реконструкции. Однако конечной целью является обучение полезным внутренним представлениям для последующих задач, для чего декодировщик отбрасывается, а кодировщик применяется к полному изображению без маскировки и со свежим набором выходных слоев, точно настроенных для требуемого применения. Отметим также, что, хотя этот алгоритм изначально был разработан для работы с данными изображений, теоретически он может быть применен к любым формам данных.



РИС. 19.6 Четыре примера изображений, восстановленных с помощью обученного маскированного автокодировщика при маскировании 80 % входных фрагментов. В каждом случае маскированное изображение находится слева, реконструированное изображение – в центре, а исходное изображение – справа. [Из (He et al., 2021) с разрешения авторов]

19.2. Вариационные автокодировщики

Ранее уже было замечено, что функция правдоподобия для модели с латентной переменной

$$p(\mathbf{x} | \mathbf{w}) = \int p(\mathbf{x} | \mathbf{z}, \mathbf{w}) p(\mathbf{z}) d\mathbf{z}, \quad (19.4)$$

в которой $p(\mathbf{x} | \mathbf{z}, \mathbf{w})$ определяется глубокой нейронной сетью, является трудновыполнимой, поскольку интеграл по \mathbf{z} не может быть оценен в аналитической форме. *Вариационный автокодировщик* (*variational autoencoder, VAE*) (Kingma and Welling, 2013; Rezende, Mohamed and Wierstra, 2014; Doersch, 2016; Kingma and Welling, 2019), напротив, при обучении модели оперирует приближением к этому правдоподобию. В VAE есть три ключевые идеи: (i) использование нижней границы доказательства (ELBO) для аппроксимации функции правдоподобия, что приводит к тесной связи с алгоритмом EM (см. раздел 15.3); (ii) *амортизированный вывод* (*amortized inference*), в котором вторая модель, сеть кодировщиков, используется для аппроксимации апостериорных распределений по латентным переменным на шаге E, а не для точной оценки апостериорного распределения для каждой точки данных; и (iii) обеспечение обучаемости модели кодировщиков с помощью метода *перепараметризации* (*reparameterization*).

Рассмотрим генеративную модель с условным распределением $p(\mathbf{x} | \mathbf{z}, \mathbf{w})$ по D -мерной переменной данных \mathbf{x} , управляемой выходом глубокой нейронной сети $\mathbf{g}(\mathbf{z}, \mathbf{w})$. Например, $\mathbf{g}(\mathbf{z}, \mathbf{w})$ может представлять собой среднее значение гауссова условного распределения. Также рассмотрим распределение по M -мерной латентной переменной \mathbf{z} , которое задается гауссовым распределением с нулевым средним значением и единичной дисперсией:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}). \quad (19.5)$$

Чтобы вывести аппроксимацию VAE, сначала вспомним, что для произвольного распределения вероятностей $q(\mathbf{z})$ в пространстве, описываемом латентной переменной \mathbf{z} (см. раздел 15.4), имеет место следующее соотношение:

$$\ln p(\mathbf{x} | \mathbf{w}) = \mathcal{L}(\mathbf{w}) + \text{KL}(q(\mathbf{z}) || p(\mathbf{z} | \mathbf{x}, \mathbf{w})), \quad (19.6)$$

где \mathcal{L} – это *нижняя граница доказательства* (*variational lower bound, ELBO*), также известная как *вариационная нижняя граница*, которая задана как

$$\mathcal{L}(\mathbf{w}) = \int q(\mathbf{z}) \ln \left\{ \frac{p(\mathbf{x} | \mathbf{z}, \mathbf{w}) p(\mathbf{z})}{q(\mathbf{z})} \right\} d\mathbf{z}, \quad (19.7)$$

а расхождение Кульбака–Лейблера $\text{KL}(\cdot || \cdot)$ определяется как

$$\text{KL}(q(\mathbf{z}) || p(\mathbf{z} | \mathbf{x}, \mathbf{w})) = - \int q(\mathbf{z}) \ln \left\{ \frac{p(\mathbf{z} | \mathbf{x}, \mathbf{w})}{q(\mathbf{z})} \right\} d\mathbf{z}. \quad (19.8)$$

Поскольку дивергенция Кульбака–Лейблера удовлетворяет условию KL ($q \parallel p$) > 0 , из этого следует, что

$$\ln p(\mathbf{x} | \mathbf{w}) \geq \mathcal{L}, \quad (19.9)$$

и поэтому \mathcal{L} является нижней границей для $\ln p(\mathbf{x} | \mathbf{w})$. Хотя логарифмическое правдоподобие $\ln p(\mathbf{x} | \mathbf{w})$ является трудноразрешимой задачей, в дальнейшем будет показан способ определения нижней границы с помощью оценки Монте-Карло. Таким образом, она дает приближение к истинному логарифмическому правдоподобию.

Теперь рассмотрим набор обучающих точек данных $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, которые считаются взятыми независимо из модельного распределения $p(\mathbf{x})$. Функция логарифмического правдоподобия для этого набора данных имеет вид:

$$\ln p(\mathcal{D} | \mathbf{w}) = \sum_{n=1}^N \mathcal{L}_n + \sum_{n=1}^N \text{KL}(q_n(\mathbf{z}_n) \parallel p(\mathbf{z}_n | \mathbf{x}_n, \mathbf{w})), \quad (19.10)$$

где

$$\mathcal{L}_n = \int q_n(\mathbf{z}_n) \ln \left\{ \frac{p(\mathbf{x}_n | \mathbf{z}_n, \mathbf{w}) p(\mathbf{z}_n)}{q_n(\mathbf{z}_n)} \right\} d\mathbf{z}_n. \quad (19.11)$$

Обратите внимание, что при этом вводится отдельная латентная переменная \mathbf{z}_n , соответствующая каждому вектору данных \mathbf{x}_n (см. раздел 15.2), как это уже было в моделях смесей и в вероятностной модели РСА. Получается, что каждая латентная переменная имеет свое независимое распределение $q_n(\mathbf{z}_n)$ (см. раздел 16.2), и каждое из них может быть оптимизировано отдельно.

Поскольку выражение (19.10) справедливо для любого выбора распределений $q_n(\mathbf{z})$, можно выбрать распределения с максимизацией ограничения \mathcal{L}_n , или, что эквивалентно, распределения с минимизацией расхождений Кульбака–Лейблера $\text{KL}(q_n(\mathbf{z}_n) \parallel p(\mathbf{z}_n | \mathbf{x}_n, \mathbf{w}))$. Для рассмотренных ранее моделей простой гауссовой смеси и вероятностного РСА можно оценить эти апостериорные распределения непосредственно на шаге Е в ЕМ-алгоритме, что соответствует установке каждого $q_n(\mathbf{z}_n)$ равным соответствующему апостериорному распределению $p(\mathbf{z}_n | \mathbf{x}_n, \mathbf{w})$. Это дает нулевую дивергенцию Кульбака–Лейблера, и, следовательно, нижняя граница равна истинному логарифмическому правдоподобию. Интерпретация апостериорного распределения проиллюстрирована на рис. 19.7 на простом примере, представленном ранее (см. раздел 16.4.1) в контексте генеративных состязательных сетей.

Точное апостериорное распределение \mathbf{z}_n определяется по теореме Байеса как

$$p(\mathbf{z}_n | \mathbf{x}_n, \mathbf{w}) = \frac{p(\mathbf{x}_n | \mathbf{z}_n, \mathbf{w}) p(\mathbf{z}_n)}{p(\mathbf{x}_n | \mathbf{w})}. \quad (19.12)$$

Оценить числитель для такой глубокой генеративной модели несложно. Однако здесь видно, что знаменатель задается функцией правдоподобия,

которая, как уже отмечалось, является трудноразрешимой. Поэтому необходимо найти приближение к апостериорному распределению. В принципе, можно было бы рассмотреть отдельную параметризованную модель для каждого из распределений $q_n(\mathbf{z}_n)$ и оптимизировать каждую модель численно, но это будет очень затратно с вычислительной точки зрения, особенно для больших наборов данных, и, кроме того, пришлось бы заново оценивать распределения после каждого обновления \mathbf{w} . Вместо этого стоит обратиться к другой, более эффективной схеме аппроксимации, основанной на введении второй нейронной сети.

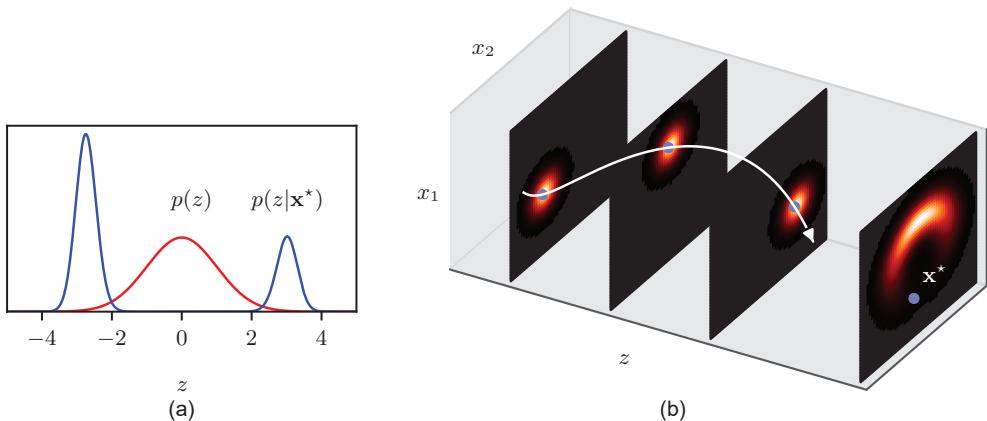


РИС. 19.7 Оценка апостериорного распределения для той же модели, что и на рис. 16.13. Маргинальное распределение $p(\mathbf{x})$, показанное на крайнем правом графике в (b), имеет форму банана, а конкретная точка данных \mathbf{x}^* находится ближе к «рогам» этой формы, чем к середине. Следовательно, апостериорное распределение $p(z|\mathbf{x}^*)$, показанное в (a), является бимодальным, даже если предшествующее распределение $p(z)$ является унимодальным. [Основано на Prince (2020) с разрешения автора]

19.2.1. Амортизированный вывод

При использовании вариационного автокодировщика вместо попыток оценить отдельное апостериорное распределение $p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{w})$ для каждой точки данных \mathbf{x}_n в отдельности выполняется обучение одной нейронной сети, называемой *сетью кодировщика* (*encoder network*), для аппроксимации всех этих распределений. Такая методика называется *амортизированным выводом* (*amortized inference*), и для ее реализации требуется кодировщик, который выдает единственное распределение $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\varphi})$, зависящее от \mathbf{x} , где $\boldsymbol{\varphi}$ – это параметры сети. Целевая функция, задаваемая нижней границей доказательств, теперь зависит как от $\boldsymbol{\varphi}$, так и от \mathbf{w} , и для максимизации этой границы совместно по обоим наборам параметров используются градиентные методы оптимизации.

Таким образом, VAE состоит из двух нейронных сетей, которые имеют независимые параметры, но обучаются совместно: сети кодировщика, которая принимает вектор данных и отображает его в латентное пространство,

и исходной сети, которая принимает вектор латентного пространства и отображает его обратно в пространство данных и которую в этой связи можно рассматривать как *сеть декодировщика* (*decoder network*) (см. раздел 19.1). Это похоже на простую нейросетевую модель автокодировщика, за исключением того, что теперь в ней задается распределение вероятностей над латентным пространством. В дальнейшем будет показано, что кодировщик вычисляет приближенную вероятностную обратную величину декодировщика в соответствии с теоремой Байеса.

Типичным выбором для кодировщика является гауссово распределение с диагональной ковариационной матрицей, средние значения и параметры дисперсии которого, μ_j и σ_j^2 , задаются выходами нейронной сети, принимающей \mathbf{x} в качестве входных данных:

$$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\varphi}) = \prod_{j=1}^M \mathcal{N}(z_j | \mu_j(\mathbf{x}, \boldsymbol{\varphi}), \sigma_j^2(\mathbf{x}, \boldsymbol{\varphi})). \quad (19.13)$$

Обратите внимание, что средние значения $\mu_j(\mathbf{x}, \boldsymbol{\varphi})$ лежат в диапазоне $(-\infty, \infty)$, и поэтому соответствующие функции активации выходных блоков могут быть линейными, в то время как дисперсии $\sigma_j^2(\mathbf{x}, \boldsymbol{\varphi})$ должны быть неотрицательными, и поэтому соответствующие выходные блоки в качестве функции активации обычно используют $\exp(\cdot)$.

Цель состоит в том, чтобы с помощью градиентной оптимизации максимизировать ограничение относительно обоих наборов параметров $\boldsymbol{\varphi}$ и \mathbf{w} , обычно с помощью стохастического градиентного спуска на основе мини-батчей. Хотя оптимизация параметров производится совместно, концептуально можно представить себе чередование оптимизации $\boldsymbol{\varphi}$ и оптимизации \mathbf{w} в духе ЕМ-алгоритма, как показано на рис. 19.8. ((a) Для заданного значения \mathbf{w}_0 параметров сети декодирования \mathbf{w} можно увеличить границу, оптимизировав параметры $\boldsymbol{\varphi}$ сети кодирования. (b) Для заданного значения $\boldsymbol{\varphi}$ можно увеличить значение функции ELBO, оптимизируя \mathbf{w} . Обратите внимание, что функция ELBO, показанная синими кривыми, всегда лежит несколько ниже функции логарифмического правдоподобия, показанной красным цветом, поскольку сеть кодирования, как правило, не может точно соответствовать истинному апостериорному распределению.)

Ключевое отличие от ЕМ заключается в том, что при заданном значении \mathbf{w} оптимизация параметров $\boldsymbol{\varphi}$ кодировщика в общем случае не сводит расходжение Кульбака–Лейблера к нулю, поскольку сеть кодировщика не является идеальным предсказателем апостериорного латентного распределения, и поэтому существует остаточный разрыв между нижней границей и истинным логарифмическим правдоподобием. Хотя кодировщик очень гибок, поскольку основан на глубокой нейронной сети, вряд ли стоит ожидать, что он будет точно моделировать истинное апостериорное распределение, поскольку (i) истинное условное апостериорное распределение не будет фактическим гауссовым, (ii) даже большая нейронная сеть обладает ограниченной гибкостью и (iii) процесс обучения является лишь приблизительной

оптимизацией. Соотношение между EM-алгоритмом и ELBO-оптимизацией представлено на рис. 19.9. ((а) В EM-алгоритме чередуется обновление вариационного апостериорного распределения на шаге E и параметров модели на шаге M. Если шаг E точен, то после каждого шага E разрыв между нижней границей и логарифмическим правдоподобием уменьшается до нуля. (б) В VAE выполняется совместная оптимизация параметров сети кодировщика ϕ (аналогично шагу E) и параметров сети декодировщика w (аналогично шагу M).)

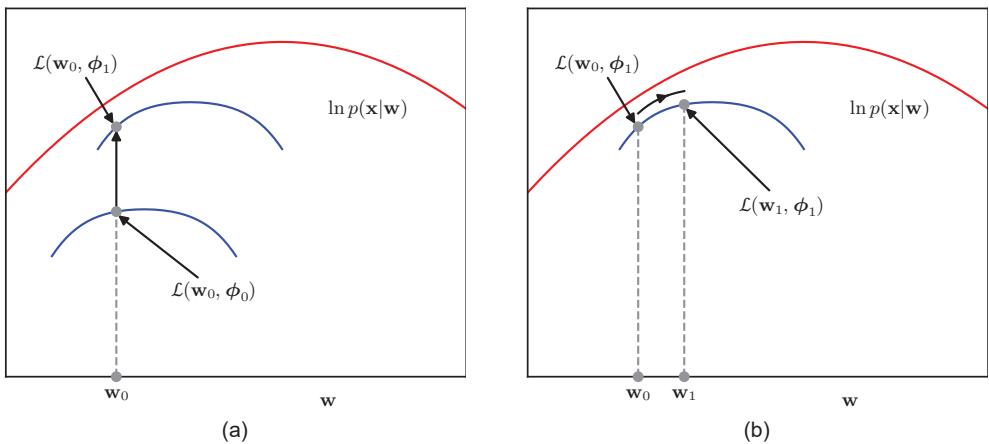


РИС. 19.8 Иллюстрация оптимизации ELBO

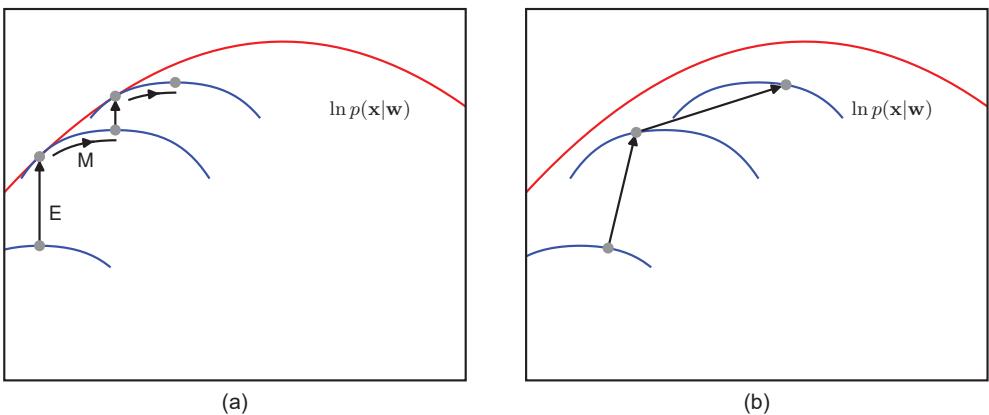


РИС. 19.9 Сравнение EM-алгоритма и ELBO-оптимизации в VAE

19.2.2. Метод перепараметризации

К сожалению, в существующем виде нижняя граница в (19.11) все еще остается трудновычислимой, поскольку она включает интегралы по латентным переменным $\{\mathbf{z}_n\}$, где интеграл имеет сложную зависимость от латентных

переменных из-за сети декодера. Для точки данных \mathbf{x}_n вклад в нижнюю границу можно записать в виде

$$\begin{aligned}\mathcal{L}_n(\mathbf{w}, \boldsymbol{\varphi}) &= \int q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \ln \left\{ \frac{p(\mathbf{x}_n | \mathbf{z}_n, \mathbf{w}) p(\mathbf{z}_n)}{q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi})} \right\} d\mathbf{z}_n \\ &= \int q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \ln p(\mathbf{x}_n | \mathbf{z}_n, \mathbf{w}) d\mathbf{z}_n - \text{KL}(q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \| p(\mathbf{z}_n)).\end{aligned}\quad (19.14)$$

Второй член в правой части представляет собой расхождение Кульбака–Лейблера между двумя гауссовыми распределениями (см. упражнение 2.27) и может быть оценен аналитически:

$$\text{KL}(q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \| p(\mathbf{z}_n)) = \frac{1}{2} \sum_{j=1}^M \{1 + \ln \sigma_j^2(\mathbf{x}_n) - \mu_j^2(\mathbf{x}_n) - \sigma_j^2(\mathbf{x}_n)\}. \quad (19.15)$$

Для первого члена в (19.14) можно попытаться аппроксимировать интеграл по \mathbf{z}_n с помощью простого оценщика Монте–Карло:

$$\int q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \ln p(\mathbf{x}_n | \mathbf{z}_n, \mathbf{w}) d\mathbf{z}_n \approx \frac{1}{L} \sum_{l=1}^L \ln p(\mathbf{x}_n | \mathbf{z}_n^{(l)}, \mathbf{w}), \quad (19.16)$$

где $\{\mathbf{z}_n^{(l)}\}$ – это выборки, взятые из распределения кодировщика $q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi})$. Это легко дифференцируется относительно \mathbf{w} , но градиент относительно $\boldsymbol{\varphi}$ является проблематичным, поскольку изменение $\boldsymbol{\varphi}$ приводит к изменению распределения $q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi})$, из которого извлекаются выборки, и при этом эти выборки имеют фиксированные значения, так что получить производные этих выборок относительно $\boldsymbol{\varphi}$ не представляется возможным. Концептуально можно считать, что процесс фиксации \mathbf{z}_n на конкретном значении выборки блокирует обратное распространение сигнала об ошибке в сети кодировщика, как показано на рис. 19.10.

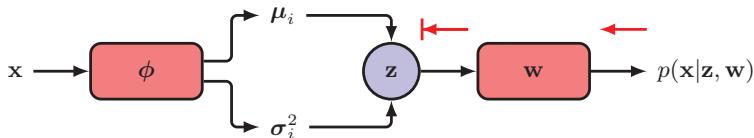


РИС. 19.10 Когда ELBO оценивается путем фиксации латентной переменной \mathbf{z} на выборочном значении, это блокирует обратное распространение сигнала об ошибке в сети кодировщика

Решить эту проблему можно с помощью *метода перепараметризации (re-parameterization trick)*, при котором процедура выборки Монте–Карло меняет формулировку таким образом, что производные по $\boldsymbol{\varphi}$ могут быть вычислены в явном виде. Прежде всего обратите внимание, что если это гауссова случайная переменная с нулевым средним значением и единичной дисперсией, то функция

$$z = \sigma\epsilon + \mu \quad (19.17)$$

будет иметь гауссово распределение со средним значением μ и дисперсией σ^2 (см. упражнение 19.2). Теперь его можно применить к выборкам в (19.16), где μ и σ определяются выходами $\mu_j(\mathbf{x}_n, \varphi)$ и $\sigma_j^2(\mathbf{x}_n, \varphi)$ сети кодировщика, которые представляют собой средние значения и дисперсии в распределении (19.13). Вместо непосредственного получения выборок \mathbf{z}_n следует взять выборки для ϵ и использовать (19.17) для оценки соответствующих выборок для \mathbf{z}_n :

$$z_{nj}^{(l)} = \mu_j(\mathbf{x}_n, \varphi)\epsilon_{nj}^{(l)} + \sigma_j^2(\mathbf{x}_n, \varphi), \quad (19.18)$$

где $l = 1, \dots, L$ – это указатели выборок. Таким образом, зависимость от φ становится явной, и теперь появляется возможность оценивать градиенты относительно φ , как показано на рис. 19.11. Метод перепараметризации можно распространить и на другие распределения, но он ограничен непрерывными переменными. Существуют методы, позволяющие оценивать градиенты напрямую, без применения метода перепараметризации (Williams, 1992), но эти оценки имеют высокую дисперсию, поэтому перепараметризацию также можно рассматривать как метод уменьшения дисперсии.

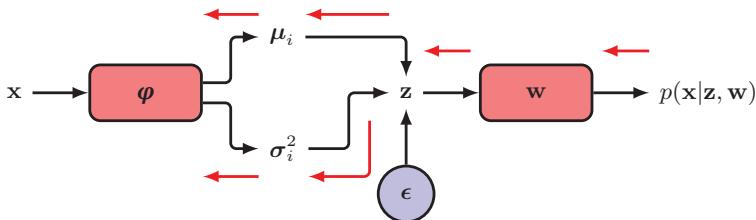


РИС. 19.11 Метод перепараметризации заменяет прямую выборку \mathbf{z} на ту, которая вычисляется по выборке независимой случайной переменной, что позволяет обратно распространять сигнал ошибки в сеть кодирования. Полученную модель можно обучить с помощью градиентной оптимизации для определения параметров сетей кодирования и декодирования

Поэтому полная функция ошибки для VAE с использованием этих конкретных модельных предположений имеет вид:

$$\mathcal{L} = \sum_n \left\{ \frac{1}{2} \sum_{j=1}^M (1 + \ln \sigma_{nj}^2 - \mu_{nj}^2 - \sigma_{nj}^2) + \frac{1}{L} \sum_{l=1}^L \ln p(\mathbf{x}_n | \mathbf{z}_n^{(l)}, \mathbf{w}) \right\}, \quad (19.19)$$

где $\mathbf{z}_n^{(l)}$ имеет компоненты $z_{nj}^{(l)} = \sigma_{nj}\epsilon^{(l)} + \mu_{nj}$, где $\mu_{nj} = \mu_j(\mathbf{x}_n, \varphi)$ и $\sigma_{nj} = \sigma_j(\mathbf{x}_n, \varphi)$, а суммирование по n в (19.19) производится по точкам данных в мини-батче. Число выборок L для каждой точки данных \mathbf{x}_n обычно устанавливается равным 1, так что используется только одна выборка. Хотя такая оценка границы является зашумленной, она является частью шага стохастической градиентной оптимизации, который уже зашумлен, и в целом приводит к более эффективной оптимизации.

Обучение VAE можно описать следующим образом. Для каждой точки данных в мини-батче через сеть кодирования передаются средние значения и отклонения приблизительного латентного распределения, выборка из этого распределения с помощью метода перепараметризации, а затем эти выборки передаются через сеть декодирования для оценки ELBO (19.19). Далее градиенты относительно w и φ оцениваются с помощью автоматического дифференцирования. Обучение VAE кратко описано в алгоритме 19.1, где для ясности опущено упоминание о том, что обычно это делается с помощью мини-батчей. После обучения модели сеть кодирования отбрасывается, а новые точки данных генерируются путем выборки из предшествующего $p(z)$ и прямого распространения через сеть декодирования для получения выборок в пространстве данных.

АЛГОРИТМ 19.1 Обучение вариационного автокодировщика

Input: набор обучающих данных $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$

Сеть кодирования $\{\mu_j(\mathbf{x}_n, \varphi), \sigma_j^2(\mathbf{x}_n, \varphi)\}, j \in \{1, \dots, M\}$

Сеть декодирования $\mathbf{g}(z, w)$

Начальные векторы весов w, φ

Скорость обучения η

Output: итоговые векторы весов w, φ

```

repeat
     $\mathcal{L} \leftarrow 0$ 
    for  $j \in \{1, \dots, M\}$  do
         $\epsilon_{nj} \sim \mathcal{N}(0, 1)$ 
         $z_{nj} \leftarrow \mu_j(\mathbf{x}_n, \varphi) + \sigma_j^2(\mathbf{x}_n, \varphi)^{1/2} \epsilon_{nj}$ 
         $\mathcal{L} \leftarrow \mathcal{L} + \frac{1}{2} \{1 + \ln \sigma_{nj}^2 - \mu_{nj}^2 - \sigma_{nj}^{-2}\}$ 
    end for
     $\mathcal{L} \leftarrow \mathcal{L} + \ln p(\mathbf{x}_n | z_n, w)$ 
     $w \leftarrow w + \eta \nabla_w \mathcal{L}$  // Обновление весов декодирошика
     $\varphi \leftarrow \varphi + \eta \nabla_\varphi \mathcal{L}$  // Обновление весов кодировщика
until converged (до сходимости)
return  $w, \varphi$ 

```

После обучения можно оценить эффективность оценки модели для новой тестовой точки \hat{x} . Поскольку логарифмическое правдоподобие не поддается вычислению, в качестве приближенного значения можно использовать нижнюю границу \mathcal{L} . Для ее оценки достаточно взять выборку из $q(z | \hat{x}, \varphi)$, которая дает более точные прогнозы, чем выборка из $p(z)$.

Существует множество вариантов VAE. При работе с данными изображений кодировщик обычно строится на основе сверток, а декодер – на основе транспонирующих сверток (см. раздел 10.5.3). В *условном (conditional)* VAE и кодировщик, и декодировщик в качестве дополнительного входа используют условную переменную c . Например, может понадобиться генерировать

изображения объектов, в которых \mathbf{c} представляет класс объекта. Априорное распределение $p(\mathbf{z})$ в латентном пространстве вновь может быть простым гауссовым, или оно может быть расширено до условного распределения $p(\mathbf{z}|\mathbf{c})$, задаваемого другой нейронной сетью. Обучение и тестирование проходят в прежнем режиме.

Обратите внимание, что первый член в ELBO (19.14) стимулирует распределение кодировщика $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\varphi})$ быть близким к предшествующему $p(\mathbf{z})$, и, таким образом, модель декодировщика стимулируется к получению реалистичных выходов при генеративном запуске обученной модели путем выборки из $p(\mathbf{z})$. При обучении VAE может возникнуть проблема, когда вариационное распределение $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\varphi})$ сходится к предшествующему распределению $p(\mathbf{z})$ и в результате становится неинформативным, поскольку больше не зависит от \mathbf{x} . По сути, латентный код игнорируется. Это явление называется *апостериорный коллапс* (*posterior collapse*). Симптомом этого является ситуация, когда при взятии входного сигнала, его кодировании и последующем декодировании получается некачественная реконструкция, которая выглядит размытой. В этом случае расхождение Кульбака–Лейблера $KL(q(\mathbf{z}|\mathbf{x}, \boldsymbol{\varphi}) \parallel p(\mathbf{z}))$ близко к нулю.

