

Содержание

От издательства	30
Предисловие	31
Глава 1. Введение	34
1.1. Что такое машинное обучение?.....	34
1.2. Обучение с учителем.....	35
1.2.1. Классификация.....	35
1.2.1.1. Пример: классификация ирисов.....	35
1.2.1.2. Разведочный анализ данных.....	37
1.2.1.3. Обучение классификатора.....	38
1.2.1.4. Минимизация эмпирического риска.....	39
1.2.1.5. Неопределенность.....	41
1.2.1.6. Оценка максимального правдоподобия.....	42
1.2.2. Регрессия.....	43
1.2.2.1. Линейная регрессия.....	44
1.2.2.2. Полиномиальная регрессия.....	45
1.2.2.3. Глубокие нейронные сети.....	46
1.2.3. Переобучение и обобщаемость.....	47
1.2.4. Теорема об отсутствии бесплатных завтраков.....	48
1.3. Обучение без учителя.....	48
1.3.1. Кластеризация.....	49
1.3.2. Обнаружение латентных «факторов изменчивости».....	50
1.3.3. Самостоятельное обучение.....	51
1.3.4. Оценка обучения без учителя.....	52
1.4. Обучение с подкреплением.....	53
1.5. Данные.....	55
1.5.1. Некоторые широко известные наборы изображений.....	55
1.5.1.1. Небольшие наборы изображений.....	55
1.5.1.2. ImageNet.....	56
1.5.2. Некоторые широко известные наборы текстовых данных.....	57
1.5.2.1. Классификация текста.....	58
1.5.2.2. Машинный перевод.....	59
1.5.2.3. Другие задачи типа seq2seq.....	59
1.5.2.4. Языковое моделирование.....	59
1.5.3. Предобработка дискретных входных данных.....	60
1.5.3.1. Унитарное кодирование.....	60
1.5.3.2. Перекрестные произведения признаков.....	60
1.5.4. Предобработка текстовых данных.....	61
1.5.4.1. Модель мешка слов.....	61
1.5.4.2 TF-IDF.....	62

1.5.4.3. Погружения слов.....	63
1.5.4.4. Обработка новых слов	63
1.5.5. Обработка отсутствующих данных	64
1.6. Обсуждение	65
1.6.1. Связь МО с другими дисциплинами	65
1.6.2. Структура книги.....	66
1.6.3. Подводные камни	66

Часть I. ОСНОВАНИЯ.....68

Глава 2. Вероятность: одномерные модели.....69

2.1. Введение.....	69
2.1.1. Что такое вероятность?.....	69
2.1.2. Типы неопределенности.....	70
2.1.3. Вероятность как обобщение логики	70
2.1.3.1. Вероятность события	70
2.1.3.2. Вероятность конъюнкции двух событий	71
2.1.3.3. Вероятность объединения двух событий.....	71
2.1.3.4. Условная вероятность одного события при условии другого	71
2.1.3.5. Независимость событий	72
2.1.3.6. Условная независимость событий	72
2.2. Случайные величины	72
2.2.1. Дискретные случайные величины	72
2.2.2. Непрерывные случайные величины.....	73
2.2.2.1. Функция распределения.....	73
2.2.2.2. Функция плотности распределения	74
2.2.2.3. Квантили.....	75
2.2.3. Множества связанных случайных величин	75
2.2.4. Независимость и условная независимость.....	76
2.2.5. Моменты распределения.....	77
2.2.5.1. Среднее распределения.....	78
2.2.5.2. Дисперсия распределения	78
2.2.5.3. Мода распределения	79
2.2.5.4. Условные моменты.....	80
2.2.6. Ограничения сводных статистик*	81
2.3. Формула Байеса	83
2.3.1. Пример: тестирование на COVID-19.....	84
2.3.2. Пример: парадокс Монти Холла.....	86
2.3.3. Обратные задачи*	88
2.4. Распределение Бернулли и биномиальное распределение.....	89
2.4.1. Определение.....	89
2.4.2. Сигмоидная (логистическая) функция.....	90
2.4.3. Бинарная логистическая регрессия	92
2.5. Категориальное и мультиномиальное распределение	93
2.5.1. Определение.....	93
2.5.2. Функция softmax	94