Другая проблема возникает в том случае, когда латентный код не сжимается, что характеризуется высокой точностью реконструкции, но при этом выходные данные, полученные путем выборки $p(\mathbf{z})$ и пропускания образцов через сеть декодирования, имеют низкое качество и не похожи на обучающие данные. В этом случае расхождение Кульбака–Лейблера относительно велико, и, поскольку обучаемая система имеет вариационное распределение, сильно отличающееся от априорного, выборки из априорного распределения не генерируют реалистичных выходов.

Обе проблемы можно решить путем введения коэффициента β перед первым членом в (19.14) для управления эффективностью регуляризации расхождения Кульбака–Лейблера, где в большинстве случаев $\beta > 1$ (Higgins et al., 2017). Если реконструкции выглядят плохо, то β можно увеличить, а если плохо выглядят выборки, то β можно уменьшить. Значение β также может быть задано по *графику отжига* (*annealing schedule*), при котором оно начинается с небольшого значения и постепенно увеличивается в процессе обучения.

Наконец, следует отметить, что здесь рассматривалась сеть декодирования $\mathbf{g}(\mathbf{z}, \mathbf{w})$, которая представляет собой среднее значение гауссова выходного распределения. Можно расширить VAE, включив выходы, представляющие дисперсию гауссова распределения или, в более общем случае, параметры, характеризующие другие более сложные распределения (см. раздел 6.5).

Упражнения

- 19.1** (**) Докажите, что для любого распределения $q(\mathbf{z}|\boldsymbol{\varphi})$ и любой функции $G(\mathbf{z})$ имеет место следующее выражение:

$$\nabla_{\varphi} \int q(\mathbf{z} | \boldsymbol{\varphi}) G(\mathbf{z}) d\mathbf{z} = \int q(\mathbf{z} | \boldsymbol{\varphi}) G(\mathbf{z}) \nabla_{\varphi} \ln q(\mathbf{z} | \boldsymbol{\varphi}) d\mathbf{z}. \quad (19.20)$$

Отсюда следует, что левая часть (19.20) может быть аппроксимирована с помощью следующего оценщика Монте-Карло:

$$\nabla_{\varphi} \int q(\mathbf{z} | \boldsymbol{\varphi}) G(\mathbf{z}) d\mathbf{z} \simeq \sum_i G(\mathbf{z}^{(i)}) \nabla_{\varphi} \ln q(\mathbf{z}^{(i)} | \boldsymbol{\varphi}), \quad (19.21)$$

где выборки $\{\mathbf{z}^{(i)}\}$ взяты независимо из распределения $q(\mathbf{z} | \boldsymbol{\varphi})$. Проверьте, что эта оценка является несмешенной, а именно что среднее значение правой части (19.21), усредненное по распределению выборок, равно левой части. По сути, если задать $G(\mathbf{z}) = p(\mathbf{x} | \mathbf{z}, \mathbf{w})$, то этот результат позволит оценить градиент второго члена правой части в (19.14) относительно $\boldsymbol{\varphi}$, не прибегая к методу перепараметризации. Кроме того, поскольку этот метод является несмешенным, он позволит получить точный ответ в пределе бесконечного числа выборок. Однако метод перепараметризации более эффективен, т. е. для достижения высокой точности требуется меньшее количество выборок, поскольку он напрямую вычисляет изменение $p(\mathbf{x} | \mathbf{z}, \mathbf{w})$ при изменении \mathbf{z} , которое происходит в результате изменения $\boldsymbol{\varphi}$.

- 19.2** (*) Докажите, что если ϵ имеет гауссово распределение с нулевым средним значением μ и единичной дисперсией σ^2 , то она будет иметь гауссово распределение со средним значением μ и дисперсией σ^2 .
- 19.3** (**) В этом упражнении рассматривается расширение диагональной ковариационной сети кодировщика VAE (19.13) до сети с общей ковариационной матрицей. Рассмотрим K -мерный случайный вектор, взятый из простого гауссова распределения:

$$\epsilon \sim \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I}), \quad (19.22)$$

которое затем линейно преобразуется с помощью соотношения

$$\mathbf{z} = \boldsymbol{\mu} + \mathbf{L}\epsilon, \quad (19.23),$$

где \mathbf{L} – это нижняя треугольная матрица (т. е. матрица $K \times K$, у которой все элементы над главной диагональю равны нулю). Докажите, что \mathbf{z} имеет распределение $\mathcal{N}(\mathbf{z} | \boldsymbol{\mu}, \Sigma)$, и запишите выражение для Σ через \mathbf{L} . Объясните, почему диагональные элементы \mathbf{L} должны быть неотрицательными. Опишите, как $\boldsymbol{\mu}$ и \mathbf{L} могут быть выражены в виде выходов нейронной сети, и определите подходящие варианты функций активации выходных элементов.

- 19.4** (**) Вычислите член дивергенции Кульбака–Лейблера в (19.14). Затем объясните, как можно оценить градиенты этого члена в зависимости от \mathbf{w} и $\boldsymbol{\varphi}$ для обучения сетей кодировщика и декодировщика.

19.5 (*) Как уже было отмечено, ELBO, заданная в (19.11), может быть записана в виде (19.14). Докажите, что ее также можно записать в виде

$$\begin{aligned}\mathcal{L}_n(\mathbf{w}, \boldsymbol{\varphi}) = & \int q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \ln \{p(\mathbf{x}_n | \mathbf{z}_n, \mathbf{w}) p(\mathbf{z}_n)\} d\mathbf{z}_n \\ & - \int q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \ln q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) d\mathbf{z}_n.\end{aligned}\quad (19.24)$$

19.6 (*) Докажите, что ELBO, заданная в (19.11), может быть записана в виде

$$\begin{aligned}\mathcal{L}_n(\mathbf{w}, \boldsymbol{\varphi}) = & \int q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \ln p(\mathbf{z}_n) d\mathbf{z}_n \\ & + \int q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi}) \ln \left\{ \frac{p(\mathbf{x}_n | \mathbf{z}_n, \mathbf{w})}{q(\mathbf{z}_n | \mathbf{x}_n, \boldsymbol{\varphi})} \right\} d\mathbf{z}_n.\end{aligned}\quad (19.25)$$

Глава 20

Диффузионные модели

Ранее было отмечено, что для построения эффективных генеративных моделей можно просто задать распределение $p(\mathbf{z})$ для латентной переменной \mathbf{z} , а затем преобразовать \mathbf{z} в пространство данных \mathbf{x} с помощью глубокой нейронной сети. Для этого вполне достаточно использовать простое фиксированное распределение $p(\mathbf{z})$, например гауссово $\mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I})$, поскольку универсальность нейронной сети превращает его в очень гибкое семейство распределений по \mathbf{x} . В предыдущих главах были рассмотрены различные модели, которые соответствуют этим правилам, но при этом используют разные методы определения и обучения глубокой нейронной сети на основе генеративных состязательных сетей, вариативных автокодировщиков и нормализующих потоков.

В этой главе рассматривается четвертый класс моделей в этой общей структуре (см. раздел 16.4.4), которые известны как *диффузионные модели* (*diffusion models*), также называемые *диффузионными вероятностными моделями шумоподавления* (*denoising diffusion probabilistic model, DDPM*) (Sohl-Dickstein et al., 2015; Ho, Jain and Abbeel, 2020), которые считаются наиболее современными решениями для множества прикладных задач. Для большей наглядности остановимся на моделях данных изображений, хотя сам фреймворк имеет гораздо более широкую область применения. Основная идея заключается в том, чтобы взять каждое обучающее изображение и исказить его с помощью многоступенчатого процесса зашумления с целью преобразования его в выборку из гауссова распределения. Этот процесс показан на рис. 20.1. Затем глубокая нейронная сеть обучается инвертировать этот процесс, и после обучения сеть может генерировать новые изображения, используя в качестве входных данных выборки из гауссова распределения.

Диффузионные модели можно рассматривать как разновидность иерархического вариационного автокодировщика (см. раздел 19.2), в котором распределение кодировщика фиксировано и определяется шумовым процессом, а обучение осуществляется только для генеративного распределения (Luo, 2022). Их легко обучать, они хорошо масштабируются на оборудовании с параллельной архитектурой и позволяют избежать проблем и недостатков состязательного обучения, выдавая при этом результаты, качество которых сопоставимо с генеративными состязательными сетями или даже превосходит их. Однако генерация новых выборок может быть вычислительно затратной

из-за необходимости многократного прохождения через сеть декодирования (Dhariwal and Nichol, 2021).

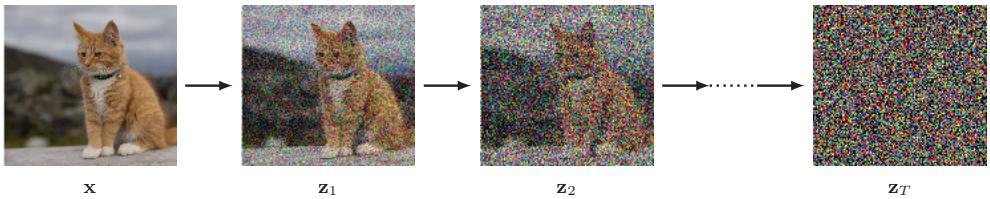


РИС. 20.1 Иллюстрация процесса кодирования в диффузионной модели показывает изображение \mathbf{x} , которое постепенно искажается несколькими этапами аддитивного гауссова шума, давая последовательность все более зашумленных изображений. После большого числа T этапов результат неотличим от выборки, взятой из гауссова распределения. Затем глубокая нейронная сеть обучается обращать этот процесс вспять

20.1. Прямой кодировщик

Предположим, что из обучающего набора берется изображение, которое обозначим \mathbf{x} , и смешивается с гауссовым шумом независимо для каждого пикселя, чтобы получить искаженное шумом изображение \mathbf{z}_1 , определяемое как

$$\mathbf{z}_1 = \sqrt{1 - \beta_1} \mathbf{x} + \sqrt{\beta_1} \boldsymbol{\epsilon}_1, \quad (20.1)$$

где $\boldsymbol{\epsilon}_1 \sim \mathcal{N}(\boldsymbol{\epsilon}_1 | \mathbf{0}, \mathbf{I})$ и $\beta_1 < 1$ – это дисперсия распределения шума. Выбор коэффициентов $\sqrt{1 - \beta_1}$ и $\sqrt{\beta_1}$ (см. упражнение 20.1) в (20.1) и (20.3) гарантирует, что среднее значение распределения \mathbf{z}_t ближе к нулю, чем среднее значение \mathbf{z}_{t-1} , и что дисперсия \mathbf{z}_t ближе к единичной матрице, чем дисперсия \mathbf{z}_{t-1} . Исходя из этого, можно записать преобразование (20.1) (см. упражнение 20.2) в виде

$$q(\mathbf{z}_1 | \mathbf{x}) = \mathcal{N}(\mathbf{z}_1 | \sqrt{1 - \beta_1} \mathbf{x}, \beta_1 \mathbf{I}). \quad (20.2)$$

Затем этот процесс повторяется с дополнительными шагами независимого гауссова шума для получения последовательности все более зашумленных изображений $\mathbf{z}_2, \dots, \mathbf{z}_T$. Обратите внимание, что в литературе по диффузионным моделям эти латентные переменные иногда обозначаются $\mathbf{x}_1, \dots, \mathbf{x}_T$, а наблюданная переменная обозначается \mathbf{x}_0 . Здесь используется обозначение \mathbf{z} для латентных переменных и \mathbf{x} для наблюдаемых переменных в целях соответствия остальной части книги. Каждое последующее изображение задается соотношением:

$$\mathbf{z}_t = \sqrt{1 - \beta_t} \mathbf{z}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon}_t, \quad (20.3)$$

где $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{\epsilon}_t | \mathbf{0}, \mathbf{I})$. Опять же, (20.3) можно записать в виде

$$q(\mathbf{z}_t | \mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t | \sqrt{1 - \beta_t} \mathbf{z}_{t-1}, \beta_t \mathbf{I}). \quad (20.4)$$

Последовательность условных распределений (20.4) образует цепь Маркова (см. раздел 11.3) и может быть выражена в виде вероятностной графовой модели, как показано на рис. 20.2. Исходное изображение \mathbf{x} показано закрашенным узлом, поскольку оно является наблюдаемой переменной, в то время как изображения $\mathbf{z}_1, \dots, \mathbf{z}_t$ рассматриваются как латентные переменные. Шумовой процесс определяется прямым распределением $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ и может рассматриваться как кодировщик. Целью является обучение модели $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})$, которая пытается обратить этот шумовой процесс и которую можно рассматривать как декодировщик. Значения параметров дисперсии $\beta_t \in (0, 1)$ задаются вручную и обычно выбираются таким образом, чтобы значения дисперсии увеличивались по цепи в соответствии с предписанным графиком, чтобы $\beta_1 < \beta_2 < \dots < \beta_T$. Как будет показано далее, условное распределение $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$ играет важную роль в определении процедуры обучения.

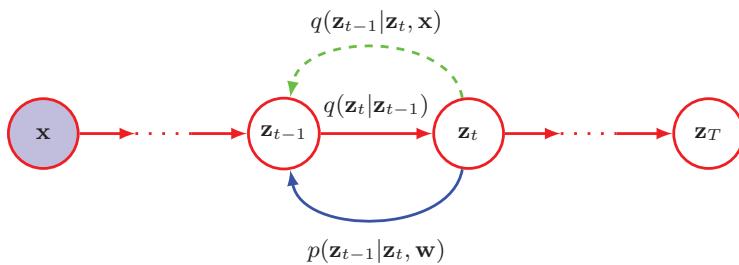


РИС. 20.2 Процесс диффузии, представленный в виде вероятностной графовой модели

20.1.1. Диффузионное ядро

Совместное распределение латентных переменных, обусловленное вектором наблюдаемых данных \mathbf{x} , задается как

$$q(\mathbf{z}_1, \dots, \mathbf{z}_t | \mathbf{x}) = q(\mathbf{z}_1 | \mathbf{x}) \prod_{\tau=2}^t q(\mathbf{z}_\tau | \mathbf{z}_{\tau-1}). \quad (20.5)$$

Если теперь провести маргинализацию по промежуточным переменным $\mathbf{z}_1, \dots, \mathbf{z}_{t-1}$, то получится так называемое *диффузионное ядро* (*diffusion kernel*) (см. упражнение 20.3):

$$q(\mathbf{z}_t | \mathbf{x}) = \mathcal{N}(\mathbf{z}_t | \sqrt{\alpha_t} \mathbf{x}, (1 - \alpha_t) \mathbf{I}), \quad (20.6)$$

где определено

$$\alpha_t = \prod_{\tau=1}^t (1 - \beta_\tau). \quad (20.7)$$

Здесь видно, что каждое промежуточное распределение имеет простое замкнутое гауссово выражение, из которого можно сделать прямую выборку, что может оказаться полезным при обучении DDPM, так как позволяет

эффективно использовать стохастический градиентный спуск со случайно выбранными промежуточными членами в цепи Маркова без необходимости прогона всей цепи. Также можно записать (20.6) в виде

$$\mathbf{z}_t = \sqrt{\alpha_t} \mathbf{x} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_t, \quad (20.8)$$

где вновь $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{\epsilon}_t | \mathbf{0}, \mathbf{I})$. Обратите внимание, что теперь это общий шум, добавленный к исходному изображению, а не инкрементный шум, который был добавлен на этом шаге цепи Маркова.

После множества шагов изображение становится неотличимым от гауссова шума, и в пределе $T \rightarrow \infty$ (см. упражнение 20.4) получается, что

$$q(\mathbf{z}_T | \mathbf{x}) = \mathcal{N}(\mathbf{z}_T | \mathbf{0}, \mathbf{I}), \quad (20.9)$$

и, следовательно, вся информация об исходном изображении теряется. Выбор коэффициентов $\sqrt{1 - \beta_1}$ и $\sqrt{\beta_1}$ в (20.3) гарантирует, что после схождения цепи Маркова к распределению с нулевым средним значением и единичной ковариацией дальнейшие обновления его уже не изменят (см. упражнение 20.5).

Поскольку правая часть (20.9) не зависит от \mathbf{x} , из этого следует, что маргинальное распределение \mathbf{z}_T имеет вид:

$$q(\mathbf{z}_T) = \mathcal{N}(\mathbf{z}_T | \mathbf{0}, \mathbf{I}). \quad (20.10)$$

Цепь Маркова в (20.4) принято называть *прямым процессом* (*forward process*), и он аналогичен кодировщику в VAE, за исключением того, что здесь он фиксирован, а не обучается. Тем не менее обратите внимание, что обычная терминология в литературе противоположна той, которая обычно используется в литературе по нормализации потоков, где прямым процессом считается отображение из латентного пространства в пространство данных.

20.1.2. Условное распределение

Целью этой работы является обучение процессу устранения шума, поэтому будет вполне естественным рассмотреть обратное условное распределение $q(\mathbf{z}_t | \mathbf{z}_{t-1})$, которое можно выразить с помощью теоремы Байеса в виде

$$q(\mathbf{z}_{t-1} | \mathbf{z}_t) = \frac{q(\mathbf{z}_t | \mathbf{z}_{t-1})q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}. \quad (20.11)$$

Маргинальное распределение $q(\mathbf{z}_{t-1})$ можно записать в виде

$$q(\mathbf{z}_{t-1}) = \int q(\mathbf{z}_{t-1} | \mathbf{x}) p(\mathbf{x}) d\mathbf{x}, \quad (20.12)$$

где $q(\mathbf{z}_{t-1} | \mathbf{x})$ задается условным гауссовым распределением (20.6). Однако это распределение является трудновыполнимым, поскольку пришлось бы интегрировать по неизвестной плотности данных $p(\mathbf{x})$. Если аппроксимировать интегрирование с помощью выборок из обучающего набора данных,

то получится сложное распределение, выраженное в виде смеси гауссовых распределений.

Вместо этого рассмотрим условную версию обратного распределения, зависящую от вектора данных \mathbf{x} и определяемую через функцию $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$, которая, как вскоре выяснится, представляет собой простое гауссово распределение. На первый взгляд, это разумно, поскольку в случае зашумленного изображения трудно угадать, какое именно изображение с меньшим шумом привело к его появлению, тогда как при известном начальном изображении задача становится гораздо проще. Это условное распределение можно вычислить с помощью теоремы Байеса:

$$q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x}) = \frac{q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}) q(\mathbf{z}_{t-1} | \mathbf{x})}{q(\mathbf{z}_t | \mathbf{x})}. \quad (20.13)$$

Теперь воспользуемся марковским свойством прямого процесса и запишем

$$q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}) = q(\mathbf{z}_t | \mathbf{z}_{t-1}), \quad (20.14)$$

где правая часть определяется из (20.4). В качестве функции от \mathbf{z}_{t-1} она принимает вид экспоненты квадратичной формы. Член $q(\mathbf{z}_{t-1} | \mathbf{x})$ в числителе (20.13) – это диффузионное ядро, заданное в (20.6), которое вновь содержит экспоненту квадратичной формы относительно \mathbf{z}_{t-1} . Знаменателем в (20.13) можно пренебречь, поскольку в функции от \mathbf{z}_{t-1} он постоянен. Таким образом, получается, что правая часть (20.13) имеет вид гауссова распределения, и можно определить его среднее значение и ковариацию (см. упражнение 20.6) при помощи техники «дополнения до полного квадрата», чтобы получить

$$q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\mathbf{z}_{t-1} | \mathbf{m}_t(\mathbf{x}, \mathbf{z}_t), \sigma_t^2 \mathbf{I}), \quad (20.15)$$

где

$$\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t) = \frac{(1 - \alpha_{t-1})\sqrt{1 - \beta_{t-1}} \mathbf{z}_t + \sqrt{\alpha_{t-1}} \beta_t \mathbf{x}}{1 - \alpha_t}, \quad (20.16)$$

$$\sigma_t^2 = \frac{\beta_t(1 - \alpha_{t-1})}{1 - \alpha_t} \quad (20.17)$$

и используется (20.7).

20.2. Обратное декодирование

Ранее было отмечено, что модель прямого кодировщика определяется последовательностью гауссовых условных распределений $q(\mathbf{z}_t | \mathbf{z}_{t-1})$, но инвертирование этого распределения напрямую приводит к распределению $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$,

которое является невыполнимым, поскольку требует интегрирования по всем возможным значениям начального вектора \mathbf{x} , но его распределение соответствует неизвестному распределению данных $p(\mathbf{x})$, которое и планируется смоделировать. Вместо этого можно получить приближение к обратному распределению с помощью распределения $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})$, управляемого глубокой нейронной сетью, где \mathbf{w} представляет собой веса и смещения сети. Этот обратный шаг аналогичен работе декодера в вариационном автокодировщике (см. главу 19), а его пример показан на рис. 20.2. Здесь красная кривая на правом графике показывает маргинальное распределение $q(z_{t-1})$, представленное смесью трех гауссовых распределений, а левый график показывает прямой шумовой гауссов процесс $q(z_t | z_{t-1})$ в виде распределения по z_t с центром в z_{t-1} . Перемножив их вместе и выполнив нормализацию, получаем распределение $q(z_{t-1} | z_t)$, показанное для конкретного выбора z_t синей кривой. Поскольку распределение слева относительно широкое, что соответствует большой дисперсии β_t , распределение $q(z_{t-1} | z_t)$ имеет сложную мультимодальную структуру. После обучения сети можно сделать выборку из простого гауссова распределения над \mathbf{z}_t и преобразовать ее в выборку из распределения данных $p(\mathbf{x})$ с помощью последовательности шагов обратной выборки путем повторного применения обученной сети.

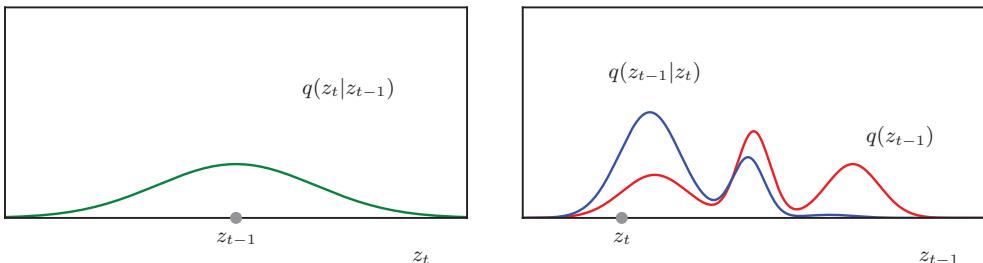


РИС. 20.3 Иллюстрация оценки обратного распределения $q(z_{t-1} | z_t)$ с помощью теоремы Байеса (20.13) для скалярных переменных

Логично предположить, что при сохранении малых дисперсий $\beta_t \ll 1$ изменение латентного вектора между шагами будет относительно небольшим, и поэтому обучение инвертированию преобразования должно быть более легким. Более конкретно, если $\beta_t \ll 1$, то распределение $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$ будет приблизительно гауссовым распределением по \mathbf{z}_{t-1} . Это видно из (20.11), поскольку правая часть зависит от \mathbf{z}_{t-1} через $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ и $q(\mathbf{z}_{t-1})$. Если $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ – это достаточно узкое гауссово распределение, то $q(\mathbf{z}_{t-1})$ будет изменяться лишь незначительно в области, где $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ имеет значительную массу, и, следовательно, $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$ также будет приблизительно гауссовым. Эту теорию можно подтвердить на простом примере, показанном на рис. 20.3 и 20.4. Тем не менее, поскольку дисперсии на каждом шаге малы, необходимо использовать большое количество шагов, чтобы распределение по конечной латентной переменной \mathbf{z}_t , полученное в результате прямого процесса зашум-

ления, по-прежнему было близко к гауссову, а это ведет к увеличению затрат на генерацию новых выборок. На практике количество T может составлять несколько тысяч.

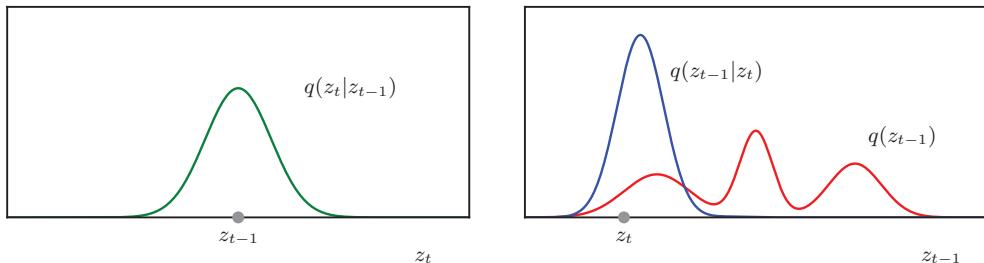


РИС. 20.4 Как и на рис. 20.3, но гауссово распределение $q(z_t | z_{t-1})$ в левой части графика имеет гораздо меньшую дисперсию β_t . Здесь видно, что соответствующее распределение $q(z_{t-1} | z_t)$, показанное синим цветом на правом графике, близко к гауссовому, с такой же дисперсией, как и $q(z_t | z_{t-1})$

Убедитесь в том, что распределение $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$ будет приблизительно гауссовым, можно формально с помощью разложения в ряд Тейлора $\ln q(\mathbf{z}_{t-1})$ около точки \mathbf{z}_t как функции от \mathbf{z}_{t-1} (см. упражнение 20.7). Это также показывает, что при малой дисперсии обратное распределение $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ будет иметь ковариацию, близкую к ковариации $\beta_t \mathbf{I}$ прямого шумового процесса $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$. Поэтому для моделирования обратного процесса используется гауссово распределение вида

$$p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w}) = \mathcal{N}(\mathbf{z}_{t-1} | \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t), \boldsymbol{\beta}_t \mathbf{I}), \quad (20.18)$$

где $\boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t)$ – это глубокая нейронная сеть под управлением набора параметров \mathbf{w} . Обратите внимание, что сеть в явном виде принимает на вход индекс шага t для учета изменения дисперсии β_t на разных шагах цепи. Это дает возможность использовать одну сеть для инвертирования всех шагов цепи Маркова вместо обучения отдельной сети для каждого шага. Также можно обучить ковариации процесса ослабления шума посредством включения в сеть дополнительных выходов для учета кривизны распределения $q(\mathbf{z}_{t-1})$ в окрестностях \mathbf{z}_t (Nichol and Dhariwal, 2021). При выборе архитектуры нейронной сети, используемой для моделирования $\boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t)$, допускается значительная гибкость при условии, что выход имеет ту же размерность, что и вход (см. раздел 10.5.4). Учитывая это ограничение, для приложений обработки изображений обычно выбирается архитектура U-сети.

Общий процесс обратного ослабления шума принимает форму цепи Маркова, заданной как

$$p(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{w}) = p(\mathbf{z}_T) \left\{ \prod_{t=2}^T p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w}) \right\} p(\mathbf{x} | \mathbf{z}_1, \mathbf{w}). \quad (20.19)$$

Здесь предполагается, что $p(\mathbf{z}_T)$ совпадает с распределением $q(\mathbf{z}_T)$ и, следовательно, задается $\mathcal{N}(\mathbf{z}_T | \mathbf{0}, \mathbf{I})$. После обучения модели выборка становится несложной, поскольку вначале производится выборка из простого гауссова распределения $p(\mathbf{z}_T)$, затем последовательно из каждого условного распределения $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})$ по очереди и, наконец, выборка из $p(\mathbf{x} | \mathbf{z}_1, \mathbf{w})$ для получения выборки \mathbf{x} в пространстве данных.