2.5.3. Многоклассовая логистическая регрессия.....	95
2.5.4. Логарифмирование, суммирование, потенцирование.....	96
2.6. Одномерное гауссово (нормальное) распределение	97
2.6.1. Функция распределения.....	98
2.6.2. Функция плотности вероятности.....	99
2.6.3. Регрессия.....	100
2.6.4. Почему гауссово распределение так широко используется?	101
2.6.5. Дельта-функция Дирака как предельный случай	102
2.7. Другие часто встречающиеся одномерные распределения*	102
2.7.1. Распределение Стьюдента	102
2.7.2. Распределение Коши	104
2.7.3. Распределение Лапласа	105
2.7.4. Бета-распределение.....	105
2.7.5. Гамма-распределение	106
2.7.6. Эмпирическое распределение	107
2.8. Преобразования случайных величин*	108
2.8.1. Дискретный случай	109
2.8.2. Непрерывный случай	109
2.8.3. Обратимые преобразования (биекции)	109
2.8.3.1. Замена переменных: скалярный случай.....	109
2.8.3.2. Замена переменных: многомерный случай.....	110
2.8.4. Моменты линейного преобразования.....	112
2.8.5. Теорема о свертке	113
2.8.6. Центральная предельная теорема.....	115
2.8.7. Аппроксимация Монте-Карло.....	115
2.9. Упражнения.....	116
Глава 3. Вероятность: многомерные модели.....	120
3.1. Совместные распределения нескольких случайных величин.....	120
3.1.1. Ковариация	120
3.1.2. Корреляция	121
3.1.3. Некоррелированные не значит независимые.....	122
3.1.4. Из коррелированности не следует наличие причинно-следственной связи.....	122
3.1.5. Парадокс Симпсона.....	123
3.2. Многомерное гауссово (нормальное) распределение	126
3.2.1. Определение	126
3.2.2. Расстояние Махаланобиса	127
3.2.3. Маргинальные и условные распределения для многомерного нормального распределения*	129
3.2.4. Пример: обусловливание двумерного гауссова распределения.....	130
3.2.5. Пример: подстановка отсутствующих значений*	131
3.3. Линейные гауссовы системы*	132
3.3.1. Формула Байеса для гауссовых распределений	132
3.3.2. Вывод*	133
3.3.3. Пример: вывод неизвестного скаляра.....	134
3.3.4. Пример: вывод неизвестного вектора.....	136

3.3.5. Пример: слияние показаний датчиков.....	137
3.4. Экспоненциальное семейство распределений*	139
3.4.1. Определение.....	139
3.4.2. Пример.....	140
3.4.3. Логарифмическая функция разбиения является производящей функцией кумулянтов.....	141
3.4.4. Вывод максимальной энтропии экспоненциального семейства.....	141
3.5. Смесевые модели.....	142
3.5.1. Модель гауссовой смеси.....	143
3.5.2. Модели бернуллиевой смеси.....	145
3.6. Графовые вероятностные модели*	146
3.6.1. Представление.....	146
3.6.1.1. Пример: оросительная система.....	147
3.6.1.2. Пример: марковская цепь.....	148
3.6.2. Вывод.....	149
3.6.3. Обучение.....	149
3.6.3.1. Блочная нотация.....	150
3.7. Упражнения.....	151
Глава 4. Статистика.....	153
4.1. Введение.....	153
4.2. Оценка максимального правдоподобия (MLE).....	153
4.2.1. Определение.....	154
4.2.2. Обоснование MLE.....	155
4.2.3. Пример: MLE для распределения Бернулли.....	156
4.2.4. Пример: MLE для категориального распределения.....	157
4.2.5. Пример: MLE для одномерного гауссова распределения.....	158
4.2.6. Пример: MLE для многомерного гауссова распределения.....	159
4.2.6.1. MLE среднего.....	159
4.2.6.2. MLE ковариационной матрицы.....	160
4.2.7. Пример: MLE для линейной регрессии.....	161
4.3. Минимизация эмпирического риска (ERM).....	162
4.3.1. Пример: минимизации частоты неправильной классификации.....	163
4.3.2. Суррогатная потеря.....	163
4.4. Другие методы оценивания*	165
4.4.1. Метод моментов.....	165
4.4.1.1. Пример: MOM для одномерного гауссова распределения.....	165
4.4.1.2. Пример: MOM для непрерывного равномерного распределения.....	166
4.4.2. Онлайнное (рекурсивное) оценивание.....	167
4.4.2.1. Пример: рекурсивная MLE среднего гауссова распределения....	167
4.4.2.2. Экспоненциально взвешенное скользящее среднее.....	167
4.5. Регуляризация.....	169
4.5.1. Пример: оценка MAP для распределения Бернулли.....	170
4.5.2. Пример: оценка MAP для многомерного гауссова распределения* ...	171
4.5.2.1. Оценка усадки.....	171
4.5.3. Пример: уменьшение весов.....	172

4.5.4. Подбор регуляризатора с помощью контрольного набора	173
4.5.5. Перекрестная проверка	174
4.5.5.1. Правило одной стандартной ошибки	175
4.5.5.2. Пример: гребневая регрессия	176
4.5.6. Ранняя остановка	176
4.5.7. Больше данных	177
4.6. Байесовские статистики*	178
4.6.1. Сопряженные априорные распределения	179
4.6.2. Бета-биномиальная модель	180
4.6.2.1. Правдоподобие Бернулли	180
4.6.2.2. Биномиальное правдоподобие	180
4.6.2.3. Априорное распределение	181
4.6.2.4. Апостериорное распределение	181
4.6.2.5. Пример	181
4.6.2.6. Апостериорная мода (оценка MAP)	182
4.6.2.7. Апостериорное среднее	183
4.6.2.8. Апостериорная дисперсия	183
4.6.2.9. Апостериорное прогнозное распределение	184
4.6.2.10. Маргинальное правдоподобие	187
4.6.2.11. Смеси сопряженных априорных распределений	187
4.6.3. Дирихле-мультиномиальная модель	189
4.6.3.1. Правдоподобие	189
4.6.3.2. Априорное распределение	189
4.6.3.3. Апостериорное распределение	191
4.6.3.4. Апостериорное прогнозное распределение	192
4.6.3.5. Маргинальное правдоподобие	192
4.6.4. Гауссова-гауссова модель	193
4.6.4.1. Одномерный случай	193
4.6.4.2. Многомерный случай	195
4.6.5. За пределами сопряженных априорных распределений	196
4.6.5.1. Неинформативные априорные распределения	197
4.6.5.2. Иерархические априорные распределения	197
4.6.5.3. Эмпирические априорные распределения	197
4.6.6. Байесовские доверительные интервалы	198
4.6.7. Байесовское машинное обучение	200
4.6.7.1. Подстановочная аппроксимация	201
4.6.7.2. Пример: скалярный вход, бинарный выход	201
4.6.7.3. Пример: бинарный вход, скалярный выход	203
4.6.7.4. Вертикальное масштабирование	205
4.6.8. Вычислительные трудности	205
4.6.8.1. Сеточная аппроксимация	206
4.6.8.2. Квадратичная аппроксимация (Лапласа)	206
4.6.8.3. Вариационная аппроксимация	207
4.6.8.4. Аппроксимация методом Монте-Карло по схеме марковских цепей	208
4.7. Частотная статистика*	208
4.7.1. Выборочное распределение	209