20.2.1. Обучение декодера

Далее необходимо выбрать целевую функцию для обучения нейронной сети. Очевидным выбором является функция правдоподобия, которая для точки данных \mathbf{x} задается как

$$p(\mathbf{x} | \mathbf{w}) = \int \cdots \int p(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{w}) d\mathbf{z}_1 \dots d\mathbf{z}_T, \quad (20.20)$$

где $p(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{w})$ определяется в (20.19). Это пример общей латентно-переменной модели из (16.81), в которой латентные переменные представлены $\mathbf{z} = (\mathbf{z}_1, \dots, \mathbf{z}_T)$, а наблюдаемая переменная – это \mathbf{x} . Обратите внимание, что все латентные переменные имеют ту же размерность, что и пространство данных, как это было в случае нормализации потоков, но не в случае вариационных автокодировщиков или генеративных состязательных сетей. Из (20.20) следует, что вероятность интегрируется по всем возможным траекториям, по которым образцы шума могут привести к наблюдаемой точке данных. Интегралы в (20.20) являются неразрешимыми, поскольку они включают интегрирование по очень сложным функциям нейронной сети.

20.2.2. Нижняя граница доказательств

Поскольку точное значение правдоподобия не поддается точному определению, можно применить подход, аналогичный используемому в вариационных автокодировщиках, и максимизировать нижнюю границу логарифмического правдоподобия (см. раздел 16.3), также известную как *нижняя граница доказательства* (*evidence lower bound, ELBO*), которую в данном случае необходимо представить в контексте диффузионных моделей. Для любого выбора распределения $q(\mathbf{z})$ всегда выполняется следующее соотношение:

$$\ln p(\mathbf{x} | \mathbf{w}) = \mathcal{L}(\mathbf{w}) + \text{KL}(q(\mathbf{z}) \| p(\mathbf{z} | \mathbf{x}, \mathbf{w})), \quad (20.21)$$

где \mathcal{L} – это нижняя граница доказательства, также известная как *вариационная нижняя граница*. Она определяется как

$$\mathcal{L}(\mathbf{w}) = \int q(\mathbf{z}) \ln \left\{ \frac{p(\mathbf{x}, \mathbf{z} | \mathbf{w})}{q(\mathbf{z})} \right\} d\mathbf{z}, \quad (20.22)$$

а расхождение Кульбака–Лейблера $\text{KL}(f \| g)$ между двумя плотностями вероятностей $f(\mathbf{z})$ и $g(\mathbf{z})$ (см. раздел 2.5.7) определяется как

$$\text{KL}(f(\mathbf{z}) \| g(\mathbf{z})) = - \int f(\mathbf{z}) \ln \left\{ \frac{q(\mathbf{z})}{f(\mathbf{z})} \right\} d\mathbf{z}. \quad (20.23)$$

Для проверки соотношения (20.21) сначала следует отметить, что, исходя из правила произведения вероятностей, имеет место

$$p(\mathbf{x}, \mathbf{z} | \mathbf{w}) = p(\mathbf{z} | \mathbf{x}, \mathbf{w})p(\mathbf{x} | \mathbf{w}). \quad (20.24)$$

Подставив (20.24) в (20.22) (см. упражнение 20.8) и воспользовавшись (20.23), получим (20.21). Дивергенция Кульбака–Лейблера обладает свойством $\text{KL}(\cdot \| \cdot) \geq 0$ (см. раздел 2.5.7), из которого следует, что

$$\ln p(\mathbf{x} | \mathbf{w}) \geq \mathcal{L}(\mathbf{w}). \quad (20.25)$$

Так как функция логарифмического правдоподобия является неразрешимой, обучение нейронной сети осуществляется путем максимизации нижней границы $\mathcal{L}(\mathbf{w})$.

Для этого сначала выведем явную форму для нижней границы диффузионной модели. При определении нижней границы можно выбрать любую форму для $q(\mathbf{z})$, лишь бы это было действительное распределение вероятностей, а именно чтобы оно было неотрицательным и интегрировалось в 1. Во многих приложениях ELBO, таких как вариационный автокодировщик, для $q(\mathbf{z})$ выбирается форма с настраиваемыми параметрами – зачастую в виде глубокой нейронной сети – и затем выполняется максимизация ELBO относительно этих параметров, а также относительно параметров распределения $p(\mathbf{x}, \mathbf{z} | \mathbf{w})$. Оптимизация распределения $q(\mathbf{z})$ приводит к тому, что граница становится достаточно точной, что приближает оптимизацию параметров в $p(\mathbf{x}, \mathbf{z} | \mathbf{w})$ к оптимизации максимального правдоподобия. Однако в диффузионных моделях $q(\mathbf{z})$ задается фиксированным распределением $q(\mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{x})$, определяемым цепью Маркова в (20.5), поэтому единственными регулируемыми параметрами являются параметры модели $p(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{w})$ для обратной цепи Маркова. Обратите внимание, что при выборе $q(\mathbf{z})$ учитывается гибкость выбора формы, зависящей от \mathbf{x} .

Поэтому подставим $q(\mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{x})$ в (20.21) с помощью (20.5) и аналогичным образом подставим $p(\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{w})$ с помощью (20.19), что дает возможность записать ELBO в виде

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \mathbb{E}_q \left[\ln \frac{p(\mathbf{z}_T) \left\{ \prod_{t=2}^T p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w}) \right\} p(\mathbf{x} | \mathbf{z}_1, \mathbf{w})}{q(\mathbf{z}_1 | \mathbf{x}) \prod_{t=2}^T q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x})} \right] \\ &= \mathbb{E}_q \left[\ln p(\mathbf{z}_T) + \sum_{t=2}^T \ln \frac{p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x})} - \ln q(\mathbf{z}_1 | \mathbf{x}) + \ln p(\mathbf{x} | \mathbf{z}_1, \mathbf{w}) \right], \end{aligned} \quad (20.26)$$

где определено

$$\mathbb{E}_q[\cdot] \equiv \int \cdots \int q(\mathbf{z}_1 | \mathbf{x}) \prod_{t=2}^T q(\mathbf{z}_t | \mathbf{z}_{t-1}) [\cdot] d\mathbf{z}_1 \dots d\mathbf{z}_T. \quad (20.27)$$

Первый член $\ln p(\mathbf{z}_t)$ в правой части (20.26) – это просто фиксированное распределение $\mathcal{N}(\mathbf{z}_t | \mathbf{0}, \mathbf{I})$. Оно не имеет обучаемых параметров и поэтому может быть опущено в ELBO, поскольку представляет собой фиксированную аддитивную константу. Аналогичным образом третий член, $\ln q(\mathbf{z}_1 | \mathbf{x}_t)$, не зависит от \mathbf{w} , поэтому его также можно опустить.

Четвертый член в правой части (20.26) соответствует члену реконструкции из вариационного автокодировщика. Его можно оценить путем аппроксимации ожидания $\mathbb{E}_q[\cdot]$ оценкой Монте-Карло, полученной путем взятия выборок из распределения по \mathbf{z}_1 , определенного в (20.2), так что

$$\mathbb{E}_q[\ln p(\mathbf{x} | \mathbf{z}_1, \mathbf{w})] \approx \sum_{l=1}^L \ln p(\mathbf{x} | \mathbf{z}_1^{(l)}, \mathbf{w}), \quad (20.28)$$

где $\mathbf{z}_1^{(l)} \sim \mathcal{N}(\mathbf{z}_1 | \sqrt{1 - \beta_1} \mathbf{x}, \beta_1 \mathbf{I})$. В отличие от VAE здесь не требуется обратного распространения ошибки через выборочное значение, поскольку q -распределение фиксировано (см. раздел 19.2.2), и поэтому здесь нет необходимости в применении метода перепараметризации.

Остается второй член в правой части (20.26), который представляет собой сумму членов, каждый из которых зависит от пары соседних значений латентных переменных \mathbf{z}_{t-1} и \mathbf{z}_t . Ранее при выводе ядра диффузии (20.6) выяснилось, что выборку из $q(\mathbf{z}_{t-1} | \mathbf{x})$ можно сделать непосредственно из гауссова распределения, а затем получить соответствующую выборку из \mathbf{z}_t с помощью (20.4), которое также является гауссовым. Хотя в пределе бесконечного числа выборок это было бы правильной процедурой, использование пар выборочных значений создает очень шумные оценки с большой дисперсией, так что требуется неоправданно большое число выборок. Вместо этого нужно переписать ELBO в форме, которая может быть оценена путем выборки только одного значения на член.

20.2.3. Переименование ELBO

После изучения ELBO для вариационного автокодировщика можно перейти к задаче записи ELBO в виде расходимостей Кульбака–Лейблера, которые впоследствии можно будет представить в замкнутой форме. Нейронная сеть является моделью распределения в обратном направлении, $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})$, в то время как q -распределение выражается в прямом направлении, $q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x})$, и поэтому с помощью теоремы Байеса можно обратить условное распределение, представив его в виде

$$q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}) = \frac{q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})q(\mathbf{z}_t | \mathbf{x})}{q(\mathbf{z}_{t-1} | \mathbf{x})}. \quad (20.29)$$

Это позволяет записать второй член в (20.26) в форме

$$\ln \frac{p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x})} = \ln \frac{p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})} + \ln \frac{q(\mathbf{z}_{t-1} | \mathbf{x})}{q(\mathbf{z}_t | \mathbf{x})}. \quad (20.30)$$

Второй член в правой части (20.30) не зависит от \mathbf{w} , и поэтому его можно опустить. Подставляя (20.30) в (20.26), получаем:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_q \left[\sum_{t=2}^T \ln \frac{p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})} + \ln p(\mathbf{x} | \mathbf{z}_1, \mathbf{w}) \right]. \quad (20.31)$$

Наконец, можно переписать (20.31) (см. упражнение 20.9) в виде

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \underbrace{\int q(\mathbf{z}_1 | \mathbf{x}) \ln p(\mathbf{x} | \mathbf{z}_1, \mathbf{w}) d\mathbf{z}_1}_{\text{член реконструкции}} \\ &\quad - \underbrace{\sum_{t=2}^T \int \text{KL}(q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x}) \| p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})) q(\mathbf{z}_t | \mathbf{x}) d\mathbf{z}_t}_{\text{условия согласованности}} \\ &\quad \quad \quad (\text{consistency terms}) \end{aligned} \quad (20.32)$$

где в первом члене упрощается ожидание по $q(\mathbf{z}_1, \dots, \mathbf{z}_T | \mathbf{x})$, поскольку \mathbf{z}_1 – это единственная латентная переменная, фигурирующая в подынтегральном показателе. Поэтому в ожидании, определяемом в (20.27), все условные распределения интегрируются к единице, оставляя только интеграл по \mathbf{z}_1 . Точно так же во втором члене каждый интеграл включает только две соседние латентные переменные \mathbf{z}_{t-1} и \mathbf{z}_t , а все остальные переменные можно не интегрировать.

Теперь граница (20.32) очень похожа на границу ELBO для вариационного автокодировщика в (19.14), за исключением того, что теперь есть несколько этапов кодирования и декодирования. Член реконструкции с высокой степенью вероятности относится к наблюдаемой выборке данных и может быть обучен, как и соответствующий член в VAE (см. главу 19), с помощью аппроксимации выборки (20.28). Члены согласованности в (20.32) определены между парами гауссовых распределений, и поэтому могут быть выражены в замкнутой форме следующим образом. Распределение $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$ задается в (20.15), а распределение $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})$ определяется в (20.18), поэтому расхождение Кульбака–Лейблера (см. упражнение 20.11) приобретает вид

$$\begin{aligned} &\text{KL}(q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x}) \| p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})) \\ &= \frac{1}{2\beta_t} \|\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t) - \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t)\|^2 + \text{const}, \end{aligned} \quad (20.33)$$

где $\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t)$ определяется в (20.16) и где все аддитивные члены, не зависящие от параметров сети \mathbf{w} , поглощены постоянным членом, который не играет никакой роли в обучении. Каждый из членов согласованности в (20.32) имеет один оставшийся интеграл по \mathbf{z}_t , взвешенный по $q(\mathbf{z}_t | \mathbf{x})$. Его можно аппроксимировать с помощью выборки из $q(\mathbf{z}_t | \mathbf{x})$, что может быть эффективно реализовано с помощью диффузионного ядра (20.6).

Как видно, KL-расхождение (20.33) имеет вид простой квадратичной функции потерь. Поскольку параметры сети настраиваются для максимизации нижней границы в (20.32), эта квадратичная ошибка будет минимизирована, так как в ELBO перед членами расхождения Кульбака–Лейблера стоит знак минус.

20.2.4. Прогнозирование шума

Одна из модификаций, которая приводит к более качественным результатам, предполагает изменение роли нейронной сети так, чтобы вместо прогнозирования снижения шума в изображении на каждом шаге цепи Маркова она прогнозировала общий компонент шума, который был добавлен к исходному изображению для создания зашумленного изображения на этом шаге (Ho, Jain and Abbeel, 2020). Для этого сначала возьмем (20.8) и перегруппируем так, чтобы получить

$$\mathbf{x} = \frac{1}{\sqrt{\alpha_t}} \mathbf{z}_t - \frac{\sqrt{1 - \alpha_t}}{\sqrt{\alpha_t}} \boldsymbol{\epsilon}_t. \quad (20.34)$$

Подставив это выражение в (20.16), можно переписать среднее значение $\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t)$ обратного условного распределения $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$ в выражениях исходного вектора данных \mathbf{x} (см. упражнение 20.12) и шума, чтобы получить

$$\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t) = \frac{1}{\sqrt{1 - \beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \boldsymbol{\epsilon}_t \right\}. \quad (20.35)$$

Аналогичным образом вместо нейронной сети $\mu(\mathbf{z}_t, \mathbf{w}, t)$, которая прогнозирует степень зашумленности изображения, введем нейронную сеть $\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t)$, целью которой является прогноз общего шума, добавленного к \mathbf{x} для генерации \mathbf{z}_t . Следуя тем же шагам, которые привели к (20.35), можно сделать вывод, что эти две сетевые функции связаны между собой уравнением:

$$\mu(\mathbf{z}_t, \mathbf{w}, t) = \frac{1}{\sqrt{1 - \beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) \right\}. \quad (20.36)$$

Теперь подставим (20.35) и (20.36) в (20.33), чтобы получить

$$\begin{aligned} & \text{KL}(q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})) \| p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w}) \\ &= \frac{\beta_t}{2(1 - \alpha_t)(1 - \beta_t)} \|\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) - \boldsymbol{\epsilon}_t\|^2 + \text{const} \\ &= \frac{\beta_t}{2(1 - \alpha_t)(1 - \beta_t)} \|\mathbf{g}(\sqrt{\alpha_t} \mathbf{x}_t + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_t, \mathbf{w}, t) - \boldsymbol{\epsilon}_t\|^2 + \text{const}, \end{aligned} \quad (20.37)$$

где в последней строке вместо \mathbf{z}_t подставлено (20.8).

Член реконструкции в ELBO (20.32) может быть аппроксимирован с помощью (20.28) с выборочным значением \mathbf{z}_1 . Используя форму (20.18) для $p(\mathbf{x}|\mathbf{z}, \mathbf{w})$, получаем:

$$\ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w}) = -\frac{1}{2\beta_1} \|\mathbf{x} - \boldsymbol{\mu}(\mathbf{z}_1, \mathbf{w}, 1)\|^2 + \text{const.} \quad (20.38)$$

Если подставить $\boldsymbol{\mu}(\mathbf{z}_1, \mathbf{w}, 1)$ с помощью (20.36) и подставить \mathbf{x} с помощью (20.1) (см. упражнение 20.13), а затем использовать $\alpha_1 = (1 - \beta_1)$, что следует из (20.7), то получим:

$$\ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w}) = -\frac{1}{2(1 - \beta_1)} \|\mathbf{g}(\mathbf{z}_1, \mathbf{w}, 1) - \boldsymbol{\epsilon}_1\|^2 + \text{const.} \quad (20.39)$$

Это точно такая же форма, как и (20.37), но для частного случая $t = 1$, и поэтому члены реконструкции и согласованности можно объединить.

В работе (Ho, Jain and Abbeel, 2020) эмпирическим путем было установлено, что производительность улучшается еще больше, если просто опустить коэффициент $\beta_t/2(1 - \alpha_t)(1 - \beta_t)$ перед (20.37), чтобы все шаги в цепи Маркова имели одинаковый вес. Подстановка этой упрощенной версии (20.37) в (20.33) дает обучающую целевую функцию в виде

$$\mathcal{L}(\mathbf{w}) = -\sum_{t=1}^T \|\mathbf{g}(\sqrt{\alpha_t} \mathbf{x} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_t, \mathbf{w}, t) - \boldsymbol{\epsilon}_t\|^2. \quad (20.40)$$

Квадратичная ошибка в правой части (20.40) имеет очень простую интерпретацию: для заданного шага t в цепи Маркова и для заданной точки обучающих данных \mathbf{x} необходимо выбрать вектор шума $\boldsymbol{\epsilon}_t$ и использовать его для создания соответствующего зашумленного латентного вектора \mathbf{z}_t для этого шага. Функция потерь – это квадратичная разность между прогнозируемым шумом и фактическим шумом. Обратите внимание, что сеть $\mathbf{g}(\cdot, \cdot, \cdot)$ предсказывает общий шум, добавленный к исходному вектору данных \mathbf{x} , а не только дополнительный шум, добавленный на шаге t .

При использовании стохастического градиентного спуска производится оценка вектора градиента функции потерь относительно параметров сети для случайно выбранной точки данных \mathbf{x} из обучающего набора. Кроме того, для каждой такой точки данных случайным образом выбирается шаг t по цепи Маркова, а не оценивается ошибка для каждого члена при суммировании по t в (20.40). Эти градиенты накапливаются на мини-батчах выборок данных и затем используются для обновления весов.

Также обратите внимание, что эта функция потерь автоматически создает форму дополнения данных, поскольку каждый раз, когда используется конкретная обучающая выборка \mathbf{x} , она комбинируется со свежей выборкой шума $\boldsymbol{\epsilon}_t$. Все вышесказанное относится к одной точке данных \mathbf{x} из обучающего набора. Соответствующее вычисление градиента показано в алгоритме 20.1.

АЛГОРИТМ 20.1 Обучение диффузионной вероятностной модели шумоподавления

Input: обучающие данные $\mathcal{D} = \{\mathbf{x}_n\}$

График зашумления $\{\beta_1, \dots, \beta_T\}$

Output: параметры сети w

```

for  $t \in \{1, \dots, T\}$  do
|    $\alpha_t \leftarrow \prod_{t=1}^t (1 - \beta_t)$  // Вычисление альфы из беты
end for
repeat
|    $\mathbf{x} \sim \mathcal{D}$  // Выборка точек данных
|    $t \sim \{1, \dots, T\}$  // Выборка точки по цепи Маркова
|    $\epsilon \sim \mathcal{N}(\epsilon | \mathbf{0}, \mathbf{I})$  // Выборка вектора шума
|    $\mathbf{z}_t \leftarrow \sqrt{\alpha_t} \mathbf{x} + \sqrt{1 - \alpha_t} \epsilon$  // Оценка зашумленной латентной переменной
|    $\mathcal{L}(w) \leftarrow \|\mathbf{g}(\mathbf{z}_t, w, t) - \epsilon\|^2$  // Вычисление члена потерь
|   Take optimizer step // Шаг оптимизатора
until converged (до сходимости)
return  $w$ 

```

20.2.5. Генерация новых выборок

После обучения сети в пространстве данных можно генерировать новые выборки, вначале отбирая из гауссова распределения $p(\mathbf{z}_t)$, а затем последовательно понижая качество на каждом шаге цепи Маркова. После получения очищенной от шумов выборки \mathbf{z}_t на шаге t генерация выборки \mathbf{z}_{t-1} выполняется в три этапа. Сначала оценивается выход нейронной сети, заданный $\mathbf{g}(\mathbf{z}_t, w, t)$. На основе этого результата определяется $\mu(\mathbf{z}_t, w, t)$ с помощью (20.36). Наконец, из $p(\mathbf{z}_{t-1} | \mathbf{z}_t, w) = \mathcal{N}(\mathbf{z}_{t-1} | \mu(\mathbf{z}_t, w, t), \beta_t \mathbf{I})$ генерируется выборка \mathbf{z}_{t-1} путем добавления шума, масштабированного по дисперсии таким образом, что

$$\mathbf{z}_{t-1} = \mu(\mathbf{z}_t, w, t) + \sqrt{\beta_t} \epsilon, \quad (20.41)$$

где $\epsilon \sim \mathcal{N}(\epsilon | \mathbf{0}, \mathbf{I})$. Обратите внимание, что сеть $\mathbf{g}(\cdot, \cdot, \cdot)$ предсказывает общий шум, добавленный к исходному вектору данных \mathbf{x} для получения \mathbf{z}_t , но на шаге выборки из \mathbf{z}_{t-1} вычитается только часть $\beta_t / \sqrt{1 - \alpha_t}$ этого шума, а затем добавляется дополнительный шум с дисперсией β_t для генерации \mathbf{z}_{t-1} . На заключительном этапе, когда производится вычисление синтетической выборки данных \mathbf{x} , дополнительный шум не добавляется, так как целью является получение не зашумленного результата. Процедура выборки кратко описана в алгоритме 20.2.

Основной недостаток диффузионных моделей для генерации данных заключается в том, что они требуют многократного последовательного прохождения выводов через обученную сеть, что может быть затратным в пла-

не вычислительных ресурсов. Один из способов ускорить процесс выборки заключается в предварительном преобразовании процесса удаления шума в дифференциальное уравнение для непрерывного времени, а затем в использовании альтернативных эффективных методов дискретизации (см. раздел 20.3.4) для эффективного решения этого уравнения.

АЛГОРИТМ 20.2 Выборка из диффузионной вероятностной модели шумоподавления

Input: обученная сеть шумоподавления $\mathbf{g}(\mathbf{z}, \mathbf{w}, t)$

График зашумления $\{\beta_1, \dots, \beta_T\}$

Output: вектор выборки \mathbf{x} в пространстве данных

$\mathbf{z}_T \sim \mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I})$ // Выборка из конечного латентного пространства

for $t \in T, \dots, 2$ **do**

$\alpha_t \leftarrow \prod_{\tau=1}^t (1 - \beta_\tau)$ // Вычисление альфы

// Оценка выхода сети

$$\mu(\mathbf{z}_t, \mathbf{w}, t) \leftarrow \frac{1}{\sqrt{1 - \beta_t}} \left\{ \mathbf{z}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) \right\}$$

$\epsilon \sim \mathcal{N}(\mathbf{\epsilon} | \mathbf{0}, \mathbf{I})$ // Выборка вектора шума

$\mathbf{z}_{t-1} \leftarrow \mu(\mathbf{z}_t, \mathbf{w}, t) + \sqrt{\beta_t} \mathbf{\epsilon}$ // Добавление масштабированного шума

end for

$$\mathbf{x} = \frac{1}{\sqrt{1 - \beta_1}} \left\{ \mathbf{z}_1 - \frac{\beta_1}{\sqrt{1 - \alpha_1}} \mathbf{g}(\mathbf{z}_1, \mathbf{w}, t) \right\} // \text{Финальный этап удаления шума}$$

return \mathbf{x}

В этой главе принято допущение, что данные и скрытые переменные непрерывны, и поэтому могут использоваться модели с гауссовым шумом. Модели диффузии могут быть определены и для дискретных пространств (Austin et al., 2021), например для генерации новых молекул-кандидатов лекарств, когда часть процесса генерации включает выбор типов атомов из подмножества химических элементов.

Как уже отмечалось, использование диффузионных моделей может быть сопряжено с большими вычислительными затратами, поскольку они последовательно обращаются к шумовому процессу, который может насчитывать сотни или тысячи шагов. В работе (Song, Meng and Ermon, 2020) представлен родственный метод, называемый *неявными диффузионными моделями шумоподавления* (*denoising diffusion implicit models*), при котором марковское предположение о шумовом процессе ослабляется, но при этом сохраняется та же объективная функция для обучения. Это позволяет ускорить выборку на один-два порядка без ухудшения качества генерируемых выборок.

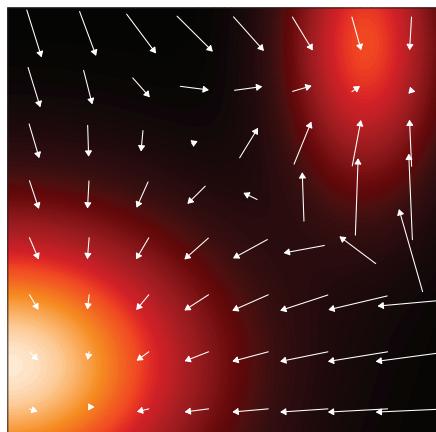
20.3. Соответствие оценок

Рассмотренные ранее в этой главе диффузионные модели удаления шума тесно связаны с другим классом глубоких генеративных моделей, разработанных относительно независимо и основанных на *соответствии оценок* (*score matching*) (Hyvärinen, 2005; Song and Ermon, 2019). В них используется функция оценки (*score function*), или оценка Штейна (*Stein score*), которая определяется как градиент логарифмического правдоподобия относительно вектора данных \mathbf{x} и задается как

$$\mathbf{s}(\mathbf{x}) = \nabla_{\mathbf{x}} \ln p(\mathbf{x}). \quad (20.42)$$

Здесь важно подчеркнуть, что градиент определяется относительно вектора данных, а не относительно какого-либо вектора параметров. Обратите внимание, что $\mathbf{s}(\mathbf{x})$ – это векторная функция той же размерности, что и \mathbf{x} , и что каждый элемент $s_i(\mathbf{x}) = \partial \ln p(\mathbf{x}) / \partial x_i$ связан с соответствующим элементом x_i из \mathbf{x} . Например, если \mathbf{x} – это изображение, то $\mathbf{s}(\mathbf{x})$ также можно представить в виде изображения той же размерности с соответствующими пикселями. На рис. 20.5 показан пример плотности распределения вероятностей в двух измерениях, а также соответствующая функция оценки.

РИС. 20.5 Иллюстрация функции оценки в двух измерениях, показывающая распределение из смеси гауссовых распределений в виде тепловой карты и соответствующую функцию оценки, определенную в (20.42) и показанную в виде векторов на регулярной сетке значений \mathbf{x}



Чтобы убедиться в полезности функции оценки, рассмотрим две функции $q(\mathbf{x})$ и $p(\mathbf{x})$, обладающие тем свойством, что их оценки равны, так что $\nabla_{\mathbf{x}} \ln q(\mathbf{x}) = \nabla_{\mathbf{x}} \ln p(\mathbf{x})$ для всех значений \mathbf{x} . Если проинтегрировать обе стороны уравнения по \mathbf{x} и взять экспоненты, то получится $q(\mathbf{x}) = K p(\mathbf{x})$, где K – это константа, не зависящая от \mathbf{x} . Таким образом, если можно обучить модель $\mathbf{s}(\mathbf{x}, \mathbf{w})$ функции оценки, тогда можно сказать, что удалось смоделировать исходную плотность данных с точностью до мультипликативной константы.

20.3.1. Оценка функции потерь

Для обучения такой модели необходимо определить функцию потерь, которая должна соответствовать модели функции оценки $s(x, w)$ и функции оценки $\nabla_x \ln p(x)$ распределения $p(x)$, которое генерировало эти данные. Примером такой функции потерь является математическое ожидание квадратичной ошибки между модельной оценкой и истинной оценкой, которое задается как

$$J(w) = \frac{1}{2} \int \|s(x, w) - \nabla_x \ln p(x)\|^2 p(x) dx. \quad (20.43)$$

Как уже говорилось при обсуждении моделей на основе энергии (см. раздел 14.3.1), функция оценки не требует нормализации связанной с ней плотности вероятности, поскольку константа нормализации упраздняется градиентным оператором, что обеспечивает значительную гибкость при выборе модели. В целом существует два способа представления функции оценки $s(x, w)$ с помощью глубокой нейронной сети. Каждый элемент s_i в s соответствует одному из элементов x_i в x , поэтому первый подход заключается в том, чтобы сеть имела столько же выходов, сколько и входов. Однако функция оценки определяется как градиент скалярной функции (логарифмической плотности вероятности), которая представляет собой более ограниченный класс функций (см. упражнение 20.14). Поэтому альтернативный способ заключается в использовании сети с одним выходом $\varphi(x)$ и последующем вычислении $\nabla_x \varphi(x)$ с помощью автоматического дифференцирования. Однако этот второй подход требует двух шагов обратного распространения и, следовательно, требует больших вычислительных затрат (см. упражнение 20.15). По этой причине в большинстве задач используется первый способ.