4.7.2. Гауссова аппроксимация выборочного распределения MLE.....	210
4.7.3. Бутстрэпная аппроксимация выборочного распределения любого оценителя.....	211
4.7.3.1. Бутстрэп – апостериорное распределение «для бедных».....	211
4.7.4. Доверительные интервалы.....	212
4.7.5. Предостережения: доверительные интервалы и байесовские доверительные интервалы не одно и то же.....	214
4.7.6. Компромисс между смещением и дисперсией.....	215
4.7.6.1. Смещение оценки.....	215
4.7.6.2. Дисперсия оценки.....	216
4.7.6.3. Компромисс между смещением и дисперсией.....	216
4.7.6.4. Пример: оценка MAP среднего гауссова распределения.....	217
4.7.6.5. Пример: оценка MAP для линейной регрессии.....	218
4.7.6.6. Применение компромисса между смещением и дисперсией для классификации.....	220
4.8. Упражнения.....	220
Глава 5. Теория принятия решений.....	225
5.1. Байесовская теория принятия решений.....	225
5.1.1. Основы.....	225
5.1.2. Проблемы классификации.....	227
5.1.2.1. Бинарная потеря.....	228
5.1.2.2. Классификация с учетом стоимости.....	228
5.1.2.3. Классификация с возможностью отклонения примера.....	229
5.1.3. ROC-кривые.....	230
5.1.3.1. Матрицы неточностей классификации.....	230
5.1.3.2. Обобщение ROC-кривой в виде скаляра.....	233
5.1.3.3. Несбалансированность классов.....	233
5.1.4. Кривые точность–полнота.....	233
5.1.4.1. Вычисление точности и полноты.....	234
5.1.4.2. Обобщение кривых точность–полнота в виде скаляра.....	234
5.1.4.3. F-мера.....	235
5.1.4.4. Несбалансированность классов.....	235
5.1.5. Задачи регрессии.....	236
5.1.5.1. ℓ_2 -потеря.....	236
5.1.5.2. ℓ_1 -потеря.....	237
5.1.5.3. Функция потерь Хьюбера.....	237
5.1.6. Задачи вероятностного предсказания.....	238
5.1.6.1. Расхождение КЛ, перекрестная энтропия и логарифмическая потеря.....	238
5.1.6.2. Правила верной оценки.....	239
5.2. Байесовская проверка гипотез.....	240
5.2.1. Пример: проверка симметричности монеты.....	241
5.2.2. Байесовский выбор модели.....	242
5.2.2.1. Пример: полиномиальная регрессия.....	243
5.2.3. Бритва Оккама.....	244
5.2.4. Связь между перекрестной проверкой и маргинальным правдоподобием.....	246

5.2.5. Информационные критерии.....	246
5.2.5.1. Байесовский информационный критерий (BIC)	247
5.2.5.2. Информационный критерий Акаике	247
5.2.5.3. Минимальная длина описания (MDL).....	248
5.3. Частотная теория принятий решений	248
5.3.1. Вычисление риска оценки.....	248
5.3.1.1. Пример	249
5.3.1.2. Байесовский риск	250
5.3.1.3. Максимальный риск	251
5.3.2. Состоятельные оценки.....	251
5.3.3. Допустимые оценки	252
5.4. Минимизация эмпирического риска.....	253
5.4.1. Эмпирический риск.....	253
5.4.1.1. Ошибка аппроксимации и ошибка оценивания	254
5.4.1.2. Регуляризованный риск.....	255
5.4.2. Структурный риск.....	255
5.4.3. Перекрестная проверка	256
5.4.4. Статистическая теория обучения*	257
5.4.4.1. Нахождение границы ошибки обобщения	257
5.4.4.2. VC-размерность	258
5.5. Частотная проверка гипотез*	258
5.5.1. Критерий отношения правдоподобия.....	259
5.5.1.1. Пример: сравнение гауссовых средних	259
5.5.1.2. Простые и сложные гипотезы.....	260
5.5.2. Проверка значимости нулевой гипотезы	260
5.5.3. p-значения	261
5.5.4. О вреде p-значений.....	261
5.5.5. Почему же не все исповедуют байесовский подход?	264
5.6. Упражнения.....	266
Глава 6. Теория информации.....	268
6.1. Энтропия	268
6.1.1. Энтропия дискретных случайных величин	268
6.1.2. Перекрестная энтропия	271
6.1.3. Совместная энтропия.....	271
6.1.4. Условная энтропия.....	272
6.1.5. Перплексия	273
6.1.6. Дифференциальная энтропия непрерывных случайных величин*	274
6.1.6.1. Пример: энтропия гауссова распределения.....	274
6.1.6.2. Связь с дисперсией.....	275
6.1.6.3. Дискретизация.....	275
6.2. Относительная энтропия (расхождение KL)*	275
6.2.1. Определение.....	276
6.2.2. Интерпретация.....	276
6.2.3. Пример: расхождение КЛ между двумя гауссовыми распределениями.....	276