20.3.2. Модифицированная оценка потерь

Одна из проблем при использовании функции потерь (20.43) заключается в отсутствии возможности ее прямой минимизации, поскольку истинная оценка данных $\nabla_x \ln p(x)$ заранее неизвестна. Все, на что можно рассчитывать, – это конечный набор данных $\mathcal{D} = (x_1, \dots, x_N)$, из которого следует построить эмпирическое распределение:

$$p_{\mathcal{D}}(x) = \frac{1}{N} \sum_{n=1}^N \delta(x - x_n). \quad (20.44)$$

Здесь $\delta(x)$ – это дельта-функция Дирака, которую неформально можно представить в виде бесконечно высокого «всплеска» при $x = 0$ со свойствами

$$\delta(x) = 0, x \neq 0, \quad (20.45)$$

$$\int \delta(x) dx = 1. \quad (20.46)$$

Поскольку (20.44) не является дифференцируемой функцией по \mathbf{x} , вычислить ее оценку не удастся. Решить эту проблему можно путем введения модели шума, чтобы «размазать» точки данных и получить гладкое, дифференцируемое представление плотности. Такая модель известна как **оценщик Парзена** (*Parzen estimator*) (см. раздел 3.5.2) или **ядерная оценка плотности** (*kernel density estimator*). Она определяется как

$$q_\sigma(\mathbf{z}) = \int q(\mathbf{z} | \mathbf{x}, \sigma) p(\mathbf{x}) d\mathbf{x}, \quad (20.47)$$

где $q(\mathbf{z} | \mathbf{x}, \sigma)$ – это **шумовое ядро** (*noise kernel*). Обычно в качестве ядра выбирается гауссово распределение:

$$q(\mathbf{z} | \mathbf{x}, \sigma) = \mathcal{N}(\mathbf{z} | \mathbf{x}, \sigma^2 \mathbf{I}). \quad (20.48)$$

Вместо минимизации функции потерь (20.43) можно использовать соответствующую потерю относительно сглаженной плотности Парзена в виде

$$J(\mathbf{w}) = \frac{1}{2} \int \|\mathbf{s}(\mathbf{z}, \mathbf{w}) - \nabla_{\mathbf{z}} \ln q_\sigma(\mathbf{z})\|^2 q_\sigma(\mathbf{z}) d\mathbf{z}. \quad (20.49)$$

Ключевой результат заключается в том, что при подстановке (20.47) в (20.49) можно переписать эту функцию потерь (см. упражнение 20.17) в эквивалентной форме, приведенной в (Vincent, 2011):

$$J(\mathbf{w}) = \frac{1}{2} \iint \|\mathbf{s}(\mathbf{z}, \mathbf{w}) - \nabla_{\mathbf{z}} \ln q(\mathbf{z} | \mathbf{x}, \sigma)\|^2 q(\mathbf{z} | \mathbf{x}, \sigma) p(\mathbf{x}) d\mathbf{z} d\mathbf{x} + \text{const.} \quad (20.50)$$

Если подставить $p(\mathbf{x})$ с использованием эмпирической плотности в (20.44), то получится

$$J(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^N \iint \|\mathbf{s}(\mathbf{z}, \mathbf{w}) - \nabla_{\mathbf{z}} \ln q(\mathbf{z} | \mathbf{x}_n, \sigma)\|^2 q(\mathbf{z} | \mathbf{x}_n, \sigma) d\mathbf{z} + \text{const.} \quad (20.51)$$

Для гауссова ядра Парзена в (20.48) функция оценки имеет вид:

$$\nabla_{\mathbf{z}} \ln q(\mathbf{z} | \mathbf{x}, \sigma) = -\frac{1}{\sigma} \boldsymbol{\epsilon}, \quad (20.52)$$

где $\boldsymbol{\epsilon} = \mathbf{z} - \mathbf{x}$ берется из $\mathcal{N}(\mathbf{z} | \mathbf{0}, \mathbf{I})$. Если рассматривать конкретную модель шума в (20.6), то получается

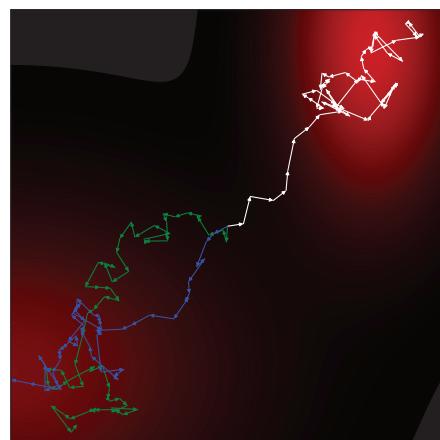
$$\nabla_{\mathbf{z}} \ln q(\mathbf{z} | \mathbf{x}, \sigma) = -\frac{1}{\sqrt{1 - \alpha_t}} \boldsymbol{\epsilon}. \quad (20.53)$$

Таким образом, получается, что оценка потерь (20.50) измеряет разницу между предсказанием нейронной сети и шумом. Следовательно, эта функция потерь имеет тот же минимум, что и форма в (20.37), используемая в диффузионной модели шумоподавления, при этом функция оценки $\mathbf{s}(\mathbf{z}, \mathbf{w})$ играет ту же роль, что и сеть прогнозирования шума $\mathbf{g}(\mathbf{z}, \mathbf{w})$ вплоть до постоянного

масштабирования $-1/\sqrt{1 - \alpha_t}$ (Song and Ermon, 2019). Минимизация в (20.50) известна как *соответствие оценки шумоподавления* (*denoising score matching*), и здесь очевидна тесная связь с диффузионными моделями шумоподавления. Остается вопрос о том, как выбрать дисперсию шума σ^2 , и он будет рассмотрен в ближайшее время.

Обучив модель на основе оценок, необходимо получить новые выборки. Динамика Ланжевена хорошо подходит для моделей на основе оценок, поскольку она построена на функции оценки (см. раздел 14.3) и поэтому не требует нормализованного распределения вероятностей, как показано на рис. 20.6.

РИС. 20.6 Примеры траекторий выборок, полученных с помощью динамики Ланжевена, как определено в (14.61), для распределения, показанного на рис. 20.5, с тремя траекториями, начинаящимися в центре графика



20.3.3. Дисперсия шума

Ранее уже рассматривался способ обучения функции оценки по набору обучающих данных и генерации новых выборок из обученного распределения с помощью выборки Ланжевена. Однако при таком подходе можно выделить три потенциальные проблемы (Song and Ermon, 2019; Luo, 2022). Во-первых, если распределение данных лежит на многообразии меньшей размерности (см. главу 16), чем пространство данных, то плотность вероятности будет равна нулю в точках вне многообразия, и здесь функция оценки не определится, поскольку $\ln p(\mathbf{x})$ не определена. Во-вторых, в областях с низкой плотностью данных прогноз функции оценки может быть неточным, поскольку функция потерь (20.43) имеет вес, зависящий от плотности. Неточность функции оценки может привести к плохим траекториям при использовании выборки Ланжевена. В-третьих, даже при наличии точной модели функции оценки процедура Ланжевена может произвести некорректную выборку (см. упражнение 20.18), если распределение данных представляет собой смесь расходящихся распределений.

Все три проблемы можно решить путем подбора достаточно большого значения дисперсии шума σ^2 , используемой в ядерной функции (20.48), поскольку это размывает распределение данных. Однако слишком большая

дисперсия приведет к значительному искажению исходного распределения, что само по себе вносит неточности в процесс моделирования функции оценки. Этот компромисс можно решить путем анализа последовательности значений дисперсии $\sigma_1^2 < \sigma_2^2 < \dots < \sigma_T^2$ (Song and Ermon, 2019), где σ_1^2 достаточно мала, чтобы распределение данных было представлено точно, а σ_T^2 достаточно велика, чтобы избежать упомянутых выше проблем. Затем оценивающая сеть модифицируется, чтобы принять дисперсию в качестве дополнительного входа $s(\mathbf{z}, \mathbf{w}, \sigma^2)$, и обучается с помощью функции потерь, которая представляет собой взвешенную сумму функций потерь вида (20.51), где каждый член представляет собой ошибку между ассоциированной сетью и соответствующим возмущенным набором данных. Для вектора данных \mathbf{x}_n функция потерь принимает вид:

$$\frac{1}{2} \sum_{i=1}^L \lambda(i) \int \|s(\mathbf{z}, \mathbf{w}, \sigma_i^2) - \nabla_{\mathbf{z}} \ln q(\mathbf{z} | \mathbf{x}_n, \sigma_i)\|^2 q(\mathbf{z} | \mathbf{x}_n, \sigma_i) d\mathbf{z}, \quad (20.54)$$

где $\lambda(i)$ – это весовые коэффициенты. Как видно, эта процедура обучения в точности повторяет ту, что используется для обучения иерархических сетей шумоподавления (см. раздел 20.2.1).

После обучения можно генерировать выборки с помощью нескольких шагов выборки Ланжевена из каждой модели для $i = L, L-1, \dots, 2, 1$ по очереди. Эта техника называется *динамикой Ланжевена с отжигом* (*annealed Langevin dynamics*). Она подобна алгоритму 20.2, используемому для выборки из диффузионных моделей шумоподавления.

20.3.4. Стохастические дифференциальные уравнения

При построении шумового процесса для диффузионной модели, как было показано, полезно использовать большое число шагов, зачастую до нескольких тысяч. Поэтому естественно задаться вопросом: что произойдет, если взять предел бесконечного числа шагов, подобно тому как это было сделано для бесконечно глубоких нейронных сетей, когда были представлены нейронные дифференциальные уравнения (см. раздел 18.3.1). Рассматривая такой предел, необходимо убедиться, что дисперсия шума β_t на каждом шаге становится меньше в соответствии с размером шага. Это приводит к формулировке диффузионных моделей для непрерывного времени в виде *стохастических дифференциальных уравнений* (*stochastic differential equations, SDE*) (Song et al., 2020). Вероятностные диффузионные модели шумоподавления и модели согласования оценок можно рассматривать как дискретизацию SDE в режиме длительного времени.

Общее SDE можно записать в виде бесконечно малого обновления вектора \mathbf{z} в форме

$$d\mathbf{z} = \underbrace{\mathbf{f}(\mathbf{z}, t) dt}_{\text{дрейф}} + \underbrace{g(t) d\mathbf{v}}_{\text{диффузия}}, \quad (20.55)$$

где член дрейфа детерминирован, как в ODE, а член диффузии стохастичен, например задается бесконечно малыми гауссовыми шагами. Здесь параметр t часто называют «временем» по аналогии с физическими системами. Прямой шумовой процесс (20.3) для диффузионной модели может быть записан как SDE в форме (20.55), если взять предел непрерывного времени (см. упражнение 20.19). Для SDE (20.55) существует соответствующее обратное SDE (Song et al., 2020), которое имеет вид

$$dz = f(z, t) - g^2(t) \nabla_z \ln p(z) dt + g(t) dv, \quad (20.56)$$

где $\nabla_z \ln p(z)$ – это функция оценки. SDE, заданное в (20.55), решается в обратном направлении от $t = T$ до $t = 0$.

Для численного решения SDE необходимо выполнить дискретизацию переменной времени. Самый простой подход заключается в использовании фиксированных, равномерно распределенных шагов по времени, что получило название *метода решения Эйлера–Маруямы (Euler–Maruyama solver)*. Для обратной SDE можно получить форму уравнения Ланжевена (см. раздел 14.3). Однако можно использовать и более сложные методы поиска решений, которые предполагают более гибкие формы дискретизации (Kloeden and Platen, 2013).

Для всех диффузионных процессов, управляемых SDE, имеется соответствующий детерминированный процесс, описываемый ODE, траектории которого имеют те же предельные плотности вероятности $p(z|t)$, что и SDE (Song et al., 2020). Для SDE в форме (20.56) соответствующая ODE имеет вид:

$$\frac{dz}{dt} = f(z, t) - \frac{1}{2} g^2(t) \nabla_z \ln p(z). \quad (20.57)$$

Формулировка ODE позволяет использовать эффективные адаптивные пошаговые решения, что значительно сокращает количество оценок функции. Более того, она позволяет связать вероятностные диффузионные модели с нормализующими потоковыми моделями (см. главу 18), на основе которых формула изменения переменных (18.1) может быть использована для точной оценки логарифмического правдоподобия.

20.4. Управляемая диффузия

До сих пор диффузионные модели рассматривались как способ представления необусловленной плотности $p(x)$, полученной из обучающего набора x_1, \dots, x_N , взятых независимо из $p(x)$. После обучения модели появляется возможность генерировать новые выборки из этого распределения. Пример безусловной выборки из глубокой генеративной модели для изображений лиц уже приводился на рис. 1.3, в данном случае из модели GAN.

Однако во многих приложениях требуется выборка из условного распределения $p(x|c)$, где условной переменной c может быть, например, метка

класса или текстовое описание желаемого содержания изображения. Это также является основой для таких приложений, как сверхвысокое разрешение изображений, ретушь изображений, генерация видео и многих других. Самый простой подход к решению этой задачи состоит в том, чтобы рассматривать \mathbf{c} в качестве дополнительного входа нейронной сети шумоподавления $g(\mathbf{z}, \mathbf{w}, t, \mathbf{c})$, а затем обучать сеть на согласованных парах $\{\mathbf{x}_n, \mathbf{c}_n\}$. Основное ограничение этого подхода заключается в том, что сеть может присваивать недостаточный вес обуславливающим переменным или даже игнорировать их. Поэтому необходимо найти способ управления тем, насколько большой вес придается обуславливающей информации, и компенсировать это разнообразием выборок. Это дополнительное воздействие для согласования обусловленной информации называется *наведением* (*guidance*). Существует два основных способа наведения в зависимости от наличия или отсутствия отдельной модели классификатора.

20.4.1. Наведение классификатора

Предположим, что имеется обученный классификатор $p(\mathbf{c} | \mathbf{x})$, и необходимо рассмотреть модель диффузии в аспекте функции оценки. С помощью теоремы Байеса можно записать функцию оценки для условной диффузационной модели в виде

$$\begin{aligned}\nabla_{\mathbf{x}} \ln p(\mathbf{x} | \mathbf{c}) &= \nabla_{\mathbf{x}} \ln \left\{ \frac{p(\mathbf{c} | \mathbf{x})p(\mathbf{x})}{p(\mathbf{c})} \right\} \\ &= \nabla_{\mathbf{x}} \ln p(\mathbf{x}) + \nabla_{\mathbf{x}} \ln p(\mathbf{c} | \mathbf{x}),\end{aligned}\quad (20.58)$$

где используется $\nabla_{\mathbf{x}} \ln p(\mathbf{c}) = 0$, поскольку $p(\mathbf{c})$ не зависит от \mathbf{x} . Первый член в правой части (20.58) – это обычная безусловная функция оценки, а второй член подталкивает процесс удаления шума в направлении максимизации вероятности получения данной метки \mathbf{c} в соответствии с моделью классификатора (Dhariwal and Nichol, 2021). Влиянием классификатора можно управлять, если ввести гиперпараметр λ , называемый *шкалой наведения* (*Guidance scale*, также встречается название «степень соответствия». – Прим. перев.), который определяет вес, придаваемый градиенту классификатора. Тогда функция оценки, используемая для выборки, приобретает вид:

$$\text{score}(\mathbf{x}, \mathbf{c}, \lambda) = \nabla_{\mathbf{x}} \ln p(\mathbf{x}) + \lambda \nabla_{\mathbf{x}} \ln p(\mathbf{c} | \mathbf{x}). \quad (20.59)$$

Если $\lambda = 0$, то восстанавливается исходная безусловная диффузационная модель, а если $\lambda = 1$, то получается оценка, соответствующая условному распределению $p(\mathbf{x} | \mathbf{c})$. При $\lambda > 1$ модель вынуждена учитывать условную метку, и можно использовать значения $\lambda \geq 1$, например $\lambda = 10$. Однако это происходит в ущерб разнообразию выборок, поскольку модель предпочитает «легкие» примеры, которые классификатор может правильно классифицировать.

Одна из проблем подхода к наведению с использованием классификатора заключается в том, что приходится обучать отдельный классификатор. Более

того, этот классификатор должен уметь классифицировать примеры с различной степенью зашумления, в то время как стандартные классификаторы обучаются на чистых примерах. Поэтому далее рассматривается альтернативный подход, который позволяет обойтись без использования отдельного классификатора.

20.4.2. Наведение без классификатора

Если использовать (20.58) для замены $\nabla_{\mathbf{x}} \ln p(\mathbf{c} | \mathbf{x})$ в (20.59), то функцию score (см. упражнение 20.20) можно записать в виде

$$\text{score}(\mathbf{x}, \mathbf{c}, \lambda) = \lambda \nabla_{\mathbf{x}} \ln p(\mathbf{x} | \mathbf{c}) + (1 - \lambda) \nabla_{\mathbf{x}} \ln p(\mathbf{x}), \quad (20.60)$$

которая для $0 < \lambda < 1$ представляет собой выпуклую комбинацию условной логарифмической плотности $\ln p(\mathbf{x} | \mathbf{c})$ и безусловной логарифмической плотности $\ln p(\mathbf{x})$. При $\lambda > 1$ вклад безусловной оценки становится отрицательным. Это означает, что модель активно снижает вероятность генерации выборок, игнорирующих обусловленную информацию, в пользу выборок, которые ее учитывают.

Более того, можно избежать обучения отдельных сетей для моделирования $p(\mathbf{x} | \mathbf{c})$ и $p(\mathbf{x})$, обучив одну обусловленную модель, в которой обусловленная переменная \mathbf{c} во время обучения устанавливается в нулевое значение, например $\mathbf{c} = \mathbf{0}$, с некоторой вероятностью, обычно около 10–20 %. Тогда $p(\mathbf{x})$ будет представлена как $p(\mathbf{x} | \mathbf{c} = \mathbf{0})$. Это в некоторой степени напоминает отсев (см. раздел 9.6.1), при котором обуславливающие входы коллективно устанавливаются на ноль для случайного подмножества обучающих векторов.

После обучения функция оценки (20.60) используется для более точного взвешивания условной информации. На практике наведение без классификатора дает гораздо более качественные результаты, чем наведение с классификатором (Nichol et al., 2021; Saharia et al., 2022). Причина в том, что классификатор $p(\mathbf{c} | \mathbf{x})$ может игнорировать большую часть входного вектора \mathbf{x} до тех пор, пока он делает качественное прогнозирование \mathbf{c} , тогда как наведение без классификатора основано на условной плотности $p(\mathbf{x} | \mathbf{c})$, которая должна присваивать высокую вероятность всем аспектам \mathbf{x} .

Диффузионные модели с текстовым наведением могут использовать методы больших языковых моделей (см. главу 12), позволяя использовать в качестве обуславливающего входа общую текстовую последовательность, известную как *подсказка* (*prompt*), а не просто выбор из заранее определенного набора меток классов. Это позволяет текстовому входу влиять на процесс удаления шума двумя способами: во-первых, объединяя внутреннее представление языковой модели на основе трансформера с входом в сеть шумоподавления, а во-вторых, позволяя слоям перекрестного внимания в сети шумоподавления обращать внимание на последовательность текстовых лексем. На рис. 20.7 показано наведение без классификатора на основе текстовой подсказки.



РИС. 20.7 Иллюстрация наведения диффузионных моделей без классификатора, сгенерированная на основе модели *GLIDE* с использованием условного текста *A stained glass window of a panda eating bamboo* (витраж с изображением панды, поедающей бамбук). Примеры слева сгенерированы при $\lambda = 0$ (никакого наведения, только простая условная модель), а примеры справа – при $\lambda = 3$. [Из (Nichol et al., 2021) с разрешения авторов]

Еще одно применение моделей условной диффузии – это изображения со сверхвысоким разрешением, преобразованные из изображений с низким разрешением. По своей сути это обратная задача, и несколько изображений высокого разрешения будут соответствовать заданному изображению низкого разрешения. Сверхвысокое разрешение может быть достигнуто путем удаления шума выборок высокого разрешения из гауссовой выборки с использованием изображения низкого разрешения в качестве условной переменной (Saharia, Ho et al., 2021). Примеры этого метода показаны на рис. 20.8. В верхнем ряду показано входное изображение 16×16 и соответствующее выходное изображение 128×128 , а также исходное изображение, из которого было сгенерировано входное изображение. В нижнем ряду показано входное изображение 64×64 и выходное изображение 256×256 , опять же с исходным изображением для сравнения. Такие модели могут объединяться каскадом для достижения очень высокого разрешения (Ho et al., 2021), например переходя от 64×64 к 256×256 , а затем от 256×256 к 1024×1024 . Каждый этап обычно представлен U-образной сетевой архитектурой (см. раздел 10.5.4), при этом каждая U-сеть зависит от конечного выхода предыдущей сети с шумоподавлением.

Этот тип построения каскада также может быть использован в диффузионных моделях генерации изображений, где удаление шума изображения выполняется в более низком разрешении, а затем результат подвергается увеличению дискретизации с помощью отдельной сети (которая также может принимать на вход текстовый запрос) для получения конечного результата в высоком разрешении (Nichol et al., 2021; Saharia et al., 2022). Это может

значительно снизить вычислительные затраты по сравнению с работой непосредственно в пространстве высокой размерности, поскольку процесс удаления шума может включать сотни проходов через сеть шумоподавления. Обратите внимание, что эти подходы по-прежнему работают в пространстве изображений напрямую, но с меньшим разрешением.

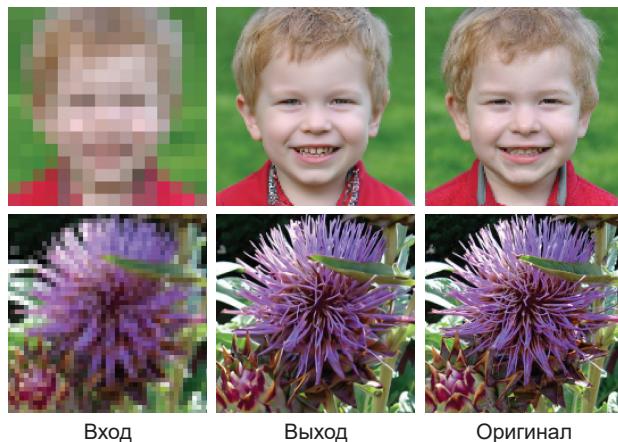


РИС. 20.8 Два примера изображений низкого разрешения с соответствующими выборками изображений высокого разрешения, полученных с помощью диффузионной модели. [Из (Saharia, Ho et al., 2021) с разрешения авторов]

Другой подход к решению проблемы высоких вычислительных затрат, связанных с применением моделей диффузии непосредственно в пространстве изображений высокого разрешения, называется *латентными диффузионными моделями* (*latent diffusion models*) (Rombach et al., 2021) (см. раздел 19.1). Здесь для получения низкоразмерного представления изображений сначала проводится обучение автокодировщика на изображениях без шума, а затем он фиксируется. Далее обучается U-образная сетевая архитектура для удаления шума в низкоразмерном пространстве, которое само по себе не может быть напрямую интерпретировано как изображение. И наконец, представление без шума отображается в пространство изображений высокого разрешения с помощью выходной половины фиксированной сети автокодировщика. Такой подход позволяет более эффективно использовать низкоразмерное пространство, в котором можно сосредоточиться на семантике изображения, оставляя на усмотрение декодера создание соответствующего четкого изображения высокого разрешения из низкоразмерного представления после шумоподавления.

Существует множество других применений обусловленного генерирования изображений, включая ретуширование, кадрирование, восстановление, морфинг, передачу стиля, цветоделение, размытие и генерирование видео (Yang, Srivastava and Mandt, 2022). Пример ретуширования изображен на рис. 20.9.



РИС. 20.9 Пример ретуширования: слева – исходное изображение, посередине – изображение с удаленными участками, справа – результат ретуширования. [Из (Saharia, Chan, Chang et al., 2021) с разрешения авторов]

Упражнения

- 20.1** (*) Используя (20.3), запишите выражения для среднего значения и ковариации \mathbf{z}_t в виде среднего значения и ковариации \mathbf{z}_{t-1} . Докажите, что для $0 < \beta_t < 1$ среднее значение распределения \mathbf{z}_t ближе к нулю, чем среднее значение \mathbf{z}_{t-1} , а ковариация \mathbf{z}_t ближе к единичной матрице \mathbf{I} , чем ковариация \mathbf{z}_{t-1} .
- 20.2** (*) Докажите, что преобразование (20.1) может быть записано в эквивалентной форме (20.2).
- 20.3** (***) В этом упражнении с помощью доказательства по индукции доказывается, что маргинальное распределение \mathbf{x}_t для прямого процесса диффузионной модели, определенного в (20.4), задается в (20.6), где α_t определяется в (20.7). Сначала убедимся, что (20.6) справедливо при $t = 1$. Теперь предположим, что (20.6) справедливо для некоторого конкретного значения t , и выведем соответствующий результат для значения $t + 1$. Для этого проще всего записать прямой процесс в виде представления (20.3) и воспользоваться результатом (3.212), который показывает, что сумма двух независимых гауссовых случайных распределений сама является гауссовой с аддитивными средними значениями и ковариациями.
- 20.4** (*) Используя результат (20.6), где α_t определяется в (20.7), докажите, что в пределе $T \rightarrow \infty$ получается (20.9).
- 20.5** (**) Рассмотрим две независимые случайные величины \mathbf{a} и \mathbf{b} вместе с фиксированным скаляром λ . Докажите, что

$$\text{cov}[\mathbf{a} + \mathbf{b}] = \text{cov}[\mathbf{a}] + \text{cov}[\mathbf{b}], \quad (20.61)$$

$$\text{cov}[\lambda \mathbf{a}] = \lambda^2 \text{cov}[\mathbf{a}]. \quad (20.62)$$

Используйте эти результаты для доказательства того, что если распределение \mathbf{z}_{t-1} имеет нулевое среднее значение и единичную ковариацию, то распределение \mathbf{z}_t , определенное в (20.3), также будет иметь нулевое среднее значение и единичную ковариацию независимо от значения β_t .

- 20.6** (★★★) В этом упражнении выведите результат (20.15) из теоремы Байеса (20.13) с помощью техники возвведения в квадрат. Сначала обратите внимание, что два члена в числителе правой части (20.13), заданные в (20.4) и (20.6), имеют форму экспоненты квадратичных функций \mathbf{z}_{t-1} . Таким образом, искомое распределение является гауссовым, и остается только найти его среднее значение и ковариацию. Для этого рассмотрим только те члены экспоненты, которые зависят от \mathbf{z}_{t-1} , и учтем, что произведение двух экспонент равно экспоненте суммы двух экспонент. Соберите вместе все члены, квадратичные по \mathbf{z}_{t-1} , а также линейные по \mathbf{z}_{t-1} , а затем переставьте их в форме $(\mathbf{z}_{t-1} - \mathbf{m}_t)^T \mathbf{S}_t^{-1} (\mathbf{z}_{t-1} - \mathbf{m}_t)$. Затем непосредственно определите выражения для $\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t)$ и \mathbf{S}_t . Обратите внимание, что аддитивными членами, не зависящими от \mathbf{z}_{t-1} , можно пренебречь.
- 20.7** (★★★) В этом упражнении нужно доказать, что обратное условное распределение $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ для прямого шумового процесса в диффузионной модели может быть аппроксимировано гауссовым распределением при малой дисперсии шума. Рассмотрим обратное условное распределение $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$, определяемое теоремой Байеса, в виде (20.11), где прямое распределение $q(\mathbf{z}_t | \mathbf{z}_{t-1})$ задается в (20.4). Взяв логарифм обеих сторон (20.11), а затем сделав разложение Тейлора для $q(\mathbf{z}_{t-1})$ с центром на значении \mathbf{z}_t , докажите, что для малых значений дисперсии шума β_t распределение $q(\mathbf{z}_{t-1} | \mathbf{z}_t)$ является приблизительно гауссовым со средним значением \mathbf{z}_t и ковариацией $\beta_t \mathbf{I}$. Найдите выражения для поправок низшего порядка к среднему значению и ковариации в виде разложений по степеням β_t .
- 20.8** (★★) Подставив правило произведения вероятностей в форме (20.24) в определение (20.22) ELBO для диффузионной модели и используя определение (20.23) дивергенции Кульбака–Лейблера, докажите, что функция логарифмического правдоподобия может быть записана как сумма нижней границы и дивергенции Кульбака–Лейблера в форме (20.21).
- 20.9** (★★) Докажите, что ELBO для диффузионной модели, заданной в (20.31), можно записать в виде (20.32), где дивергенция Кульбака–Лейблера определяется в (20.23).
- 20.10** (★★) При выводении ELBO для диффузионной модели, заданной в (20.32), первый и третий члены в (20.26) были опущены, поскольку они не зависят от \mathbf{w} . Аналогичным образом был опущен второй член в правой части (20.30), поскольку он также не зависит от \mathbf{w} . Докажите, что при сохранении всех этих опущенных членов в ELBO $\mathcal{L}(\mathbf{x})$ появляется дополнительный член, заданный

$$\text{KL}(q(\mathbf{z}_T | \mathbf{x}) \| p(\mathbf{z}_T)). \quad (20.63)$$

Обратите внимание, что шумовой процесс построен так, что распределение $q(\mathbf{z}_t | \mathbf{x})$ равно гауссову $\mathcal{N}(\mathbf{x} | \mathbf{0}, \mathbf{I})$. Аналогичным образом распределение $p(\mathbf{z}_t)$ определено как равное $\mathcal{N}(\mathbf{x} | \mathbf{0}, \mathbf{I})$, и, следовательно, два распределения в (20.63) равны, а значит, расхождение Кульбака–Лейблера исчезает.

- 20.11** (**) Используя (20.15) для распределения $q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x})$ и (20.18) для распределения $p(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{w})$, докажите, что дивергенция Кульбака–Лейблера, появляющаяся в членах согласованности в (20.32), определяется в (20.33).
- 20.12** (**) Подставив (20.34) в (20.16), перепишите среднее значение $\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t)$ в виде вектора исходных данных \mathbf{x} и шума ϵ в виде (20.35), где α_t определяется в (20.7).
- 20.13** (**) Докажите, что член реконструкции (20.38) в ELBO для диффузионных моделей может быть записан в виде (20.39). Для этого подставьте $\mu(\mathbf{z}_1, \mathbf{w}, 1)$ в (20.36) и подставьте \mathbf{x} в (20.1), а затем используйте $\alpha_1 = (1 - \beta_1)$, что следует из (20.7).
- 20.14** (*) Функция оценки определяется $\mathbf{s}(\mathbf{x}) = \nabla_{\mathbf{x}} p(\mathbf{x} | \mathbf{w})$ и, следовательно, является вектором той же размерности, что и входной вектор \mathbf{x} . Рассмотрим матрицу, элементы которой заданы в виде

$$M_{ij} = \frac{\partial s_i}{\partial x_j} - \frac{\partial s_j}{\partial x_i}. \quad (20.64)$$

Докажите, что если функция оценки определяется через градиент $\mathbf{s} = \nabla_{\mathbf{x}} \varphi(\mathbf{x})$ выхода нейронной сети с одной выходной переменной $\varphi(\mathbf{x})$, то все элементы матрицы $M_{ij} = 0$ для всех пар i, j . Обратите внимание, что если функция оценки $\mathbf{s}(\mathbf{x}) = \nabla_{\mathbf{x}} p(\mathbf{x} | \mathbf{w})$ вместо этого представлена непосредственно глубокой нейронной сетью с тем же количеством выходов, что и входов, то только диагональные элементы матрицы $M_{ii} = 0$, и поэтому выход сети в общем случае не соответствует градиенту какой-либо скалярной функции.

- 20.15** (**) Рассмотрим представление глубокой нейронной сети $\mathbf{s}(\mathbf{x}, \mathbf{w})$ для функции оценки, определенной в (20.42), где \mathbf{x} и \mathbf{s} имеют размерность D . Сравните вычислительную сложность определения оценки для сети с D выходами, которая представляет функцию оценки напрямую, с той, которая вычисляет единственную скалярную функцию $\varphi(\mathbf{x}, \mathbf{w})$, и функция оценки вычисляется косвенно с помощью автоматического дифференцирования. Докажите, что последний подход, как правило, требует больших вычислительных затрат.
- 20.16** (***) Минимизировать функцию оценки (20.43) напрямую невозможно, поскольку функциональная форма истинной плотности данных $p(\mathbf{x})$ неизвестна, и поэтому нельзя записать выражение для функции

оценки $\nabla_x \ln p(\mathbf{x})$. Тем не менее, используя интегрирование по частям (Hyvärinen, 2005), можно переписать (20.43) в виде

$$J(\mathbf{w}) = \int \left\{ \nabla \cdot \mathbf{s}(\mathbf{x}, \mathbf{w}) + \frac{1}{2} \|\mathbf{s}(\mathbf{x}, \mathbf{w})\|^2 \right\} p(\mathbf{x}) d\mathbf{x} + \text{const}, \quad (20.65)$$

где постоянный член не зависит от параметров сети \mathbf{w} , а расходимость $\nabla \cdot \mathbf{s}(\mathbf{x}, \mathbf{w})$ определяется как

$$\nabla \cdot \mathbf{s} = \sum_{i=1}^D \frac{\partial s_i}{\partial x_i} = \sum_{i=1}^D \frac{\partial^2 \ln p(\mathbf{x})}{\partial x_i^2}, \quad (20.66)$$

где D – это размерность \mathbf{x} . Выведите результат (20.65), сначала разложив квадрат в (20.43) и обратив внимание на то, что член с $\|\mathbf{s}(\mathbf{x}, \mathbf{w})\|^2$ уже присутствует в (20.43), тогда как член с $\|\mathbf{s}_d\|^2$ может быть поглощен аддитивной константой, для которой определено $\mathbf{s}_d = \nabla \ln p_d(\mathbf{x})$. Теперь рассмотрим формулу

$$\frac{d}{dx} \{p(x)g(x)\} = \frac{dp(x)}{dx} g(x) + p(x) \frac{dg(x)}{dx} \quad (20.67)$$

для производной от произведения двух функций. Проинтегрируйте обе стороны этой формулы относительно \mathbf{x} и перегруппируйте так, чтобы получить формулу интегрирования по частям:

$$\int_{-\infty}^{\infty} \frac{dp(x)}{dx} g(x) dx = - \int_{-\infty}^{\infty} \frac{dg(x)}{dx} p(x) dx, \quad (20.68)$$

где предполагается, что $p(\infty) = p(-\infty) = 0$. Примените этот результат вместе с определением $\mathbf{s}_d = \nabla \ln p(\mathbf{x})$ к члену с участием $\mathbf{s}(\mathbf{x}, \mathbf{w})^T \mathbf{s}_d$ для завершения доказательства. Обратите внимание, что оценка вторых производных в (20.66) требует отдельного прохода обратного распространения для каждой производной и, следовательно, имеет общие вычислительные затраты, которые растут квадратично с размерностью D пространства данных (Martens, Sutskever and Swersky, 2012). Это исключает прямое применение данной функции потерь в пространствах высокой размерности, поэтому для устранения этой неэффективности были разработаны такие методы, как *согласование по нарезанным оценкам* (sliced score matching) (Song et al., 2019).

- 20.17** (**) В этом упражнении нужно доказать, что функция потерь (20.50) эквивалентна вплоть до аддитивной константы форме (20.49). Для этого сначала разверните квадрат в (20.49) и, используя (20.47), докажите, что член в $\mathbf{s}^T \mathbf{s}$ из (20.49) совпадает с соответствующим членом, полученным при развертывании квадрата в (20.50). Далее обратите внимание, что член $\|\nabla_z \ln q\|^2$ в (20.49) не зависит от \mathbf{w} и аналогично соответствующий член в (20.50) также не зависит от \mathbf{w} , поэтому их

можно рассматривать как аддитивные константы в функции потерь, не играющие никакой роли в обучении. Наконец, обратимся к перекрестному члену в (20.49). Подставив $q(\mathbf{z})$ в (20.47), докажите, что оно равно соответствующему перекрестному члену из (20.50). Следовательно, нужно доказать, что эти две функции потерь равны с точностью до аддитивной константы.

- 20.18** (*) Рассмотрим распределение вероятностей, состоящее из смеси двух несмешивающихся распределений (т. е. распределений, обладающих тем свойством, что когда одно из них ненулевое, то другое должно быть нулевым) вида

$$p(\mathbf{x}) = \lambda p_A(\mathbf{x}) + (1 - \lambda)p_B(\mathbf{x}). \quad (20.69)$$

Докажите, что, когда функция оценки, определенная в (20.42), вычисляется для любой заданной точки \mathbf{x} , коэффициент смещивания λ не появляется. Отсюда следует, что динамика Ланжевена, определяемая в (14.61), не будет делать выборки из двухкомпонентных распределений с корректными пропорциями. Эта проблема решается добавлением шума из широкого распределения, как обсуждалось в этой главе.

- 20.19** (**) Для дискретных шагов прямой шумовой процесс в диффузационной модели определяется в (20.3). Здесь рассмотрим предел непрерывного времени и преобразуем его в SDE. Сначала введем непрерывно меняющуюся дисперсионную функцию $\beta(t)$ такую, что $\beta_t = \beta(t)\Delta t$. Разложив по Тейлору квадратный корень в первом члене правой части (20.3), докажите, что бесконечно малое обновление можно записать в виде

$$d\mathbf{z} = -\frac{1}{2}\beta(t)\mathbf{z}dt + \sqrt{\beta(t)}d\mathbf{v}. \quad (20.70)$$

Из этого следует, что это частный случай общего SDE из (20.55).

- 20.20** (*) Используя (20.58) для замены $\nabla_{\mathbf{x}} \ln p(\mathbf{c} | \mathbf{x})$, докажите, что функция оценки в (20.59) может быть записана в виде (20.60).

Приложение А

Линейная алгебра

В этом приложении собраны воедино полезные свойства и формулы для работы с матрицами и определителями. Это не вводное учебное пособие, и предполагается, что читатель уже знаком с основами линейной алгебры. Для некоторых результатов указаны способы их доказательства, в то время как в более сложных случаях заинтересованный читатель может обратиться к стандартным учебникам по этому предмету. Во всех случаях подразумевается, что инверсии имеют место, а размерность матриц соответствует корректности формул. Всестороннее обсуждение линейной алгебры можно найти в работе (Golub and Van Loan, 1996), а обширное описание свойств матриц приведено в работе (Lutkepohl, 1996). Производные матрицы обсуждаются в работе (Magnus and Neudecker, 1999).

A.1. Матричные тождества

Матрица \mathbf{A} состоит из элементов A_{ij} , где i обозначает строки, а j – столбцы. Для обозначения матрицы тождества $N \times N$ (также называемой единичной матрицей) используется обозначение \mathbf{I}_N , а если нет неоднозначности в отношении размерности, то просто \mathbf{I} . Транспонирующая матрица \mathbf{A}^T имеет элементы $(\mathbf{A}^T)_{ij} = A_{ji}$. Из определения транспонирования следует, что

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T, \quad (\text{A.1})$$

в чем можно убедиться, если переписать индексы. Инверсия \mathbf{A} , обозначаемая \mathbf{A}^{-1} , удовлетворяет условию

$$\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}. \quad (\text{A.2})$$

Поскольку $\mathbf{ABB}^{-1}\mathbf{A}^{-1} = \mathbf{I}$, получается, что

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}. \quad (\text{A.3})$$

Также имеет место

$$(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T, \quad (\text{A.4})$$

что легко доказать, произведя транспонирование (A.2) и применив (A.1).

Полезным тождеством, связанным с инверсиями матриц, является следующее:

$$(\mathbf{P}^{-1} + \mathbf{B}^T \mathbf{R}^{-1} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{R}^{-1} = \mathbf{P} \mathbf{B}^T (\mathbf{B} \mathbf{P} \mathbf{B}^T + \mathbf{R})^{-1}, \quad (\text{A.5})$$

что легко проверить, умножив обе стороны направо на $(\mathbf{B} \mathbf{P} \mathbf{B}^T + \mathbf{R})$. Предположим, что \mathbf{P} имеет размерность $N \times N$, а \mathbf{R} – размерность $M \times M$, так что у \mathbf{B} получается $M \times N$. Тогда при $M \ll N$ оценивать правую часть (A.5) будет гораздо легче, чем левую. Иногда встречается особый случай:

$$(\mathbf{I} + \mathbf{AB})^{-1} \mathbf{A} = \mathbf{A} (\mathbf{I} + \mathbf{BA})^{-1}. \quad (\text{A.6})$$

Еще одно полезное тождество, связанное с инверсиями:

$$(\mathbf{A} + \mathbf{BD}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} + \mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1}, \quad (\text{A.7})$$

которое называют *тождеством Вудбери (Woodbury identity)*. Его можно проверить, умножив обе стороны на $(\mathbf{A} + \mathbf{BD}^{-1}\mathbf{C})$. Такое тождество полезно, например, когда \mathbf{A} велика и является диагональной, а значит, ее легко инвертировать, и когда \mathbf{B} имеет много строк, но мало столбцов (и наоборот для \mathbf{C}), так что правую часть оценивать гораздо легче, чем левую.

Набор векторов $\{\mathbf{a}_1, \dots, \mathbf{a}_N\}$ называется *линейно независимым (linearly independent)*, если соотношение $\sum_n \alpha_n \mathbf{a}_n = 0$ выполняется только при условии, что все $\alpha_n = 0$. Это означает, что ни один из векторов не может быть выражен в виде линейной комбинации остальных. Ранг матрицы – это максимальное количество линейно независимых строк (или, что эквивалентно, максимальное количество линейно независимых столбцов).

A.2. Следы и определители

Квадратные матрицы имеют следы и определители. След $\text{Tr}(\mathbf{A})$ матрицы \mathbf{A} определяется как сумма элементов на ведущей диагонали. Выписав индексы, можно увидеть, что

$$\text{Tr}(\mathbf{AB}) = \text{Tr}(\mathbf{BA}). \quad (\text{A.8})$$

Применив эту формулу несколько раз к произведению трех матриц, можно убедиться, что

$$\text{Tr}(\mathbf{ABC}) = \text{Tr}(\mathbf{CAB}) = \text{Tr}(\mathbf{BCA}). \quad (\text{A.9})$$

Это называется *циклическим (cyclic)* свойством оператора следа. Оно однозначно распространяется на произведение любого числа матриц. Определитель $|\mathbf{A}|$ матрицы \mathbf{A} размером $N \times N$ определяется как

$$|\mathbf{A}| = \sum (\pm 1) A_{1i_1} A_{2i_2} \cdots A_{Ni_N}, \quad (\text{A.10})$$

где сумма берется по всем произведениям, состоящим ровно из одного элемента в каждой строке и одного элемента в каждом столбце, с коэффициентом +1 или -1 в зависимости от того, является ли перестановка $i_1 i_2 \dots i_N$ четной или нечетной соответственно. Обратите внимание, что $|I| = 1$ и что определитель диагональной матрицы описывается произведением элементов на ведущей диагонали. Получается, что для матрицы 2×2 определитель имеет вид:

$$|A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}. \quad (A.11)$$

Определитель произведения двух матриц имеет вид:

$$|AB| = |A| |B|, \quad (A.12)$$

как следует из (A.10). Кроме того, определитель обратной матрицы имеет вид:

$$|A^{-1}| = \frac{1}{|A|}, \quad (A.13)$$

что можно доказать, взяв определитель из (A.2) и применив (A.12).

Если A и B – это матрицы размера $N \times M$, то

$$|I_N + AB^T| = |I_M + A^T B|. \quad (A.14)$$

Полезным частным случаем является

$$|I_N + ab^T| = 1 + a^T b, \quad (A.15)$$

где a и b – это N -мерные векторы-столбцы.

A.3. Производные матрицы

Иногда необходимо определить производные векторов и матриц по скалярам. Производная вектора a по скаляру x – это вектор, компоненты которого определяются как

$$\left(\frac{\partial a}{\partial x} \right)_i = \frac{\partial a_i}{\partial x} \quad (A.16)$$

с аналогичным определением для производной матрицы. Производные по векторам и матрицам также могут быть определены, как, например,

$$\left(\frac{\partial x}{\partial a} \right)_i = \frac{\partial x}{\partial a_i} \quad (A.17)$$

и аналогично

$$\left(\frac{\partial \mathbf{a}}{\partial \mathbf{b}} \right)_{ij} = \frac{\partial a_i}{\partial b_j}. \quad (\text{A.18})$$

Следующее легко доказать, выписав компоненты:

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{x}^T \mathbf{a}) = \frac{\partial}{\partial \mathbf{x}} (\mathbf{a}^T \mathbf{x}) = \mathbf{a}. \quad (\text{A.19})$$

Аналогичным образом

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{AB}) = \frac{\partial \mathbf{A}}{\partial \mathbf{x}} \mathbf{B} + \mathbf{A} \frac{\partial \mathbf{B}}{\partial \mathbf{x}}. \quad (\text{A.20})$$

Производная от обратной матрицы может быть выражена как

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{A}^{-1}) = -\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \mathbf{x}} \mathbf{A}^{-1}, \quad (\text{A.21})$$

что можно доказать, продифференцировав уравнение $\mathbf{A}^{-1} \mathbf{A} = \mathbf{I}$ по (A.20), а затем умножив вправо на \mathbf{A}^{-1} . Также

$$\frac{\partial}{\partial \mathbf{x}} \ln |\mathbf{A}^{-1}| = \text{Tr} \left(\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \mathbf{x}} \right), \quad (\text{A.22})$$

что будет доказано позже. Если выбрать x в качестве одного из элементов \mathbf{A} , то получится

$$\frac{\partial}{\partial A_{ij}} \text{Tr}(\mathbf{AB}) = B_{ji}, \quad (\text{A.23})$$

что очевидно из записи матриц с использованием индексных обозначений. Более компактно этот результат можно записать в виде

$$\frac{\partial}{\partial \mathbf{A}} \text{Tr}(\mathbf{AB}) = \mathbf{B}^T. \quad (\text{A.24})$$

В этой формулировке получаются следующие свойства:

$$\frac{\partial}{\partial \mathbf{A}} \text{Tr}(\mathbf{A}^T \mathbf{B}) = \mathbf{B}, \quad (\text{A.25})$$

$$\frac{\partial}{\partial \mathbf{A}} \text{Tr}(\mathbf{A}) = \mathbf{I}, \quad (\text{A.26})$$

$$\frac{\partial}{\partial \mathbf{A}} \text{Tr}(\mathbf{ABA}^T) = \mathbf{A}(\mathbf{B} + \mathbf{B}^T), \quad (\text{A.27})$$

что вновь можно доказать, выписав индексы матриц. Также имеется

$$\frac{\partial}{\partial \mathbf{A}} \ln|\mathbf{A}| = (\mathbf{A}^{-1})^T, \quad (\text{A.28})$$

что следует из (A.22) и (A.24).

A.4. Собственные векторы

Для квадратной матрицы \mathbf{A} размером $M \times M$ уравнение собственных векторов определяется как

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i \quad (\text{A.29})$$

для $i = 1, \dots, M$, где \mathbf{u}_i – это *собственный вектор* (*eigenvector*), а λ_i – соответствующее *собственное значение* (*eigenvalue*). Это можно рассматривать в виде набора из M одновременных однородных линейных уравнений, и условие для решения сводится к

$$|\mathbf{A} - \lambda_i \mathbf{I}| = 0, \quad (\text{A.30})$$

что называют *характеристическим уравнением* (*characteristic equation*). Поскольку это многочлен порядка M по λ , он должен иметь M решений (хотя они не обязательно должны быть разными). Ранг \mathbf{A} равен числу ненулевых собственных значений.

Особый интерес представляют симметричные матрицы, которые применяются в качестве ковариационных матриц, ядерных матриц и матриц Гессе. Симметричные матрицы обладают свойством $\mathbf{A}_{ij} = \mathbf{A}_{ji}$, или аналогичным образом $\mathbf{A}^T = \mathbf{A}$. Обратная симметричная матрица также является симметричной, в чем можно убедиться, взяв транспонирование $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ и используя $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ вместе с симметрией \mathbf{I} .

В общем случае собственные значения матрицы являются комплексными числами, но для симметричных матриц собственные значения λ являются вещественными. В этом можно убедиться, если сначала умножить (A.29) влево на $(\mathbf{u}_i^*)^T$, где $*$ обозначает комплексную сопряженную величину, чтобы получить

$$(\mathbf{u}_i^*)^T \mathbf{A} \mathbf{u}_i = \lambda_i (\mathbf{u}_i^*)^T \mathbf{u}_i. \quad (\text{A.31})$$

Далее возьмем комплексную сопряженную величину из (A.29) и умножим влево на \mathbf{u}_i^T , чтобы получить

$$\mathbf{u}_i^T \mathbf{A} \mathbf{u}_i^* = \lambda_i^* \mathbf{u}_i^T \mathbf{u}_i^*, \quad (\text{A.32})$$

где используется $\mathbf{A}^* = \mathbf{A}$, поскольку здесь рассматриваются только вещественные матрицы \mathbf{A} . Взяв транспонирование второго из этих уравнений

и используя свойство $\mathbf{A}^T = \mathbf{A}$, получаем, что левые стороны двух уравнений равны, а значит, $\lambda_i^* = \lambda_i$, и, следовательно, λ_i должна быть вещественной.

Собственные векторы \mathbf{u}_i вещественной симметричной матрицы могут быть выбраны ортогональными (т. е. ортогональными и единичной длины) так, что

$$\mathbf{u}_i^T \mathbf{u}_j = I_{ij}, \quad (\text{A.33})$$

где I_{ij} – это элементы матрицы тождества \mathbf{I} . Чтобы доказать это, сначала умножим (A.29) влево на \mathbf{u}_j^T , чтобы получить

$$\mathbf{u}_j^T \mathbf{A} \mathbf{u}_i = \lambda_i \mathbf{u}_j^T \mathbf{u}_i, \quad (\text{A.34})$$

и, поменяв индексы местами, получим

$$\mathbf{u}_i^T \mathbf{A} \mathbf{u}_j = \lambda_j \mathbf{u}_i^T \mathbf{u}_j. \quad (\text{A.35})$$

Теперь возьмем транспонирование второго уравнения и воспользуемся свойством симметрии $\mathbf{A}^T = \mathbf{A}$, а затем вычтем эти два уравнения, чтобы получить

$$(\lambda_i - \lambda_j) \mathbf{u}_i^T \mathbf{u}_j = 0. \quad (\text{A.36})$$

Следовательно, для $\lambda_i \neq \lambda_j$ справедливо $\mathbf{u}_i^T \mathbf{u}_j = 0$, так что \mathbf{u}_i и \mathbf{u}_j ортогональны. Если два собственных значения равны, то любая линейная комбинация $a\mathbf{u}_i + b\mathbf{u}_j$ также является собственным вектором с тем же собственным значением, поэтому можно произвольно выбрать одну линейную комбинацию, а затем выбрать вторую, ортогональную первой (можно доказать, что вырожденные собственные векторы никогда не являются линейно зависимыми). Следовательно, собственные векторы могут быть выбраны ортогональными и с помощью нормализации могут быть приведены к единичной длине. Поскольку существует M собственных значений, соответствующие M ортогональных собственных векторов образуют полный набор, и поэтому любой M -мерный вектор может быть выражен как линейная комбинация собственных векторов.

Собственные векторы \mathbf{u}_i можно рассматривать как столбцы матрицы \mathbf{U} размером $M \times M$, которая в силу ортонормированности удовлетворяет условию:

$$\mathbf{U}^T \mathbf{U} = \mathbf{I}. \quad (\text{A.37})$$

Такая матрица называется *ортогональной* (*orthogonal*). Примечательно, что строки этой матрицы также ортогональны, так что $\mathbf{U} \mathbf{U}^T = \mathbf{I}$. Чтобы доказать это, обратите внимание, что из (A.37) следует $\mathbf{U}^T \mathbf{U} \mathbf{U}^{-1} = \mathbf{U}^{-1} = \mathbf{U}^T$, и поэтому $\mathbf{U} \mathbf{U}^{-1} = \mathbf{U} \mathbf{U}^T = \mathbf{I}$. Исходя из (A.12), также следует, что $|\mathbf{U}| = 1$.

Уравнение собственных векторов (A.29) может быть выражено через \mathbf{U} в виде

$$\mathbf{A} \mathbf{U} = \mathbf{U} \Lambda, \quad (\text{A.38})$$

где Λ – это диагональная матрица $M \times M$, диагональные элементы которой задаются собственными значениями λ_i .

Если рассматривать вектор-столбец \mathbf{x} , который преобразуется ортогональной матрицей \mathbf{U} и дает новый вектор

$$\tilde{\mathbf{x}} = \mathbf{U}\mathbf{x}, \quad (\text{A.39})$$

то длина вектора сохраняется, так как

$$\tilde{\mathbf{x}}^T \tilde{\mathbf{x}} = \mathbf{x}^T \mathbf{U}^T \mathbf{U} \mathbf{x} = \mathbf{x}^T \mathbf{x}, \quad (\text{A.40})$$

и точно так же сохраняется угол между любыми двумя такими векторами, потому что

$$\tilde{\mathbf{x}}^T \tilde{\mathbf{y}} = \mathbf{x}^T \mathbf{U}^T \mathbf{U} \mathbf{y} = \mathbf{x}^T \mathbf{y}. \quad (\text{A.41})$$

Таким образом, умножение на \mathbf{U} можно интерпретировать как жесткий поворот системы координат.

Из (A.38) следует, что

$$\mathbf{U}^T \mathbf{A} \mathbf{U} = \Lambda, \quad (\text{A.42})$$

и поскольку Λ – это диагональная матрица, то можно сказать, что матрица \mathbf{A} диагонализована (*diagonalized*) матрицей \mathbf{U} . Если умножить на \mathbf{U} влево и умножить на \mathbf{U}^T вправо, то получается

$$\mathbf{A} = \mathbf{U} \Lambda \mathbf{U}^T. \quad (\text{A.43})$$

Взяв инверсию этого уравнения и используя (A.3) вместе с $\mathbf{U}^{-1} = \mathbf{U}^T$, получаем

$$\mathbf{A}^{-1} = \mathbf{U} \Lambda^{-1} \mathbf{U}^T. \quad (\text{A.44})$$

Эти два последних уравнения также можно записать в виде

$$\mathbf{A} = \sum_{i=1}^M \lambda_i \mathbf{u}_i \mathbf{u}_i^T, \quad (\text{A.45})$$

$$\mathbf{A}^{-1} = \sum_{i=1}^M \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T. \quad (\text{A.46})$$

Если взять определитель из (A.43) и воспользоваться (A.12), то получится

$$|\mathbf{A}| = \prod_{i=1}^M \lambda_i. \quad (\text{A.47})$$

Аналогично, взяв след из (A.43) и используя циклическое свойство (A.8) оператора следа вместе с $\mathbf{U}^T \mathbf{U} = \mathbf{I}$, получим

$$\text{Tr}(\mathbf{A}) = \sum_{i=1}^M \lambda_i. \quad (\text{A.48})$$

Оставим читателю возможность проверить (A.22), используя результаты (A.33), (A.45), (A.46) и (A.47).

Матрица A считается *положительно определенной* (*positive definite*), что обозначается как $A > 0$, если $w^T A w > 0$ для всех ненулевых значений вектора w . Точно так же положительно определенная матрица имеет $\lambda_i > 0$ для всех своих собственных значений (в чем можно убедиться, подставляя w в каждый из собственных векторов по очереди и отмечая, что произвольный вектор может быть разложен как линейная комбинация собственных векторов). Обратите внимание, что наличие всех положительных элементов не обязательно означает, что матрица положительно определенная. Например, матрица

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad (\text{A.49})$$

имеет собственные значения $\lambda_1 \approx 5,37$ и $\lambda_2 \approx -0,37$. Матрица считается *положительно полуопределенной* (*positive semidefinite*), если при всех значениях w справедливо $w^T A w \geq 0$, что обозначается как $A \succeq 0$ и соответствует $\lambda_i \geq 0$.

Условное число матрицы определяется как

$$\text{CN} = \left(\frac{\lambda_{\max}}{\lambda_{\min}} \right)^{1/2}, \quad (\text{A.50})$$

где λ_{\max} – это наибольшее собственное значение, а λ_{\min} – это наименьшее собственное значение.