6.2.4. Неотрицательность расхождения КЛ.....	277
6.2.5. Расхождение КЛ и оценка максимального правдоподобия.....	278
6.2.6. Прямое и обратное расхождение КЛ.....	279
6.3. Взаимная информация*.....	280
6.3.1. Определение.....	280
6.3.2. Интерпретация.....	280
6.3.3. Пример.....	282
6.3.4. Условная взаимная информация.....	282
6.3.5. Взаимная информация как «обобщенный коэффициент корреляции».....	283
6.3.6. Нормированная взаимная информация.....	284
6.3.7. Максимальный коэффициент информации.....	285
6.3.8. Неравенство обработки данных.....	287
6.3.9. Достаточные статистики.....	288
6.3.10. Неравенство Фано*.....	288
6.4. Упражнения.....	289
Глава 7. Линейная алгебра.....	292
7.1. Введение.....	292
7.1.1. Обозначения.....	292
7.1.1.1. Векторы.....	292
7.1.1.2. Матрицы.....	293
7.1.1.3. Тензоры.....	294
7.1.2. Векторные пространства.....	295
7.1.2.1. Сложение векторов и умножение вектора на скаляр.....	295
7.1.2.2. Линейная независимость, линейная оболочка и базисы.....	296
7.1.2.3. Линейные отображения и матрицы.....	296
7.1.2.4. Образ и ядро матрицы.....	297
7.1.2.5. Линейная проекция.....	297
7.1.3. Нормы вектора и матрицы.....	298
7.1.3.1. Нормы вектора.....	298
7.1.3.2. Нормы матрицы.....	299
7.1.4. Свойства матриц.....	300
7.1.4.1. След квадратной матрицы.....	300
7.1.4.2. Определитель квадратной матрицы.....	300
7.1.4.3. Ранг матрицы.....	301
7.1.4.4. Числа обусловленности.....	301
7.1.5. Специальные типы матриц.....	303
7.1.5.1. Диагональная матрица.....	303
7.1.5.2. Треугольные матрицы.....	304
7.1.5.3. Положительно определенные матрицы.....	304
7.1.5.4. Ортогональные матрицы.....	305
7.2. Умножение матриц.....	306
7.2.1. Умножение векторов.....	307
7.2.2. Произведение матрицы на вектор.....	307
7.2.3. Произведение матриц.....	308
7.2.4. Приложение: манипулирование матрицами данных.....	310

7.2.4.1. Суммирование срезов матрицы	310
7.2.4.2. Масштабирование строк и столбцов матрицы.....	311
7.2.4.3. Матрица сумм квадратов и матрица рассеяния	311
7.2.4.4. Матрица Грама	312
7.2.4.5. Матрица расстояний	313
7.2.5. Произведения Кронекера*	313
7.2.6. Суммирование Эйнштейна*	314
7.3. Обращение матриц	315
7.3.1. Обращение квадратной матрицы.....	315
7.3.2. Дополнения Шура*	316
7.3.3. Лемма об обращении матрицы*	317
7.3.4. Лемма об определителе матрицы*	318
7.3.5. Приложение: вывод условных распределений для многомерного гауссова распределения	319
7.4. Спектральное разложение	320
7.4.1. Основные сведения.....	320
7.4.2. Диагонализация	321
7.4.3. Собственные значения и собственные векторы симметричных матриц	322
7.4.3.1. Проверка на положительную определенность.....	322
7.4.4. Геометрия квадратичных форм.....	323
7.4.5. Стандартизация и отбеливание данных.....	323
7.4.6. Степенной метод.....	324
7.4.7. Понижение порядка	326
7.4.8. Собственные векторы оптимизируют квадратичные формы.....	326
7.5. Сингулярное разложение (SVD)	327
7.5.1. Основные сведения.....	327
7.5.2. Связь между сингулярным и спектральным разложением.....	328
7.5.3. Псевдообратная матрица.....	329
7.5.4. SVD для образа и ядра матрицы*	330
7.5.5. Усеченное сингулярное разложение	331
7.6. Другие матричные разложения*	332
7.6.1. LU-разложение	332
7.6.2. QR-разложение	333
7.6.3. Разложение Холецки	334
7.6.3.1. Приложение: выборка из многомерного гауссова распределения	334
7.7. Решение систем линейных уравнений*	335
7.7.1. Решение квадратных систем	336
7.7.2. Решение недоопределенных систем (оценка по наименьшей норме).....	336
7.7.3. Решение переопределенных систем (оценка по методу наименьших квадратов)	338
7.8. Матричное исчисление.....	339
7.8.1. Производные	339
7.8.2. Градиенты	340
7.8.3. Производная по направлению	340