Приложение В

Вариационное исчисление

Функцию $y(x)$ можно представить в качестве оператора, который для любого входного значения x возвращает выходное значение y . Таким же образом можно определить функционал (*functional*) $F[y]$, который в качестве функционального оператора получает функцию $y(x)$ и возвращает выходное значение F . Примером функционала может служить длина кривой, проведенной на двумерной плоскости, путь которой определяется с помощью функции. В контексте машинного обучения широко используется функция энтропии $H[x]$ для непрерывной переменной x , поскольку для любого выбора функции распределения плотности вероятности $p(x)$ она возвращает скалярное значение энтропии x под этой плотностью. Получается, что энтропия $p(x)$ с тем же успехом может быть записана как $H[p]$.

Обычная задача в обычном исчислении – это поиск значения x , которое максимизирует (или минимизирует) функцию $y(x)$. Точно так же в вариационном исчислении требуется найти функцию $y(x)$, которая максимизирует (или минимизирует) функционал $F[y]$. То есть среди всех возможных функций $y(x)$ необходимо найти ту, для которой функционал $F[y]$ является максимальным (или минимальным). Вариационное исчисление можно использовать, например, для доказательства того, что кратчайший путь между двумя точками – это прямая линия или что максимум распределения энтропии – это распределение Гаусса.

Если допустить отсутствие знаний правил обычного исчисления, то можно попробовать оценить обычную производную dy/dx , внеся небольшое изменение в переменную x и затем разложив ее по степеням ϵ , так что

$$y(x + \epsilon) = y(x) + \frac{dy}{dx} \epsilon + \mathcal{O}(\epsilon^2), \quad (\text{B.1})$$

и, наконец, взяв предел $\epsilon \rightarrow 0$. Аналогичным образом для функции нескольких переменных $y(x_1, \dots, x_D)$ соответствующие частные производные определяются как

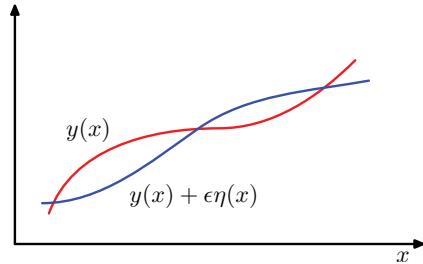
$$y(x_1 + \epsilon_1, \dots, x_D + \epsilon_D) = y(x_1, \dots, x_D) + \sum_{i=1}^D \frac{dy}{dx_i} \epsilon_i + \mathcal{O}(\epsilon^2). \quad (\text{B.2})$$

Подобное определение функциональной производной вытекает из определения того, насколько изменяется функционал $F[y]$ при небольшом из-

менении $\epsilon\eta(x)$ функции $y(x)$, где $\eta(x)$ – это произвольная функция от x , как показано на рис. В.1. Обозначим функциональную производную $F[y]$ по $y(x)$ через $\delta F/\delta y(x)$ и определим ее следующим соотношением:

$$F[y(x) + \epsilon\eta(x)] = F[y(x)] + \epsilon \int \frac{\delta F}{\delta y(x)} \eta(x) dx + \mathcal{O}(\epsilon^2). \quad (\text{B.3})$$

РИС. В.1 Функциональную производную можно определить исходя из изменения значения функционала $F[y]$ при изменении функции $y(x)$ по $y(x) + \eta(x)$, где $\eta(x)$ – это произвольная функция от x



Это можно рассматривать в качестве естественного расширения (B.2), в котором $F[y]$ теперь зависит от непрерывного набора переменных, а именно от значений y во всех точках x . Необходимость того, чтобы функционал был стационарным относительно малых изменений функции $y(x)$, приводит к

$$\int \frac{\delta F}{\delta y(x)} \eta(x) dx = 0. \quad (\text{B.4})$$

Поскольку это должно выполняться для произвольного выбора $\eta(x)$, получается, что функциональная производная должна исчезать. Чтобы убедиться в этом, предположим, что возмущение $\eta(x)$ равно нулю везде, кроме окрестности точки \hat{x} , и тогда функциональная производная должна равняться нулю при $x = \hat{x}$. Но поскольку это должно быть верно при любом выборе \hat{x} , функциональная производная должна исчезать при всех значениях x . Рассмотрим функционал, определяемый интегралом по функции $G(y, y', x)$, которая зависит как от $y(x)$, так и от своей производной $y'(x)$, а также имеет прямую зависимость от x :

$$F[y] = \int G(y(x), y'(x), x) dx, \quad (\text{B.5})$$

где предполагается, что значение $y(x)$ фиксировано на границе области интегрирования (которая может находиться на бесконечности). Если теперь рассматривать вариации функции $y(x)$, то получится

$$F[y(x) + \epsilon\eta(x)] = F[y(x)] + \epsilon \int \left\{ \frac{\partial G}{\partial y} \eta(x) + \frac{\partial G}{\partial y'} \eta'(x) \right\} dx + \mathcal{O}(\epsilon^2). \quad (\text{B.6})$$

Теперь необходимо привести это к виду (B.3). Для этого нужно проинтегрировать второй член по частям и обратить внимание на то, что $\eta(x)$ должна исчезать на границе интеграла (поскольку $y(x)$ фиксирована на границе). Это дает

$$F[y(x) + \epsilon\eta(x)] = F[y(x)] + \epsilon \int \left\{ \frac{\partial G}{\partial y} - \frac{d}{dx} \left(\frac{\partial G}{\partial y'} \right) \right\} \eta(x) dx + \mathcal{O}(\epsilon^2), \quad (\text{B.7})$$

из которого можно вычислить функциональную производную путем сравнения с (B.3). Условие, согласно которому функциональная производная исчезает, приводит к

$$\frac{\partial G}{\partial y} - \frac{d}{dx} \left(\frac{\partial G}{\partial y'} \right) = 0, \quad (\text{B.8})$$

что называют *уравнениями Эйлера–Лагранжа (Euler–Lagrange equations)*. Например, если

$$G = y(x)^2 + (y'(x))^2, \quad (\text{B.9})$$

то уравнения Эйлера–Лагранжа имеют вид:

$$y(x) - \frac{d^2 y}{dx^2} = 0. \quad (\text{B.10})$$

Это дифференциальное уравнение второго порядка можно решить для $y(x)$ с помощью граничных условий по $y(x)$.

Зачастую рассматриваются функционалы по интегралам, подынтегральные значения которых имеют вид $G(y, x)$ и не зависят от производных $y(x)$. В этом случае стационарность просто подразумевает, чтобы $\partial G / \partial y(x) = 0$ для всех значений x . Если оптимизировать функционал относительно вероятностного распределения, то необходимо сохранить ограничение нормализации для вероятностей. Чаще всего это удобнее всего сделать с помощью множителя Лагранжа (см. приложение C), который позволяет провести оптимизацию без каких-либо ограничений.

Применение приведенных выше результатов к многомерной переменной x не представляет трудностей. Более полное обсуждение вариационного исчисления см. в (Sagan, 1969).

Приложение С

Множители Лагранжа

Множители Лагранжа (Lagrange multipliers), также иногда называемые *неопределенными множителями (undefined multipliers)*, используются для нахождения стационарных точек функции нескольких переменных при одном или нескольких ограничениях.

Рассмотрим задачу нахождения максимума функции $f(x_1, x_2)$ при наличии ограничения, связывающего x_1 и x_2 , которое запишем в виде

$$g(x_1, x_2) = 0. \quad (\text{C.1})$$

Один из возможных способов предполагает решить уравнение с ограничением (C.1) и, таким образом, выразить x_2 как функцию x_1 в виде $x_2 = h(x_1)$. Затем это можно подставить в $f(x_1, x_2)$ и получить функцию только от x_1 в виде $f(x_1, h(x_1))$. Максимум по отношению к x_1 можно найти обычным способом дифференцирования, в результате чего будет получено стационарное значение x_1^* , а соответствующее значение x_2 будет задано $x_2^* = h(x_1^*)$.

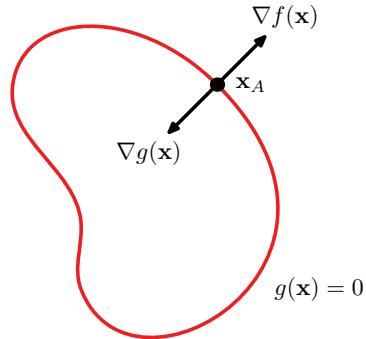
Одна из проблем этого подхода заключается в том, что аналитическое решение уравнения ограничений, позволяющее выразить x_2 в виде явной функции от x_1 , может быть затруднительным. Кроме того, при таком подходе x_1 и x_2 рассматриваются в различных аспектах, что нарушает естественную симметрию между этими переменными.

Более эффективный и зачастую более простой способ заключается введении параметра λ , называемого множителем Лагранжа. Поясним эту технику с геометрической точки зрения. Рассмотрим D -мерную переменную \mathbf{x} с компонентами x_1, \dots, x_D . Тогда уравнение ограничения $g(\mathbf{x}) = 0$ представляет собой $(D - 1)$ -мерную поверхность в пространстве \mathbf{x} , как показано на рис. C.1. Если \mathbf{x} имеет размерность D , то ограничение $g(\mathbf{x}) = 0$ соответствует подпространству размерности $D - 1$, как показано красной кривой. Задача решается путем оптимизации функции Лагранжа $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$.

Сначала нужно отметить, что в любой точке поверхности ограничений градиент $\nabla g(\mathbf{x})$ функции ограничений ортогонален поверхности. Чтобы убедиться в этом, рассмотрим точку \mathbf{x} , лежащую на ограничивающей поверхности, и близлежащую точку $\mathbf{x} + \boldsymbol{\epsilon}$, которая также лежит на этой поверхности. Если выполнить разложение Тейлора около \mathbf{x} , то получится

$$g(\mathbf{x} + \boldsymbol{\epsilon}) \simeq g(\mathbf{x}) + \boldsymbol{\epsilon}^T \nabla g(\mathbf{x}). \quad (\text{C.2})$$

РИС. С.1 Геометрическое представление метода множителей Лагранжа, в котором требуется максимизировать функцию $f(\mathbf{x})$ при условии ограничения $g(\mathbf{x}) = 0$



Поскольку оба \mathbf{x} и $\mathbf{x} + \boldsymbol{\epsilon}$ лежат на ограничивающей поверхности, получается $g(\mathbf{x}) = g(\mathbf{x} + \boldsymbol{\epsilon})$ и, следовательно, $\boldsymbol{\epsilon}^T \nabla g(\mathbf{x}) \simeq 0$. В пределе $\|\boldsymbol{\epsilon}\| \rightarrow 0$ получается $\boldsymbol{\epsilon}^T \nabla g(\mathbf{x}) = 0$, и поскольку $\boldsymbol{\epsilon}$ в этом случае параллелен ограничивающей поверхности $g(\mathbf{x}) = 0$, значит, что вектор ∇g является нормалью к поверхности.

Далее необходимо найти точку \mathbf{x}^* на поверхности ограничений, где $f(\mathbf{x})$ максимальна. Такая точка должна обладать тем свойством, что вектор $\nabla f(\mathbf{x})$ также ортогонален поверхности ограничений, как показано на рис. С.1, поскольку в противном случае значение $f(\mathbf{x})$ пришлось бы увеличить, перемещаясь на небольшое расстояние вдоль поверхности ограничений. Получается, что ∇f и ∇g – это параллельные (или антипараллельные) векторы, и поэтому должен существовать параметр λ , при котором

$$\nabla f + \lambda \nabla g = 0, \quad (\text{C.3})$$

где $\lambda \neq 0$ называют **множителем Лагранжа** (*Lagrange multiplier*). Обратите внимание, что λ может иметь любой знак.

На этом этапе будет удобно ввести функцию **Лагранжа** (*Lagrangian function*), определяемую как

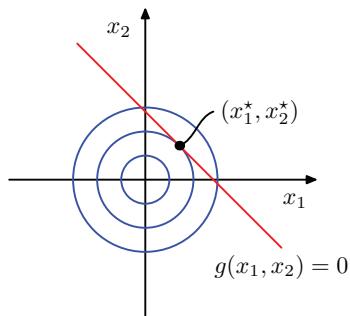
$$L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x}). \quad (\text{C.4})$$

Ограниченое условие стационарности (C.3) можно определить, задав $\nabla_{\mathbf{x}} L = 0$. Более того, условие $\partial L / \partial \lambda = 0$ приводит к уравнению ограничения $g(\mathbf{x}) = 0$. Получается, что для нахождения максимума функции $f(\mathbf{x})$ при ограничении $g(\mathbf{x}) = 0$ необходимо определить функцию Лагранжа, заданную в (C.4), а затем найти стационарную точку $L(\mathbf{x}, \lambda)$ относительно \mathbf{x} и λ . Для D -мерного вектора \mathbf{x} это дает $D + 1$ уравнений, которые определяют и стационарную точку \mathbf{x}^* , и значение λ . Если интерес представляет только \mathbf{x}^* , то можно исключить из уравнений стационарности без необходимости искать его значение (отсюда термин «неопределенный множитель»).

В качестве простого примера предположим, что требуется найти стационарную точку функции $f(x_1, x_2) = 1 - x_1^2 - x_2^2$ при условии ограничения $g(x_1, x_2) = x_1 + x_2 - 1 = 0$, как показано на рис. С.2. Соответствующая функция Лагранжа имеет вид:

$$L(\mathbf{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1). \quad (\text{C.5})$$

РИС. С.2 Простой пример использования множителей Лагранжа, где целью является максимизация $f(x_1, x_2) = 1 - x_1^2 - x_2^2$ при ограничении $g(x_1, x_2) = 0$, где $g(x_1, x_2) = x_1 + x_2 - 1$. Кружки показывают контуры функции $f(x_1, x_2)$, а диагональная линия – это поверхность ограничений $g(x_1, x_2) = 0$



Условия стационарности этой функции Лагранжа относительно x_1 , x_2 и λ дают следующие связные уравнения:

$$-2x_1 + \lambda = 0, \quad (C.6)$$

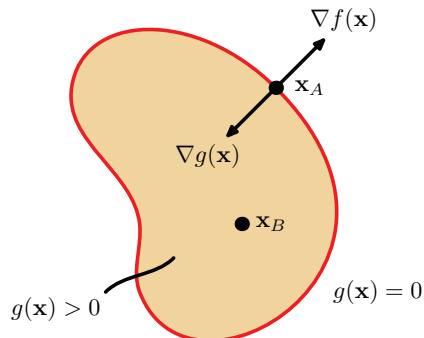
$$-2x_2 + \lambda = 0, \quad (C.7)$$

$$x_1 + x_2 - 1 = 0. \quad (C.8)$$

Решение этих уравнений позволяет определить стационарную точку как $(x_1^*, x_2^*) = (1/2, 1/2)$, а соответствующее значение множителя Лагранжа равно $\lambda = 1$.

До сих пор рассматривалась задача максимизации функции при *ограничении равенства* вида $g(\mathbf{x}) = 0$. Теперь рассмотрим задачу максимизации $f(\mathbf{x})$ при *ограничении неравенства* вида $g(\mathbf{x}) \geq 0$, как показано на рис. С.3.

РИС. С.3 Иллюстрация задачи максимизации $f(\mathbf{x})$ при ограничении в виде неравенства $g(\mathbf{x}) > 0$



Теперь возможны два вида решений. Если ограниченная стационарная точка лежит в области $g(\mathbf{x}) > 0$, тогда ограничение *неактивно*, если же она лежит на границе $g(\mathbf{x}) = 0$, то в этом случае ограничение считается *активным*. В первом случае функция $g(\mathbf{x})$ не играет никакой роли, и поэтому стационарное условие просто $\nabla f(\mathbf{x}) = 0$. Это также соответствует стационарной точке функции Лагранжа (С.4), но на этот раз с $\lambda = 0$. Последний случай, когда решение лежит на границе, аналогичен рассмотренному ранее ограничению равенства и соответствует стационарной точке функции Лагранжа (С.4).

с $\lambda \neq 0$. Однако теперь знак множителя Лагранжа имеет решающее значение, поскольку функция $f(\mathbf{x})$ достигает максимума только в том случае, если ее градиент направлен в сторону от области $g(\mathbf{x}) > 0$, как показано на рис. С.3. Поэтому для некоторого значения $\lambda > 0$ имеет место $\nabla f(\mathbf{x}) = -\lambda \nabla g(\mathbf{x})$.

Для любого из этих двух случаев $\lambda g(\mathbf{x}) = 0$. Таким образом, решение задачи максимизации $f(\mathbf{x})$ при условии $g(\mathbf{x}) \geq 0$ получается путем оптимизации функции Лагранжа (С.4) относительно \mathbf{x} и λ при соблюдении условий:

$$g(\mathbf{x}) \geq 0, \quad (\text{C.9})$$

$$\lambda \geq 0, \quad (\text{C.10})$$

$$\lambda g(\mathbf{x}) = 0. \quad (\text{C.11})$$

Они известны как условия *Каруша–Куна–Таккера* (*Karush–Kuhn–Tucker conditions, KKT*) (Karush, 1939; Kuhn and Tucker, 1951).

Обратите внимание, что если необходимо минимизировать (а не максимизировать) функцию $f(\mathbf{x})$ при ограничении в виде неравенства $g(\mathbf{x}) \geq 0$, то нужно минимизировать функцию Лагранжа $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$ относительно \mathbf{x} опять же при условии $\lambda \geq 0$. Наконец, технику множителей Лагранжа легко расширить на случаи с несколькими ограничениями в виде равенств и неравенств. Предположим, что необходимо максимизировать $f(\mathbf{x})$ при условии $g_j(\mathbf{x}) = 0$ для $j = 1, \dots, J$ и при условии $h_k(\mathbf{x}) \geq 0$ для $k = 1, \dots, K$. Здесь вводятся множители Лагранжа $\{\lambda_j\}$ и $\{\mu_k\}$, и оптимизируется функция Лагранжа

$$L(\mathbf{x}, \{\lambda_j\}, \{\mu_k\}) = f(\mathbf{x}) + \sum_{j=1}^J \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^K \mu_k h_k(\mathbf{x}) \quad (\text{C.12})$$

при соблюдении $\mu_k \geq 0$ и $\mu_k h_k(\mathbf{x}) = 0$ для $k = 1, \dots, K$. Расширения для ограниченных функциональных производных (см. приложение В) также просты. Более подробное обсуждение техники множителей Лагранжа см. в (Nocedal and Wright, 1999).

Список литературы

- Abramowitz, M., and I. A. Stegun. 1965. *Handbook of Mathematical Functions*. Dover.
- Adler, S. L. 1981. Over-relaxation method for the Monte Carlo evaluation of the partition function for multiquadratic actions. *Physical Review D* 23: 2901–2904.
- Aghajanyan, Armen, Bernie Huang, Candace Ross, Vladimir Karpukhin, Hu Xu, Naman Goyal, Dmytro Okhonko, et al. 2022. CM3: A Causal Masked Multimodal Model of the Internet. Technical report. arXiv: 2201.07520.
- Aghajanyan, Armen, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. Technical report. arXiv: 2012.13255.
- Ahn, J. H., and J. H. Oh. 2003. A constrained EM algorithm for principal component analysis. *Neural Computation* 15 (1): 57–65.
- Alayrac, Jean-Baptiste, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, et al. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. Technical report. arXiv: 2204.14198.
- Amari, S., A. Cichocki, and H. H. Yang. 1996. A new learning algorithm for blind signal separation. In *Advances in Neural Information Processing Systems*, edited by D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, 8: 757–763. MIT Press.
- Anderson, J. A., and E. Rosenfeld. 1988. *Neurocomputing: Foundations of Research*. MIT Press.
- Anderson, T. W. 1963. Asymptotic Theory for Principal Component Analysis. *Annals of Mathematical Statistics* 34: 122–148.
- Arjovsky, M., S. Chintala, and L. Bottou. 2017. Wasserstein GAN. Technical report. arXiv: 1701.07875.
- Attias, H. 1999. Independent factor analysis. *Neural Computation* 11 (4): 803–851.
- Austin, Jacob, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured Denoising Diffusion Models in Discrete State-Spaces. In *Advances in Neural Information Processing Systems*, 34: 17981–17993.
- Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. Technical report. arXiv: 1607.06450.
- Bach, F. R., and M. I. Jordan. 2002. Kernel Independent Component Analysis. *Journal of Machine Learning Research* 3: 1–48.
- Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. 2015. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. Technical report. arXiv: 1511.00561.

- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. *Neural Machine Translation by Jointly Learning to Align and Translate*. Technical report. arXiv: 1409.0473.
- Baldi, P., and K. Hornik. 1989. Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks* 2 (1): 53–58.
- Baldazzi, David, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. 2017. *The Shattered Gradients Problem: If resnets are the answer, then what is the question?* Technical report. arXiv: 1702.08591.
- Bartholomew, D J. 1987. *Latent Variable Models and Factor Analysis*. Charles Griffin.
- Basilevsky, Alexander. 1994. *Statistical Factor Analysis and Related Methods: Theory and Applications*. Wiley.
- Bather, J. 2000. *Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions*. Wiley.
- Battaglia, Peter W., Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, et al. 2018. *Relational inductive biases, deep learning, and graph networks*. Technical report. arXiv: 1806.01261.
- Baydin, A. G., B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. 2018. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research* 18: 1–43.
- Becker, S., and Y. LeCun. 1989. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 Connectionist Models Summer School*, edited by D. Touretzky, G. E. Hinton, and T. J. Sejnowski, 29–37. Morgan Kaufmann.
- Belkin, Mikhail, Daniel Hsu, Siyuan Ma, and Soumik Mandal. 2019. Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences* 116 (32): 15849–15854.
- Bell, A. J., and T. J. Sejnowski. 1995. An information maximization approach to blind separation and blind deconvolution. *Neural Computation* 7 (6): 1129–1159.
- Bellman, R. 1961. *Adaptive Control Processes: A Guided Tour*. Princeton University Press.
- Bengio, Yoshua, Aaron Courville, and Pascal Vincent. 2012. *Representation Learning: A Review and New Perspectives*. Technical report. arXiv: 1206.5538.
- Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. 2013. *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*. Technical report. arXiv: 1308.3432.
- Berger, J. O. 1985. *Statistical Decision Theory and Bayesian Analysis*. Second. Springer.
- Bernardo, J. M., and A. F. M. Smith. 1994. *Bayesian Theory*. Wiley.
- Bishop, C. M. 1995a. Regularization and Complexity Control in Feed-forward Networks. In *Proceedings International Conference on Artificial Neural Networks ICANN'95*, edited by F. Foujelman-Soulie and P. Gallinari, 1: 141–148. EC2 et Cie.

- Bishop, Christopher M. 1992. Exact Calculation of the Hessian Matrix for the Multilayer Perceptron. *Neural Computation* 4 (4): 494–501.
- Bishop, Christopher M. 1994. Novelty Detection and Neural Network Validation. *IEE Proceedings: Vision, Image and Signal Processing* 141 (4): 217–222.
- Bishop, Christopher M. 1995b. *Neural Networks for Pattern Recognition*. Oxford University Press.
- Bishop, Christopher M. 1995c. Training with noise is equivalent to Tikhonov regularization. *Neural Computation* 7 (1): 108–116.
- Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer.
- Bommasani, Rishi, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, et al. 2021. *On the Opportunities and Risks of Foundation Models*. Technical report. arXiv: 2108.07258.
- Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings COMPSTAT 2010*, 177–186. Springer.
- Bourlard, H., and Y. Kamp. 1988. Autoassociation by multilayer perceptrons and singular value decomposition. *Biological Cybernetics* 59: 291–294.
- Breiman, L. 1996. Bagging predictors. *Machine Learning* 26: 123–140.
- Brinker, T. J., A. Hekler, A. H. Enk, C. Berking, S Haferkamp, A. Hauschild, M. Weichenthal, et al. 2019. Deep neural networks are superior to dermatologists in melanoma image classification. *European Journal of Cancer* 119: 11–17.
- Brock, Andrew, Jeff Donahue, and Karen Simonyan. 2018. Large-Scale GAN Training for High Fidelity Natural Image Synthesis. In *Proceedings of the International Conference Learning Representations (ICLR)*. ArXiv: 1809.11096.
- Bronstein, Michael M., Joan Bruna, Taco Cohen, and Petar Velickovic. 2021. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. Technical report. arXiv: 2104.13478.
- Bronstein, Michael M., Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going Beyond Euclidean Data. In *IEEE Signal Processing Magazine*, vol. 34. 4. IEEE, July.
- Broomhead, D. S., and D. Lowe. 1988. Multivariable functional interpolation and adaptive networks. *Complex Systems* 2: 321–355.
- Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. *Language Models are Few-Shot Learners*. Technical report. arXiv: 2005.14165.
- Bubeck, Sébastien, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, et al. 2023. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. Technical report. arXiv: 2303.12712.
- Cardoso, J-F. 1998. Blind signal separation: statistical principles. *Proceedings of the IEEE* 9 (10): 2009–2025.
- Caruana, R. 1997. Multitask learning. *Machine Learning* 28: 41–75.
- Casella, G., and R. L. Berger. 2002. *Statistical Inference*. Second. Duxbury.

- Chan, K., T. Lee, and T. J. Sejnowski. 2003. Variational Bayesian learning of ICA with missing data. *Neural Computation* 15 (8): 1991–2011.
- Chen, A. M., H. Lu, and R. Hecht-Nielsen. 1993. On the geometry of feedforward neural network error surfaces. *Neural Computation* 5 (6): 910–927.
- Chen, Mark, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative Pretraining From Pixels. *Proceedings of Machine Learning Research* 119: 1691–1703.
- Chen, R. T. Q., Rubanova Y., J. Bettencourt, and D. Duvenaud. 2018. Neural Ordinary Differential Equations. Technical report. arXiv: 1806.07366.
- Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. *A Simple Framework for Contrastive Learning of Visual Representations*. Technical report. arXiv: 2002.05709.
- Cho, Kyunghyun, Bart van Merriënboer, Çağlar Gülcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translations*. Technical report. arXiv: 1406.1078.
- Choudrey, R. A., and S. J. Roberts. 2003. Variational mixture of Bayesian independent component analyzers. *Neural Computation* 15 (1): 213–252.
- Christiano, Paul, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. Technical report. arXiv: 1706.03741.
- Collobert, R. 2004. Large Scale Machine Learning. PhD diss., Université Paris VI.
- Comon, P., C. Jutten, and J. Herault. 1991. Blind source separation, 2: problems statement. *Signal Processing* 24 (1): 11–20.
- Cover, T., and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* IT-11: 21–27.
- Cover, T. M., and J. A. Thomas. 1991. *Elements of Information Theory*. Wiley.
- Cox, R. T. 1946. Probability, frequency and reasonable expectation. *American Journal of Physics* 14 (1): 1–13.
- Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* 2: 304–314.
- Dawid, A. P. 1979. Conditional Independence in Statistical Theory (with discussion). *Journal of the Royal Statistical Society, Series B* 4: 1–31.
- Dawid, A. P. 1980. Conditional Independence for Statistical Operations. *Annals of Statistics* 8: 598–617.
- Deisenroth, M. P., A. A. Faisal, and C. S. Ong. 2020. *Mathematics for Machine Learning*. Cambridge University Press.
- Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B* 39 (1): 1–38.

- Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A largescale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. *BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding*. Technical report. arXiv: 1810.04805.
- Dhariwal, Prafulla, and Alex Nichol. 2021. *Diffusion Models Beat GANs on Image Synthesis*. Technical report. arXiv: 2105.05233.
- Dinh, Laurent, David Krueger, and Yoshua Bengio. 2014. *NICE: Non-linear Independent Components Estimation*. Technical report. arXiv: 1410.8516.
- Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. 2016. *Density estimation using Real NVP*. Technical report. arXiv: 1605.08803.
- Dodge, Samuel, and Lina Karam. 2017. *A Study and Comparison of Human and Deep Learning Recognition Performance Under Visual Distortions*. Technical report. arXiv: 1705.02498.
- Doersch, C. 2016. *Tutorial on Variational Autoencoders*. Technical report. arXiv: 1606.05908.
- Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, et al. 2020. *An Image is Worth 16 \times 16 Words: Transformers for Image Recognition at Scale*. Technical report. arXiv: 2010.11929.
- Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth. 1987. Hybrid Monte Carlo. *Physics Letters B* 195 (2): 216–222.
- Duchi, J., E. Hazan, and Y. Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12: 2121–2159.
- Duda, R. O., and P. E. Hart. 1973. *Pattern Classification and Scene Analysis*. Wiley.
- Duffter, Philipp, Martin Schmitt, and Hinrich Schütze. 2021. *Position Information in Transformers: An Overview*. Technical report. arXiv: 2102.11090.
- Dumoulin, Vincent, and Francesco Visin. 2016. *A guide to convolution arithmetic for deep learning*. Technical report. arXiv: 1603.07285.
- Elliott, R. J., L. Aggoun, and J. B. Moore. 1995. *Hidden Markov Models: Estimation and Control*. Springer.
- Esser, Patrick, Robin Rombach, and Björn Ommer. 2020. *Taming Transformers for High-Resolution Image Synthesis*. Technical report. arXiv: 2012.09841.
- Esteva, A., B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. 2017. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542: 115–118.
- Everitt, B. S. 1984. *An Introduction to Latent Variable Models*. Chapman / Hall.
- Eykholz, Kevin, Ivan Evtimov, Earlene Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-

- World Attacks on Deep Learning Visual Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27: 861–874.
- Feller, W. 1966. *An Introduction to Probability Theory and its Applications*. Second. Vol. 2. Wiley.
- Fletcher, R. 1987. *Practical Methods of Optimization*. Second. Wiley.
- Forsyth, D. A., and J. Ponce. 2003. *Computer Vision: A Modern Approach*. Prentice Hall.
- Freund, Y., and R. E. Schapire. 1996. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, edited by L. Saitta, 148–156. Morgan Kaufmann.
- Fukushima, K. 1980. Neocognitron: A Selforganizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics* 36: 193–202.
- Funahashi, K. 1989. On the approximate realization of continuous mappings by neural networks. *Neural Networks* 2 (3): 183–192.
- Fung, R., and K. C. Chang. 1990. Weighting and Integrating Evidence for Stochastic Simulation in Bayesian Networks. In *Uncertainty in Artificial Intelligence*, edited by P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, 5: 208–219. Elsevier.
- Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. 2015. *A Neural Algorithm of Artistic Style*. Technical report. arXiv: 1508.06576.
- Geman, S., and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE PAMI* 6 (1): 721–741.
- Gemmeke, Jort F., Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio Set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*. New Orleans, LA.
- Germain, Mathieu, Karol Gregor, Iain Murray, and Hugo Larochelle. 2015. *MADE: Masked Autoencoder for Distribution Estimation*. Technical report. arXiv: 1502.03509.
- Gilks, W. R. 1992. Derivative-free adaptive rejection sampling for Gibbs sampling. In *Bayesian Statistics*, edited by J. Bernardo, J. Berger, A. P. Dawid, and A. F. M. Smith, vol. 4. Oxford University Press.
- Gilks, W. R., N. G. Best, and K. K. C. Tan. 1995. Adaptive rejection Metropolis sampling. *Applied Statistics* 44: 455–472.
- Gilks, W. R., S. Richardson, and D. J. Spiegelhalter. 1996. *Markov Chain Monte Carlo in Practice*. Chapman / Hall.
- Gilks, W. R., and P. Wild. 1992. Adaptive rejection sampling for Gibbs sampling. *Applied Statistics* 41:337–348.

- Gilmer, Justin, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. *Neural Message Passing for Quantum Chemistry*. Technical report. arXiv: 1704.01212.
- Girshick, Ross B. 2015. *Fast R-CNN*. Technical report. arXiv: 1504.08083.
- Golub, G. H., and C. F. Van Loan. 1996. *Matrix Computations*. Third. John Hopkins University Press.
- Gong, Yuan, Yu-An Chung, and James R. Glass. 2021. *AST: Audio Spectrogram Transformer*. Technical report. arXiv: 2104.01778.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
- Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. *Generative Adversarial Networks*. Technical report. arXiv: 1406.2661.
- Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. 2014. *Explaining and Harnessing Adversarial Examples*. Technical report. arXiv: 1412.6572.
- Grathwohl, Will, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. 2018. FFJORD: *Free-form Continuous Dynamics for Scalable Reversible Generative Models*. Technical report. arXiv: 1810.01367.
- Griewank, A., and A Walther. 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Second. SIAM.
- Grosse, R. 2018. *Automatic Differentiation*. CSC321 Lecture 10. University of Toronto.
- Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. 2017. *Improved training of Wasserstein GANs*. Technical report. arXiv: 1704.00028.
- Gutmann, Michael, and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *Journal of Machine Learning Research* 9: 297–304.
- Hamilton, W. L. 2020. *Graph Representation Learning*. Morgan / Claypool.
- Hartley, R., and A. Zisserman. 2004. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press.
- Hassibi, B., and D. G. Stork. 1993. Second order derivatives for network pruning: optimal brain surgeon. In *Proceedings International Conference on Neural Information Processing Systems (NeurIPS)*, edited by S. J. Hanson, J. D. Cowan, and C. L. Giles, 5: 164–171. Morgan Kaufmann.
- Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning*. Second. Springer.
- Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57: 97–109.
- He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. 2021. *Masked Autoencoders Are Scalable Vision Learners*. Technical report. arXiv: 2111.06377.

- He, Kaiming, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2019. *Momentum Contrast for Unsupervised Visual Representation Learning*. Technical report. arXiv: 1911.05722.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015a. *Deep Residual Learning for Image Recognition*. Technical report. arXiv: 1512.03385.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015b. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. Technical report. arXiv: 1502.01852.
- Henrion, M. 1988. Propagation of Uncertainty by Logic Sampling in Bayes' Networks. In *Uncertainty in Artificial Intelligence*, edited by J. F. Lemmer and L. N. Kanal, 2: 149–164. North Holland.
- Higgins, I., L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinik, S. Mohamed, and A. Lerchner. 2017. β -VAE: learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference Learning Representations (ICLR)*.
- Hinton, G. E. 2012. *Neural Networks for Machine Learning. Lecture 6.5. Coursera Lectures*.
- Hinton, G. E., M. Welling, Y. W. Teh, and S Osindero. 2001. A new view of ICA. In *Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation*, vol. 3.
- Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. 2015. *Distilling the Knowledge in a Neural Network*. Technical report. arXiv: 1503.02531.
- Hinton, Geoffrey E. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14: 1771–1800.
- Ho, Jonathan, Ajay Jain, and Pieter Abbeel. 2020. *Denoising Diffusion Probabilistic Models*. Technical report. arXiv: 2006.11239.
- Ho, Jonathan, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. 2021. *Cascaded Diffusion Models for High Fidelity Image Generation*. Technical report. arXiv: 2106.15282.
- Hochreiter, S., and J. Schmidhuber. 1997. Long short-term Memory. *Neural Computation* 9 (8): 1735–1780.
- Højen-Sørensen, P. A., O. Winther, and L. K. Hansen. 2002. Mean field approaches to independent component analysis. *Neural Computation* 14 (4): 889–918.
- Holtzman, Ari, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. *The Curious Case of Neural Text Degeneration*. Technical report. arXiv: 1904.09751.
- Hornik, K., M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2 (5): 359–366.
- Hospedales, Timothy, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Metal-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (9): 5149–5169.
- Hotelling, H. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* 24: 417–441.

- Hotelling, H. 1936. Relations between two sets of variables. *Biometrika* 28: 321–377.
- Hu, Anthony, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. 2023. *GAIA- 1: A Generative World Model for Autonomous Driving*. Technical report. arXiv: 2309.17080.
- Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. *LoRA: LowRank Adaptation of Large Language Models*. Technical report. arXiv: 2106.09685.
- Hubel, D. H., and T. N. Wiesel. 1959. Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology* 148: 574–591.
- Hyvärinen, A. 2005. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research* 6: 695–709.
- Hyvärinen, A., and E. Oja. 1997. A fast fixed-point algorithm for independent component analysis. *Neural Computation* 9 (7): 1483–1492.
- Hyvärinen, Aapo, Jarmo Hurri, and Patrick O. Hoyer. 2009. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*. Springer.
- Ioffe, S., and C. Szegedy. 2015. Batch normalization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 448–456.
- Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation* 3 (1): 79–87.
- Jebara, T. 2004. *Machine Learning: Discriminative and Generative*. Kluwer.
- Jensen, C., A. Kong, and U. Kjaerulff. 1995. Blocking Gibbs sampling in very large probabilistic expert systems. *International Journal of Human Computer Studies. Special Issue on Real-World Applications of Uncertain Reasoning*. 42: 647–666.
- Jolliffe, I. T. 2002. *Principal Component Analysis*. Second. Springer.
- Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, and Olaf Ronneberger. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596: 583–589.
- Jutten, C., and J. Herault. 1991. Blind separation of sources, 1: An adaptive algorithm based on neuromimetic architecture. *Signal Processing* 24 (1): 1–10.
- Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. *Scaling Laws for Neural Language Models*. Technical report. arXiv: 2001.08361.
- Karras, Tero, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. Technical report. arXiv: 1710.10196.
- Karush, W. 1939. Minima of functions of several variables with inequalities as side constraints. Master's thesis, Department of Mathematics, University of Chicago.
- Khosla, Prannay, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. *Supervised Contrastive Learning*. Technical report. arXiv: 2004.11362.

- Kingma, D., and J. Ba. 2014. Adam: *A method for stochastic optimization*. Technical report. arXiv: 1412.6980.
- Kingma, D. P., and M. Welling. 2013. Autoencoding variational Bayes. In *Proceedings of the International Conference on Machine Learning (ICML)*. ArXiv: 1312.6114.
- Kingma, Diederik P., and Max Welling. 2019. *An Introduction to Variational Auto-encoders*. Technical report. arXiv: 1906.02691.
- Kingma, Durk P, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems* 29.
- Kipf, Thomas N., and Max Welling. 2016. *Semi-Supervised Classification with Graph Convolutional Networks*. Technical report. arXiv: 1609.02907.
- Kloeden, Peter E, and Eckhard Platen. 2013. *Numerical solution of stochastic differential equations*. Vol. 23. Stochastic Modelling and Applied Probability. Springer.
- Kobyzev, I., S. J. D. Prince, and M. A. Brubaker. 2019. Normalizing flows: an introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (11): 3964–3979.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, vol. 25.
- Kuhn, H. W., and A. W. Tucker. 1951. Nonlinear programming. In *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probabilities*, 481–492. University of California Press.
- Kullback, S., and R. A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics* 22 (1): 79–86.
- Kurková, V., and P. C. Kainen. 1994. Functionally Equivalent Feed-forward Neural Networks. *Neural Computation* 6 (3): 543–558.
- Lasserre, J., Christopher M. Bishop, and T. Minka. 2006. Principled hybrids of generative and discriminative models. In *Proceedings 2006 IEEE Conference on Computer Vision and Pattern Recognition*, New York.
- Lauritzen, S. L. 1996. *Graphical Models*. Oxford University Press.
- Lawley, D. N. 1953. A Modified Method of Estimation in Factor Analysis and Some Large Sample Results. In *Uppsala Symposium on Psychological Factor Analysis*, 35–42. Number 3 in Nordisk Psykologi Monograph Series. Uppsala: Almqvist / Wiksell.
- Lazarsfeld, P. F., and N. W. Henry. 1968. *Latent Structure Analysis*. Houghton Mifflin.
- LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation Applied to Handwritten ZIP Code Recognition. *Neural Computation* 1 (4): 541–551.

- LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* 86: 2278–2324.
- LeCun, Y., J. S. Denker, and S. A. Solla. 1990. Optimal Brain Damage. In *Proceedings International Conference on Neural Information Processing Systems (NeurIPS)*, edited by D. S.
- Touretzky, 2: 598–605. Morgan Kaufmann. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature* 512: 436–444.
- LeCun, Yann, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. 2006. A Tutorial on Energy-Based Learning. In *Predicting Structured Data*, edited by G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar. MIT Press.
- Leen, T. K. 1995. From data distributions to regularization in invariant learning. *Neural Computation* 7: 974–981.
- Leshno, M., V. Y. Lin, A. Pinkus, and S. Schocken. 1993. Multilayer feedforward networks with a polynomial activation function can approximate any function. *Neural Networks* 6: 861–867.
- Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2017. *Visualizing the Loss Landscape of Neural Nets*. Technical report. arXiv: 1712.09913.
- Li, Junnan, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation*. Technical report. arXiv: 2201.12086.
- Lin, Min, Qiang Chen, and Shuicheng Yan. 2013. *Network in Network*. Technical report. arXiv: 1312.4400.
- Lin, Tianyang, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. *A Survey of Transformers*. Technical report. arXiv: 2106.04554.
- Lipman, Yaron, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2022. *Flow Matching for Generative Modeling*. Technical report arXiv: 2210.02747. <https://arxiv.org/>.
- Liu, Pengfei, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*. Technical report. arXiv: 2107.13586.
- Lloyd, S. P. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28 (2): 129–137.
- Long, Jonathan, Evan Shelhamer, and Trevor Darrell. 2014. *Fully Convolutional Networks for Semantic Segmentation*. Technical report. arXiv: 1411.4038.
- Luo, Calvin. 2022. *Understanding Diffusion Models: A Unified Perspective*. Technical report. arXiv: 2208.11970.
- Lütkepohl, H. 1996. *Handbook of Matrices*. Wiley.
- MacKay, D. J. C. 1992. A Practical Bayesian Framework for Back-propagation Networks. *Neural Computation* 4 (3): 448–472.
- MacKay, D. J. C. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

- MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, edited by L. M. LeCam and J. Neyman, I: 281–297. University of California Press.
- Magnus, J. R., and H. Neudecker. 1999. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley.
- Mallat, S. 1999. *A Wavelet Tour of Signal Processing*. Second. Academic Press.
- Mao, X., Q. Li, H. Xie, R. Lau, Z. Wang, and S. Smolley. 2016. *Least Squares Generative Adversarial Networks*. Technical report. arXiv: 1611.04076.
- Mardia, K. V., and P. E. Jupp. 2000. *Directional Statistics*. Wiley.
- Martens, James, Ilya Sutskever, and Kevin Swersky. 2012. Estimating the Hessian by Backpropagating Curvature. In *Proceedings of the International Conference on Machine Learning (ICML)*. ArXiv: 1206.6464.
- McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. Second. Chapman / Hall.
- McCulloch, W. S., and W. Pitts. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. Reprinted in Anderson and Rosenfeld (1988), *Bulletin of Mathematical Biophysics* 5: 115–133.
- McLachlan, G. J., and T. Krishnan. 1997. *The EM Algorithm and its Extensions*. Wiley.
- McLachlan, G. J., and D. Peel. 2000. *Finite Mixture Models*. Wiley.
- Meng, X. L., and D. B. Rubin. 1993. Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika* 80: 267–278.
- Mescheder, L., A. Geiger, and S. Nowozin. 2018. *Which Training Methods for GANs do actually Converge?* Technical report. arXiv: 1801.04406.
- Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics* 21 (6): 1087–1092.
- Metropolis, N., and S. Ulam. 1949. The Monte Carlo method. *Journal of the American Statistical Association* 44 (247): 335–341.
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. *Efficient Estimation of Word Representations in Vector Space*. Technical report. arXiv: 1301.3781.
- Minsky, M. L., and S. A. Papert. 1969. *Perceptrons*. Expanded edition 1990. MIT Press.
- Mirza, M., and S. Osindero. 2014. *Conditional Generative Adversarial Nets*. Technical report. arXiv: 1411.1784.
- Miskin, J. W., and D. J. C. MacKay. 2001. Ensemble learning for blind source separation. In *Independent Component Analysis: Principles and Practice*, edited by S. J. Roberts and R. M. Everson. Cambridge University Press.
- Møller, M. 1993. Efficient Training of Feed-Forward Neural Networks. PhD diss., Aarhus University, Denmark.

- Montúfar, G. F., R. Pascanu, K. Cho, and Y. Bengio. 2014. On the number of linear regions of deep neural networks. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. ArXiv: 1402.1869.
- Mordvintsev, Alexander, Christopher Olah, and Mike Tyka. 2015. *Inceptionism: Going Deeper into Neural Networks*. Google AI blog.
- Murphy, Kevin P. 2022. *Probabilistic Machine Learning: An introduction*. MIT Press. probml.ai.
- Murphy, Kevin P. 2023. *Probabilistic Machine Learning: Advanced Topics*. MIT Press. <http://probml.github.io/book2>.
- Nakkiran, Preetum, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. 2019. *Deep Double Descent: Where Bigger Models and More Data Hurt*. Technical report. arXiv: 1912.02292.
- Neal, R. M. 1993. *Probabilistic inference using Markov chain Monte Carlo methods*. Technical report CRG-TR-93-1. Department of Computer Science, University of Toronto, Canada.
- Neal, R. M. 1999. Suppressing random walks in Markov chain Monte Carlo using ordered over-relaxation. In *Learning in Graphical Models*, edited by Michael I. Jordan, 205–228. MIT Press.
- Neal, R. M., and G. E. Hinton. 1999. A new view of the EM algorithm that justifies incremental and other variants. In *Learning in Graphical Models*, edited by M. I. Jordan, 355–368. MIT Press.
- Nelder, J. A., and R. W. M. Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society, A* 135: 370–384.
- Nesterov, Y. 2004. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer.
- Nichol, Alex, and Prafulla Dhariwal. 2021. *Improved Denoising Diffusion Probabilistic Models*. Technical report. arXiv: 2102.09672.
- Nichol, Alex, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. *GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models*. Technical report. arXiv: 2112.10741.
- Nocedal, J., and S. J. Wright. 1999. *Numerical Optimization*. Springer.
- Noh, Hyeyoung, Seunghoon Hong, and Bohyung Han. 2015. *Learning Deconvolution Network for Semantic Segmentation*. Technical report. arXiv: 1505.04366.
- Nowlan, S. J., and G. E. Hinton. 1992. Simplifying neural networks by soft weight sharing. *Neural Computation* 4 (4): 473–493.
- Ogden, R. T. 1997. *Essential Wavelets for Statistical Applications and Data Analysis*. Birkhäuser.
- Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. *Pixel Recurrent Neural Networks*. Technical report. arXiv: 1601.06759.

- Oord, Aaron van den, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016. *Conditional Image Generation with PixelCNN Decoders*. Technical report. arXiv: 1606.05328.
- Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. 2018. *Representation Learning with Contrastive Predictive Coding*. Technical report. arXiv: 1807.03748.
- Oord, Aaron van den, Oriol Vinyals, and Koray Kavukcuoglu. 2017. *Neural Discrete Representation Learning*. Technical report. arXiv: 1711.00937.
- OpenAI. 2023. *GPT-4 Technical Report*. Technical report. arXiv: 2303.08774.
- Opper, M., and O. Winther. 2000. Gaussian processes and SVM: mean field theory and leave-one-out. In *Advances in Large Margin Classifiers*, edited by A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Shuurmans, 311–326. MIT Press.
- Papamakarios, G., T. Pavlakou, and Iain Murray. 2017. Masked Autoregressive Flow for Density Estimation. In Proceedings of the *International Conference on Neural Information Processing Systems (NeurIPS)*, vol. 30.
- Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. 2019. *Normalizing Flows for Probabilistic Modeling and Inference*. Technical report. arXiv: 1912.02762.
- Parisi, Giorgio. 1981. Correlation functions and computer simulations. *Nuclear Physics B* 180: 378–384.
- Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.
- Pearlmutter, B. A. 1994. Fast exact multiplication by the Hessian. *Neural Computation* 6 (1): 147–160.
- Pearlmutter, B. A., and L. C. Parra. 1997. Maximum likelihood source separation: a context-sensitive generalization of ICA. In *Advances in Neural Information Processing Systems*, edited by M. C. Mozer, M. I. Jordan, and T. Petsche, 9: 613–619. MIT Press.
- Pearson, Karl. 1901. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, Sixth Series* 2: 559–572.
- Phuong, Mary, and Marcus Hutter. 2022. *Formal Algorithms for Transformers*. Technical report. arXiv: 2207.09238.
- Prince, Simon J.D. 2020. *Variational autoencoders*. <https://www.borealisai.com/researchblogs/tutorial-5-variational-auto-encoders>.
- Prince, Simon J.D. 2023. *Understanding Deep Learning*. MIT Press. <http://udlbook.com>.
- Radford, A., L. Metz, and S. Chintala. 2015. *Unsupervised representation learning with deep convolutional generative adversarial networks*. Technical report. arXiv: 1511.06434.
- Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, et al. 2021. *Learning Transferable Visual Models From Natural Language Supervision*. Technical report. arXiv: 2103.00020.

- Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. *Language Models are Unsupervised Multitask Learners*. Technical report. OpenAI.
- Rakhimov, Ruslan, Denis Volkonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. 2020. *Latent Video Transformer*. Technical report. arXiv: 2006.10704.
- Ramachandran, P., B. Zoph, and Q. V. Le. 2017. *Searching for Activation Functions*. Technical report. arXiv:1710.05941v2.
- Rao, C. R., and S. K. Mitra. 1971. *Generalized Inverse of Matrices and Its Applications*. Wiley.
- Redmon, Joseph, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. *You Only Look Once: Unified, Real-Time Object Detection*. Technical report. arxiv: 1506.02640.
- Ren, Shaoqing, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. *Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks*. Technical report. arxiv: 1506.01497.
- Rezende, Danilo J, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 1278–1286.
- Ricotti, L. P., S. Ragazzini, and G. Martinelli. 1988. Learning of word stress in a sub-optimal second order backpropagation neural network. In *Proceedings of the IEEE International Conference on Neural Networks*, 1: 355–361. IEEE.
- Robert, C. P., and G. Casella. 1999. *Monte Carlo Statistical Methods*. Springer.
- Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. *High-Resolution Image Synthesis with Latent Diffusion Models*. Technical report. arXiv: 2112.10752.
- Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, edited by N. Navab, J. Hornegger, W. Wells, and A. Frangi. Springer.
- Rosenblatt, F. 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan.
- Roweis, S. 1998. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*, edited by M. I. Jordan, M. J. Kearns, and S. A. Solla, 10: 626–632. MIT Press.
- Roweis, S., and Z. Ghahramani. 1999. A unifying review of linear Gaussian models. *Neural Computation* 11 (2): 305–345.
- Rubin, D. B., and D. T. Thayer. 1982. EM algorithms for ML factor analysis. *Psychometrika* 47 (1): 69–76.
- Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations*

- in the Microstructure of Cognition*, edited by D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, vol. 1: Foundations, 318–362. Reprinted in Anderson and Rosenfeld (1988). MIT Press.
- Ruthotto, L., and E. Haber. 2021. *An introduction to deep generative modeling*. Technical report. arXiv: 2103.05180.
- Sagan, H. 1969. *Introduction to the Calculus of Variations*. Dover.
- Saharia, Chitwan, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. 2021. *Palette: Image-to-Image Diffusion Models*. Technical report. arXiv: 2111.05826.
- Saharia, Chitwan, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, et al. 2022. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. Technical report. arXiv: 2205.11487.
- Saharia, Chitwan, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. 2021. *Image Super-Resolution via Iterative Refinement*. Technical report. arXiv: 2104.07636.
- Santurkar, S., D. Tsipras, A. Ilyas, and A. Madry. 2018. *How does batch normalization help optimization?* Technical report. arXiv: 1805.11604.
- Satorras, Victor Garcia, Emiel Hoogeboom, and Max Welling. 2021. *E(n) Equivariant Graph Neural Networks*. Technical report. arXiv: 2102.09844.
- Schölkopf, B., and A. J. Smola. 2002. *Learning with Kernels*. MIT Press.
- Schuhmann, Christoph, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsu-zaki. 2021. *LAION400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs*. Technical report. arXiv: 2111.02114.
- Schuster, Mike, and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5149–5152.
- Selvaraju, Ramprasaath R., Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*. Technical report. arXiv: 1610.02391.
- Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2015. *Neural Machine Translation of Rare Words with Subword Units*. Technical report. arXiv: 1508.07909.
- Sermanet, Pierre, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. 2013. *OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks*. Technical report. arXiv: 1312.6229.
- Shachter, R. D., and M. Peot. 1990. Simulation Approaches to General Probabilistic Inference on Belief Networks. In *Uncertainty in Artificial Intelligence*, edited by P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, vol. 5. Elsevier.
- Shannon, C. E. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27 (3): 379–423 and 623–656.

- Shen, Sheng, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2019. *Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT*. Technical report. arXiv: 1909.05840.
- Simard, P., B. Victorri, Y. LeCun, and J. Denker. 1992. Tangent prop – a formalism for specifying selected invariances in an adaptive network. In *Advances in Neural Information Processing Systems*, edited by J. E. Moody, S. J. Hanson, and R. P. Lippmann, 4: 895–903. Morgan Kaufmann.
- Simard, P. Y., D. Steinkraus, and J. Platt. 2003. Best practice for convolutional neural networks applied to visual document analysis. In *Proceedings International Conference on Document Analysis and Recognition (ICDAR)*, 958–962. IEEE Computer Society.
- Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Computer Vision and Pattern Recognition*. ArXiv: 1312.6034.
- Simonyan, Karen, and Andrew Zisserman. 2014. *Very Deep Convolutional Networks for LargeScale Image Recognition*. Technical report. arXiv: 1409.1556.
- Sirovich, L. 1987. Turbulence and the Dynamics of Coherent Structures. *Quarterly Applied Mathematics* 45 (3): 561–590.
- Sohl-Dickstein, Jascha, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. Technical report. arXiv: 1503.03585.
- Sønderby, C., J. Caballero, L. Theis, W. Shi, and F. Huszár. 2016. *Amortised MAP inference for image super-resolution*. Technical report. arXiv: 1610.04490.
- Song, Jiaming, Chenlin Meng, and Stefano Ermon. 2020. *Denoising Diffusion Implicit Models*. Technical report. arXiv: 2010.02502.
- Song, Yang, and Stefano Ermon. 2019. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, 11895–11907. ArXiv: 1907.05600.
- Song, Yang, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. 2019. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, 204. ArXiv: 1905.07088.
- Song, Yang, and Diederik P. Kingma. 2021. *How to Train Your Energy-Based Models*. Technical report. arXiv: 2101.03288.
- Song, Yang, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. *Score-Based Generative Modeling through Stochastic Differential Equations*. Technical report. arXiv: 2011.13456.
- Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15: 1929–1958.
- Stone, J. V. 2004. *Independent Component Analysis: A Tutorial Introduction*. MIT Press.

- Sutskever, I., J. Martens, G. Dahl, and G. E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.
- Sutton, R. 2019. *The Bitter Lesson*. URL: incompleteideas.net/IncIdeas/BitterLesson.html.
- Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. *Intriguing properties of neural networks*. Technical report. arXiv: 1312.6199.
- Szeliski, R. 2022. *Computer Vision: Algorithms and Applications*. Second. Springer.
- Tarassenko, L. 1995. Novelty detection for the identification of masses in mammograms. In *Proceedings of the Fourth IEE International Conference on Artificial Neural Networks*, 4: 442–447. IEE.
- Tay, Yi, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. *Efficient Transformers: A Survey*. Technical report. arXiv: 2009.06732.
- Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B* 58: 267–288.
- Tipping, M. E., and Christopher M. Bishop. 1997. *Probabilistic Principal Component Analysis*. Technical report NCRG/97/010. Neural Computing Research Group, Aston University.
- Tipping, M. E., and Christopher M. Bishop. 1999. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society, Series B* 21 (3): 611–622.
- Vapnik, V. N. 1995. *The nature of statistical learning theory*. Springer.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. *Attention Is All You Need*. Technical report. arXiv: 1706.03762.
- Veličković, Petar. 2023. *Everything is Connected: Graph Neural Networks*. Technical report. arXiv: 2301.08210.
- Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. *Graph Attention Networks*. Technical report. arXiv: 1710.10903.
- Vidakovic, B. 1999. *Statistical Modelling by Wavelets*. Wiley.
- Vig, Jesse, Ali Madani, Lav R. Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2020. *BERTology Meets Biology: Interpreting Attention in Protein Language Models*. Technical report. arXiv: 2006.15222.
- Vincent, P. 2011. A connection between score matching and denoising autoencoders. *Neural Computation* 23: 1661–1674.
- Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)*.
- Walker, A. M. 1969. On the asymptotic behavior of posterior distributions. *Journal of the Royal Statistical Society, B* 31 (1): 80–88.