7.8.4. Полная производная*	341
7.8.5. Якобиан	341
7.8.5.1. Умножение якобиана на вектор	342
7.8.5.2. Якобиан композиции	342
7.8.6. Гессиан	342
7.8.7. Градиенты часто встречающихся функций	343
7.8.7.1. Функции, отображающие скаляры в скаляры	343
7.8.7.2. Функции, отображающие векторы в скаляры	343
7.8.7.3. Функции, отображающие матрицы в скаляры	344
7.9. Упражнения	345
Глава 8. Оптимизация	346
8.1. Введение	346
8.1.1. Локальная и глобальная оптимизация	346
8.1.1.1. Условия оптимальности для локальных и глобальных оптимумов	347
8.1.2. Условная и безусловная оптимизация	348
8.1.3. Выпуклая и невыпуклая оптимизация	349
8.1.3.1. Выпуклые множества	349
8.1.3.2. Выпуклые функции	350
8.1.3.3. Характеристика выпуклых функций	351
8.1.3.4. Сильно выпуклые функции	352
8.1.4. Гладкая и негладкая оптимизация	353
8.1.4.1. Субградиенты	354
8.2. Методы первого порядка	355
8.2.1. Направление спуска	356
8.2.2. Размер шага (скорость обучения)	356
8.2.2.1. Постоянный размер шага	356
8.2.2.2. Линейный поиск	358
8.2.3. Скорость сходимости	359
8.2.4. Метод импульса	360
8.2.4.1. Импульс	360
8.2.4.2. Импульс Нестерова	361
8.3. Методы второго порядка	362
8.3.1. Метод Ньютона	362
8.3.2. BFGS и другие квазиньютоновские методы	364
8.3.3. Методы на основе доверительных областей	365
8.4. Стохастический градиентный спуск	366
8.4.1. Приложение к задачам с конечной суммой	367
8.4.2. Пример: СГС для обучения модели линейной регрессии	368
8.4.3. Выбор размера шага (скорости обучения)	369
8.4.4. Итеративное усреднение	371
8.4.5. Уменьшение дисперсии*	372
8.4.5.1. SVRG	372
8.4.5.2. SAGA	373
8.4.5.3. Применение в глубоком обучении	373
8.4.6. Предобусловленный СГС	374