- Wang, Chengyi, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, et al. 2023. *Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers*. Technical report. arXiv: 2301.02111.
- Weisstein, E. W. 1999. *CRC Concise Encyclopedia of Mathematics*. Chapman / Hall / CRC.
- Welling, Max, and Yee Whye Teh. 2011. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the International Conference on Machine Learning (ICML)*.
- Williams, P. M. 1996. Using neural networks to model conditional multivariate densities. *Neural Computation* 8 (4): 843–854.
- Williams, R J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8: 229–256.
- Winn, J., C. M. Bishop, T. Diethe, J. Guiver, and Y. Zaykov. 2023. *Model-Based Machine Learning*. www.mbmbook.com. Chapman / Hall.
- Wolpert, D. H. 1996. The lack of a-priori distinctions between learning algorithms. *Neural Computation* 8: 1341–1390.
- Wu, Zhirong, Yuanjun Xiong, Stella Yu, and Dahua Lin. 2018. *Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination*. Technical report. arXiv: 1805.01978.
- Wu, Zonghan, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. *A Comprehensive Survey on Graph Neural Networks*. Technical report. arXiv: 1901.00596.
- Yan, Wilson, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. 2021. *VideoGPT: Video Generation using VQ-VAE and Transformers*. Technical report. arXiv: 2104.10157.
- Yang, Ruihan, Prakhar Srivastava, and Stephan Mandt. 2022. *Diffusion Probabilistic Modeling for Video Generation*. Technical report. arXiv: 2203.09481.
- Yilmaz, Fatih Furkan, and Reinhard Heckel. 2022. *Regularization-wise double descent: Why it occurs and how to eliminate it*. Technical report. arXiv: 2206.01378.
- Yosinski, Jason, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. 2015. *Understanding Neural Networks Through Deep Visualization*. Technical report. arXiv: 1506.06579.
- Yu, Jiahui, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. 2021. *Vector-quantized Image Modeling with Improved VQGAN*. Technical report. arXiv: 2110.04627.
- Yu, Jiahui, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, et al. 2022. *Scaling Autoregressive Models for Content-Rich Text-to-Image Generation*. Technical report. arXiv: 2206.10789.
- Yu, Lili, Bowen Shi, Ramakanth Pasunuru, Benjamin Muller, Olga Golovneva, Tianlu Wang, Arun Babu, et al. 2023. *Scaling Autoregressive Multi-Modal Models: Pretraining and Instruction Tuning*. Technical report. arXiv: 2309.02591.

- Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. 2017. *Deep Sets*. Technical report. arXiv: 1703.06114.
- Zarchan, P., and H. Musoff. 2005. *Fundamentals of Kalman Filtering: A Practical Approach*. Second. AIAA.
- Zeiler, Matthew D., and Rob Fergus. 2013. *Visualizing and Understanding Convolutional Networks*. Technical report. arXiv: 1311.2901.
- Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. *Understanding deep learning requires rethinking generalization*. Technical report. arXiv: 1611.03530.
- Zhao, Wayne Xin, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, et al. 2023. *A Survey of Large Language Models*. Technical report. arXiv: 2303.18223.
- Zhou, Jie, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. *Graph Neural Networks: A Review of Methods and Applications*. Technical report. arXiv: 1812.08434.
- Zhou, Y., and R. Chellappa. 1988. Computation of optic flow using a neural network. In *International Conference on Neural Networks*, 71–78. IEEE.
- Zhu, J-Y, T. Park, P. Isola, and A. Efros. 2017. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. Technical report. arXiv: 1703.10593.

Предметный указатель

Латиница

AdaGrad, 275
AlexNet, 362
BERT, 460
BigGAN, 624
ChatGPT, 467
CLIP (модель предварительного обучения контрастивно-языковым изображениям), 240
CycleGAN, 624
D-разделение, 404
DeepDream, 371
E шаг, 539
ЕМ для факторного анализа, 604
F-оценка, 192
GAN Вассерштейна, 622
Grad-CAM, 368
ImageGPT, 472
M шаг, 539
N-граммная модель, 450
PixelCNN, 471
PixelRNN, 471
ReLU, см. Выпрямленный линейный элемент
RMSProp, 275
ROC-кривая, 190

А

Автоассоциативная нейронная сеть, 647
Автокодировщик, 235, 647
Автоматическое дифференцирование, 46, 286, 301
Агрегация, 490

Адаптация низкого ранга (LoRA), 465
Адаптивная выборка
 Метрополиса с отклонением, 512
 с отклонением, 511
Адъюнкт, 641
Активация, 40
Активное ограничение, 709
Алгоритм
 блокирующей выборки Гиббса, 527
 максимизации ожиданий
 (ЕМ-алгоритм), 599
 Метрополиса, 517
 Метрополиса–Гастингса, 521
 наименьших среднеквадратичных значений (LMS), 154
 ожидания-максимизации
 (ЕМ-алгоритм), 549, 554
 условной максимизации ожиданий, 569
 К-средних, 472, 539
Алеаторная неопределенность, 47
Амортизированный вывод, 655, 657
Анализ
 главных компонентов (PCA), 229, 576
 канонических корреляций, 581
 латентных классов, 560
 независимых компонентов (ICA), 596
Ансамбль, см. Комитет
Апостериорная вероятность, 56
Апостериорный коллапс, 663
Априорная вероятность, 56
Априорный параметр, 568
Архитектура U-net, 384
Аугментация данных, 239
Аудиоданные, 473

Б

Базисная функция, 148
Базовая модель, 46, 425, 465
Базовое распределение, 631

Байесова вероятность, 83
 Байесовское машинное обучение, 86
 Биграммная модель, 450
 Биективная функция, 631
 Бинарный классификатор, 399
 Биноминальное распределение, 96
 Блокированный путь, 410
 Большая языковая модель (LLM), 26, 453
 Быстрая региональная CNN (fast R-CNN), 379

В

Валидационный (удержанный) набор, 37
 Вариационный автокодировщик, 611, 655
 Вариационный вывод, 599
 Вейвлет, 150
 Вектор
 весовых коэффициентов, 171
 вложения, 484
 кодовой книги, 471, 544
 Векторное квантование, 471, 544
 Векторное представление слов, 444
 Вероятностная модель графа, 390
 Вероятность, 49
 Верхняя релаксация, 526
 Вес, 40, 76, 225
 важности, 514
 Взаимная информация, 82, 83
 Взрывающийся градиент, 279, 452
 Внимание, 425, 426
 Внутренний сдвиг ковариаций, 281
 Вогнутая функция, 80
 Воксель, 349
 Вспомогательное распределение, 510
 Встраиваемое пространство, 235
 Выборка, 505
 верхнего p (ядерная выборка), 459
 Гиббса, 523
 Ланжевена, 528, 533
 Монте-Карло, 505
 по важности, 509, 513
 по значимости с повторной
 выборкой, 516
 по предкам, 527
 по собственной важности, 528
 со взвешенной вероятностью, 528
 с отклонением, 509
 top-K, 459
 Выбор модели, 31, 37
 Выборочное среднее значение, 65
 Выброс, 177, 209

Выпрямленный линейный элемент
 (ReLU), 230
 Выявление
 выбросов, 185
 новизны, 185

Г

Гамильтонов метод Монте-Карло, 529
 Гамма-функция, 168
 Гауссова смесь, 119, 544
 Гауссово (нормальное) распределение, 61
 Генеративная вероятностная модель, 171
 Генеративная модель, 26, 185, 587
 Генеративный визуальный
 трансформер, 470
 Генеративный ИИ, 26
 Гетероскедастическая модель, 249
 Гибридный метод Монте-Карло, 529
 Гиперпараметр, 37
 Гипотеза масштабной
 инвариантности, 426
 Главное подпространство, 576
 Глобальный минимум, 261
 Глубокая нейронная сеть, 44
 Глубокий набор, 494
 Глубокое геометрическое обучение, 314
 Глубокое обучение, 22, 44
 Голова внимания, 436
 Градиентная динамика Ланжевена, 532
 Градиентный спуск, 265
 Градиентный штраф GAN
 Вассерштейна, 623
 Граница Маркова, 415
 График отжига, 663
 Графический процессор (GPU), 44, 242,
 426
 Графовая нейронная сеть, 483
 Графовая сеть с вниманием, 497
 Группа, 314

Д

Данные
 изображения, 349
 Олд Фейтфул, 540, 551, 583
 Данные MNIST, 575
 Двойной спуск, 327
 Дельта-функция Дирака, 59
 Диагональная ковариационная
 матрица, 105
 Динамика Ланжевена с отжигом, 685

Дискретная латентная переменная, 537
Дискретная переменная, 95, 394
Дискриминантная вероятностная модель, 171
Дискриминантная функция, 170, 184
Дискриминативная модель, 185
Дискриминация по образцу, 239
Дисперсия, 61, 163
 выборки, 65
 шума, 684
Дифференциальная энтропия, 78
Диффузионная вероятностная модель шумоподавления (DDPM), 666
Диффузионная модель, 611
Диффузионное ядро, 668
Дополнение Шура, 109
Достаточная статистика, 98, 116, 131
Дочерний узел, 305, 392

Ж

Жесткое внимание, 431

З

Загрузочный набор данных, 338
Задача заполнения графа, 483
Закон
 Гука, 603
 Мура, 45
Заполнение, 355
Запрос, 467
Знаковый метод быстрого градиента, 369

И

Идентичная свертка, 355
Извлечение признаков, 44, 148
Изотропная ковариация, 106
Импульс Нестерова, 273
Импульсный параметр, 271
Инвариантность
 перевода, 314
 преобразования, 312
Инверсия, 314
Индуктивное обучение, 496
Индуктивное смещение, 223, 311
Инженерия запросов, 467
Инициализация Ге, 268
Интегральная функция распределения, 57
Искусственная нейронная сеть, 40
Искусственный интеллект, 22
 общего назначения, 22

История машинного обучения, 39
Исчезающий градиент, 278
Итеративный пересчет наименьших квадратов (IRLS), 205

Й

Йеллоустонский национальный парк, 540

К

Каноническая функция связи, 209
Карта
 значимости, 368
 признаков, 353
Касательное распространение, 315
Квадратичное смещение, 163
Квадратичный дискриминант, 196
Классификатор
 наивного Байеса, 448
 «один против одного», 174
 «один против остальных», 173
Классификация, 24
 графов, 496
 изображений, 348
 ребер, 496
Кластеризация К-средних, 538
Ковариационная матрица, 105
Ковариация, 61
Кодирование «1 из K», 97, 175, 176, 245, 561
Кодировка пар байтов, 447
Кодирующий трансформер, 460
Комитет, 337
Компенсация априорных вероятностей классов, 187
Композиционное индуктивное смещение, 234
Компьютерное зрение, 348
Контрастивное обучение, 238
Контролируемое обучение, 496
Коэффициенты смешивания, 120
Кривая обучения, 274, 325
Кривая операционной характеристики приемника, 191
Критерий принятия, 182, 186, 189
Круговое нормальное распределение, см. Распределение фон Мизеса

Л

Лассо, 323
Латентная диффузионная модель, 690

Латентная (скрытая) переменная, 106
 Лексема, 428
 Лексическая обработка, 447
 Линейная гауссова модель, 397
 Линейная динамическая система (фильтр Калмана), 598
 Линейная модель, 30
 Линейная регрессия, 67, 147
 Линейно независимый набор векторов, 697
 Линейно разделимый набор данных, 170
 Линейный дискриминант, 171
 Логарифм отношения шансов, 194
 Логистическая регрессия, 177, 203
 Логистическая регрессия для нескольких классов, 205
 Логистическая сигмоидная функция, 71, 128, 203
 Логическая выборка, 528
 Ложноотрицательный результат, 50
 Ложноположительный результат, 50
 Локальная квадратичная аппроксимация, 261
 Локальный минимум, 261
 Лучевой поиск, 458

M

Макросостояние, 76
 Максимальная апостериорная оценка (MAP), 85
 Максимальная отмена пулинга, 382
 Максимальная энтропия, 78
 Максимальное правдоподобие, 64, 151, 549
 Максимальный пулинг, 358
 Максимизация ожидания, 121, 600
 правдоподобия, 530
 Максимум апостериорной вероятности (MAP), 556
 Марковское покрытие (покрытие Маркова), 415
 Маскированное внимание, 456
 Маскированный автокодировщик, 652
 Маскированный поток авторегрессии, 637
 Массив данных Iris, 217
 Масштабированное точечное самонимание, 435
 Масштабная инвариантность, 314
 Матрица Гессиана, 262, 296

корреляции, 584
 неточностей, 189
 плана, 152
 смежности, 485
 Якоби, 72, 294
 Машина опорных векторов (SVM), 224
 Машинное обучение на графах, 483
 Метаобучение, 238
 Метод Бокса–Мюллера, 508
 дополнения набора данных, 315
 Монте–Карло с цепями Маркова, 517
 перепараметризации, 655, 660, 662
 «резинового листа», 316
 решения Эйлера–Маруямы, 686
 скользящего окна, 375
 сопряженной чувствительности, 641
 усадки, 35
 К-ближайших соседей, 135, 137
 skip-grams, 445
 word2vec, 445
 Методика ближайших соседей, 137
 Микросостояние, 76
 Мини-батч, 267
 Минимизация риска, 186
 Многоголовое внимание, 436
 Многозадачное обучение, 237
 Многообразие, 221
 Многослойная свертка, 360
 Многослойный перцепtron (MLP), 42, 439, 461, 493
 Множественность, 76
 Множитель Лагранжа, 707, 708
 Мода (вершина распределения), 63
 Модель авторегрессии, 418
 графа, 391
 Маркова, 419
 мультимножества слов, 448
 на основе энергии, 529
 пространства состояний, 420
 с долговременной и кратковременной памятью (LSTM), 453
 смеси мнений экспертов, 249
 Момент первого порядка, 63
 Мультимодальный трансформер, 425, 467
 Мягкое внимание, 431
 Мягкое разделение весов, 331

H

Набор данных ImageNet, 361
 для разработки, 37

- Наведение, 687
без классификатора, 688
- Надежность, 177
- Наивная модель Байеса, 188, 411
- Наискорейший спуск, 265
- Направленная модель графа, 391
- Направленная факторизация, 416
- Направленный ациклический граф, 394
- Направленный цикл, 394
- Наращивание данных, 315
- Нарушение симметрии, 268
- Нат, 75
- Неактивное ограничение, 709
- Независимая переменная, 56
- Независимый и одинаково распределенный (i.i.d., IID), 64
- Независимый факторный анализ, 597
- Нейронаука, 364
- Нейронная сеть
- с передачей сообщений, 491
 - с прямой связью, 42
- Нейронное обыкновенное дифференциальное уравнение нейронное обыкновенное дифференциальное уравнение (нейронное ОДУ, neural ODE), 639
- Нелинейная модель латентных переменных, 605
- Немаксимальное подавление, 378
- Неокогнитрон, 365
- Неориентированная модель графа, 391
- Непараметрический метод оценки распределения плотности, 132
- Неполный набор данных, 557
- Непрерывное мультимножество слов, 445
- Неравенство Йенсена, 81
- Несобственное распределение, 59
- Неявная диффузионная модель шумоподавления, 680
- Нижняя вариационная граница, 564
- Нижняя граница доказательства (ELBO), 538, 564, 599, 655, 673
- Нормализация слоя, 281
- Нормализованная экспоненциальная функция (функция softmax), 194
- Нормализованный экспоненциал, 194
- Нормализующий поток, 69
- Нормальное уравнение, 152
- Обобщенная линейная модель, 202, 210
- Обобщенное расстояние, 101
- Обобщенный ЕМ-алгоритм (GEM), 569
- Обработка естественного языка (NLP), 425
- Обратная задача, 247, 311
- Обратная кинематика, 247
- Обратная свертка (деконволюция), 384
- Обратное отношение между смещением и дисперсией, 161
- Обратное распространение, 259, 286
- ошибки, 43, 264, 286
 - по времени, 452
- Обратный поток авторегрессии, 638
- Обратный режим автоматического дифференцирования, 305
- Обучение
- на ограниченных примерах, 238, 467
 - по одному примеру, 238
 - представлениям, 46, 235, 627
 - представлениям графов, 484
 - с подкреплением и обратной связью от человека (RLHF), 467
 - с частичным контролем, 496
 - учиться, 238
- Объединение моделей, 187
- Объяснение, 408
- Обязанность, 120
- Ограничение
- неравенства, 709
 - равенства, 709
- Ограничения фиксированных базисных функций, 215
- Ограничительная рамка, 373
- Однородная цепь Маркова, 520
- Одноточечное (прямое) кодирование, 98
- Ожидание, 60, 539
- Окно Парзена, 135
- Онлайн-алгоритм, 154
- Онлайн-градиентный спуск, 266
- Оператор обновления, 494
- Операторы агрегации, 491
- Оптимизация Adam, 275
- Опция отказа, 183, 186
- Остаточная связь, 334
- Остаточное соединение, 46
- Остаточный блок, 335
- Отбеливание данных, 584
- Отмена пулинга, 382
- Относительная энтропия (дивергенция Кульбака–Лейблера), 80
- Отождествляемость, 549
- Отсев Монте-Карло, 341

0

Обобщение, 29

Оценка
ожиданий, 506
плотности распределения, 63, 94
Штейна, 681

Оценщик
Парзена, 137
трассировки Хатчinsonа, 643

Ошибка перекрестной энтропии, 244

П

Пакетная нормализация, 278
Пакетный метод, 154, 265

Параметр
дисперсии, 63
концентрации, 124
смещения, 148
ширины, 327

Параметрический метод моделирования
распределения плотности, 132

Параметрическое распределение, 94

Передача стиля, 348

Перекрестная валидация, 37
по отдельным образцам, 37

Перекрестная энтропия, 204

Пересечение по объединению (IoU), 374

Периодическая переменная, 121

Перцептрон, 41

Плотность (распределения)
вероятности, 57

Площадь под кривой, 192

Подсказка, 688

Позиционное кодирование, 440

Покрытие Маркова, 415

Полиномиальное распределение, 99

Полное логарифмическое правдоподобие
данных, 600

Полносвязный граф, 393

Полносвязный слой, 363, 367

Полностью сверточная сеть, 384

Полный набор данных, 557

Положительно определенная
матрица, 103, 263, 703

Положительно полуопределенная
матрица, 103

Порог интерполяции, 327

Последовательная оценка, 117

Последовательность, 347

Последовательный алгоритм, см. Онлайн-
алгоритм

Последовательный градиентный
спуск, 154

Потери Минковского, 160

Поток
авторегрессии, 633
сопряжения, 633, 636

Правило
ближайших соседей, 139
произведения, 48
произведения вероятностей, 50, 52
суммы, 48
суммы вероятностей, 50, 52

Правильная свертка, 355

Предварительная активация, 40, 225

Предварительная обработка, 315

Предварительное знание, 311

Предварительное обучение, 237

Предварительно обученный
генеративный трансформер (GPT), 454,
464

Предельная вероятность, 52

Предсказание
глубины, 349
ребер графа, 483

Предшественник узла, 241

Преобразование
Косамби–Карунена–Лоэва, 576

Префиксный запрос, 467

Приближение
внешнего произведения, 297
Левенберга–Марквардта, 298
Стирлинга, 76

Признак, 429

Причинно-следственное внимание, 456

Причинно-следственный процесс, 414

Пробит-регрессия, 209

Проблема узкого места, 452

Прогнозируемое распределение, 69

Произведение Адамара, 634

Проклятие размерности, 133, 216

Прореживание (отсев), 340

Простое случайное блуждание, 519

Протекающий ReLU, 231

Прямая задача, 247

Прямая кинематика, 247

Прямая оценка градиента, 473

Прямое распространение, 288

Псевдоинверсия Мура–Пенроуза, 152

Псевдослучайная выборка, 505

Пулинг, 358

Путь
«голова к голове», 408
«голова к хвосту», 406

P

Равновесное распределение, 521
Радиальная базисная функция, 224
Разбухание выражений, 300
Разделенное представление, 627
Разложение
 по сингулярным значениям, 154
 Холецкого, 509
Размерность встраивания, 329
Разреженная модель, 323
Разупорядоченность, 75
Ранняя остановка, 325
Распознавание объектов на изображении, 348
Распределение
 Бернулли, 95, 244
 Коши, 508
 Лапласа, 59
 условного смешивания, 248
 фон Мизеса, 123
Распределенное представление, 234
Расстояние Махаланобиса, 101
Растровая развертка, 470
Расхождение Дженсена–Шеннона, 629
Реальная NVP, 634, 636
Ребро (связь), 391
Регрессия, 24, 242
Регуляризация, 35, 85
Регуляризованная функция ошибки, 315
Реконструкция сцены, 349
Рекуррентная нейронная сеть (RNN), 425
Рецептивное поле, 492
Рецептивное (рецепторное) поле, 351
Родительский узел, 392

C

Самовнимание, 432
 по точечному произведению, 432
Свертка, 353
 1×1 , 358
 с дробным шагом, 384
 с шагом (сдвигом, страйдом), 356
Сверточная нейронная сеть (CNN), 44, 347
Сверточная сеть LeNet, 361
Сверхвысокое разрешение, 349
Свойства графов, 484
Свойство условной независимости, 188, 403
Связывание параметров, 330
Связь пропущенного слоя, 334

Сегментация изображения, 317, 348
Семантическая сегментация, 380
Семейство альфа, 91
Сеть
 Байеса, 391
 декодировщика, 658
 кодировщика, 657
 прямого распространения (многослойный перцептрон), 215
 смешанной плотности, 246, 249
 с остаточными связями (ResNet), 335
Сжатие данных, 582
 без потерь, 543
 с потерями, 543
Сжимающая функция, 193
Симметрия, 314
 весового пространства, 231
Синтез
 изображений, 348
 речи по тексту, 474
Скорость обучения, 265
Скрытая (латентная) переменная, 420
Скрытая марковская модель, 421
Скрытый элемент, 42, 225
Слабо контролируемый метод, 240
Слепое разделение входных сигналов, 596
Случайная переменная, 50
Случайное поле Маркова (Марковское случайное поле), 391
Смежная переменная, 305
Смесь распределений Бернулли, 118
Смешанное распределение, 118
Смещение, 66, 225
Снижение шума, 611
Совместное использование
 весов, 330
 параметров, 330
Согласование потоков, 644
Создание ансамбля, 338
Соответствие оценки, 681
 шумоподавления, 684
Состязательная атака, 369
Спектrogramма mel, 473
Сравнение моделей, 31
Среднее значение, 62
Среднеквадратичная ошибка (RMS), 32
Средний пулинг, 359
Стандартизация, 540, 583
Стандартное отклонение, 62
Стационарная точка, 261
Степенной метод, 578
Степень свободы, 575

Стохастическая переменная, 50
 Стохастический градиентный спуск, 43, 154, 265
 Стохастическое дифференциальное уравнение (SDE), 685
 Строго вогнутая функция, 81
 Строго выпуклая функция, 80
 Структурированные данные, 94, 347
 Сферизация данных, 584
 Схлопывание мод распределения, 620
 Считывающий слой, 495

Т

Табличка, 400
 Температура, 459
 Тензор, 242
 Теорема
 Байеса, 53
 о бесшумном кодировании, 75
 об отсутствии бесплатного обеда, 228, 312
 о среднем значении, 78
 Теория
 вероятностей, 48
 информации, 73
 меры, 58
 принятия решений, 48, 157
 Тестовый набор, 32, 37
 Тождество Вудбери, 697
 Точная матрица, 107
 Точная настройка, 237
 Точность, 62, 322
 классификатора, 188
 Трансдуктивное обучение, 496
 Транспонирующая свертка, 384
 Трансферное обучение, 24, 236
 Трансформер, 425
 Траншейное расстояние (расстояние Вассерштейна), 622
 Триграммная модель, 450

У

Удержаненный набор, 37
 Узел (вершина), 391
 Узел-коллайдер, 407
 Узел-потомок, 407
 Узел-сородитель, 415
 Улучшение агрегации, 338
 Уменьшение
 весов, 35, 319

параметров, 155
 Универсальная аппроксимация, 227
 Универсальный аппроксиматор, 228
 Уникальность, 595
 Упорядоченная верхняя релаксация, 526
 Управляемый рекуррентный элемент (GRU), 453
 Уравнения Эйлера–Лагранжа, 706
 Усиление (бустинг), 339
 Условная вероятность, 52
 Условная энтропия, 82
 Условное ожидание, 60
 Условное распределение, 107

Ф

Фактор-граф, 392
 Факторизация, 392, 394
 Факторная нагрузка, 595
 Факторный анализ, 576, 594
 Фильтр
 Габора, 364
 Калмана, 421
 ядро, 351
 Фонема, 475
 Формирователь сигнала, 636
 Функционал, 704
 Функция
 активации, 40, 225, 229, 230, 233, 242, 287
 активации swish, 256
 выгоды, 182
 Лагранжа, 708
 отсчета, 533
 оценки, 681
 ошибки, 30, 242
 ошибки суммы квадратов, 69
 потерь, 182
 потерь InfoNCE, 239
 правдоподобия, 64, 608
 пробита, 208
 разбиения, 530
 регрессии, 158
 связи, 636
 сопоставления, 231
 сравнения, 510
 стоимости, 182
 энергии, 529
 logit, 194
 softmax, 129

Х

Характеристическое уравнение, 700

Ц

Центральная предельная теорема, 100

Центральная разность, 292

Цепь Маркова, 419

Ч

Частотная вероятность, 49

Число обусловленности гессиана, 271

Чрезмерная подгонка, 32, 67

Чрезмерное сглаживание, 500

Ш

Шаг (страйд, сдвиг), 356

Шкала наведения, 687

Штрафное слагаемое, 35

Шум, см. Алеаторная неопределенность

Шумное ИЛИ (noisy-OR), 422

Шумовое ядро, 683

Шумоподавляющий автокодировщик, 651

Э

Эквивариантность, 317, 353, 358
перестановок, 486

Экспоненциальное распределение, 59

Экспоненциальное семейство, 127

Эмпирическое распределение, 60

Энтропия, 73, 74

Эпистемическая неопределенность, 47

Эпоха обучения, 266

Этап

вывода, 184

максимизации (M step), 551

ожидания (E step), 551

принятия решения, 184

Эффективная сложность модели, 328

Эффективное число параметров, 320

Я

Ядерная оценка плотности, 137, 683

Ядерная функция, 135

Ядерный метод, 135

Якорь, 238

Книги издательства «ДМК Пресс»
можно купить оптом и в розницу на складе издательства по адресу:
Москва, ул. Электродная, д. 2, стр. 12, офис 7, тел. +7 (499) 322-19-38,
а также заказать на сайте www.dmkpress.com
с доставкой в любой регион РФ

Кристофер М. Бишоп, Хью Бишоп

Глубокое обучение: принципы и концепции

Главный редактор *Мовчан Д. А.*
Зам. главного редактора *Яценков В. С.*
editor@dmkpress.com
Перевод *Бахур В. И.*
Корректор *Абросимова Л. А.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Гарнитура РТ Serif. Печать цифровая.
Усл. печ. л. 60,13. Тираж 200 экз.

Веб-сайт издательства: www.dmkpress.com

«Эта замечательная книга поможет читателю ознакомиться с основными концепциями и достижениями в области глубокого обучения, лежащими в основе современных промышленных систем искусственного интеллекта. Вероятно, на них будут опираться и дальнейшие разработки в области ИИ».

Иошуа Бенджио,
ведущий эксперт в области ИИ, лауреат премии Тьюринга (2018)

Книга предлагает исчерпывающее описание фундаментальных идей, лежащих в основе глубокого обучения и выдержавших испытание временем. Для глубокого понимания принципов машинного обучения требуется определенная математическая подготовка, поэтому в книгу включено основательное введение в теорию вероятностей. Особое внимание уделяется практической ценности изучаемых методов в реальном мире. Сложные концепции представлены в нескольких взаимодополняющих ракурсах, включая текстовые описания, диаграммы, математические формулы и программные псевдокоды.

Издание адресовано как новичкам в машинном обучении, так и опытным специалистам в этой области.



Кристофер М. Бишоп – научный сотрудник Microsoft и директор Microsoft Research AI4Science. Является членом научного общества Дарвиновского колледжа в Кембридже, Королевской инженерной академии и Королевского общества.



Хью Бишоп – ведущий научный сотрудник компании Wayve, работает над созданием и обучением глубоких нейронных сетей для автономного вождения. Имеет степень магистра в области машинного обучения и искусственного интеллекта.

 Springer


издательство
www.dmk.ru



ISBN 978-5-93700-281-5



9 785937 002815 >