8.4.6.1. AdaGrad	374
8.4.6.2. RMSProp и AdaDelta.....	375
8.4.6.3. Adam	376
8.4.6.4. Проблемы, связанные с адаптивной скоростью обучения	376
8.4.6.5. Недиagonальные матрицы преобусловливания	377
8.5. Условная оптимизация.....	377
8.5.1. Множители Лагранжа.....	378
8.5.1.1. Пример: двумерная квадратичная целевая функция с одним линейным ограничением в виде равенства.....	379
8.5.2. Условия Каруша–Куна–Таккера	380
8.5.3. Линейное программирование	381
8.5.3.1. Симплекс-метод	382
8.5.3.2. Приложения.....	382
8.5.4. Квадратичное программирование.....	382
8.5.4.1. Пример: квадратичная целевая функция в двумерном случае с линейными ограничениями в виде равенств	383
8.5.4.2. Приложения.....	384
8.5.5. Смешанно-целочисленное линейное программирование*	384
8.6. Проксимальный градиентный метод*	384
8.6.1. Спроецированный градиентный спуск.....	385
8.6.2. Проксимальный оператор для регуляризатора по норме ℓ_1	387
8.6.3. Применение проксимального оператора в случае квантования	388
8.6.4. Инкрементные (онлайнные) проксимальные методы	389
8.7. Граничная оптимизация*	389
8.7.1. Общий алгоритм	389
8.7.2. EM-алгоритм	391
8.7.2.1. Нижняя граница.....	392
8.7.2.2. E-шаг	392
8.7.2.3. M-шаг	393
8.7.3. Пример: EM-алгоритм для смеси гауссовых распределений.....	394
8.7.3.1. E-шаг	394
8.7.3.2. M-шаг	394
8.7.3.3. Пример	395
8.7.3.4. Оценка MAP	395
8.7.3.5. Невыпуклость NLL	398
8.8. Оптимизация черного ящика и оптимизация без использования производных.....	399
8.9. Упражнения.....	399

Часть II. ЛИНЕЙНЫЕ МОДЕЛИ..... 400

Глава 9. Линейный дискриминантный анализ..... 401

9.1. Введение	401
9.2. Гауссов дискриминантный анализ	401
9.2.1. Квадратичные решающие границы.....	402
9.2.2. Линейные решающие границы	403

9.2.3. Связь между ЛДА и логистической регрессией.....	403
9.2.4. Обучение модели	405
9.2.4.1. Связанные ковариационные матрицы	406
9.2.4.2. Диагональные ковариационные матрицы	406
9.2.4.3. Оценка MAP.....	406
9.2.5. Классификатор по ближайшему центру.....	407
9.2.6. Линейный дискриминантный анализ Фишера*	407
9.2.6.1. Нахождение оптимального одномерного направления	409
9.2.6.2. Обобщение на большую размерность и несколько классов	411
9.3. Наивные байесовские классификаторы	412
9.3.1. Примеры моделей.....	413
9.3.2. Обучение модели	413
9.3.3. Байесовская интерпретация наивной байесовской модели	415
9.3.4. Связь между наивной байесовской моделью и логистической регрессией.....	416
9.4. Порождающие и дискриминантные классификаторы.....	417
9.4.1. Преимущества дискриминантных классификаторов	417
9.4.2. Преимущества порождающих классификаторов.....	418
9.4.3. Обработка отсутствующих признаков.....	419
9.5. Упражнения.....	419
Глава 10. Логистическая регрессия	420
10.1. Введение	420
10.2. Бинарная логистическая регрессия.....	420
10.2.1. Линейные классификаторы	421
10.2.2. Нелинейные классификаторы	422
10.2.3. Оценка максимального правдоподобия	423
10.2.3.1. Целевая функция	423
10.2.3.2. Оптимизация целевой функции	424
10.2.3.3. Вывод градиента.....	425
10.2.3.4. Вывод гессиана	426
10.2.4. Стохастический градиентный спуск.....	427
10.2.5. Алгоритм перцептрона	427
10.2.6. Метод наименьших квадратов с итеративным пересчетом весов.....	428
10.2.7. Оценка MAP	430
10.2.8. Стандартизация	431
10.3. Мультиномиальная логистическая регрессия	432
10.3.1. Линейные и нелинейные классификаторы	433
10.3.2. Оценка максимального правдоподобия	433
10.3.2.1. Целевая функция	434
10.3.2.2. Оптимизация целевой функции	434
10.3.2.3. Вывод градиента.....	434
10.3.2.4. Вывод гессиана	435
10.3.3. Градиентная оптимизация	436
10.3.4. Граничная оптимизация.....	436
10.3.5. Оценка MAP	438

10.3.6. Классификаторы максимальной энтропии	439
10.3.7. Иерархическая классификация.....	440
10.3.8. Работа с большим числом классов	440
10.3.8.1. Иерархическая softmax-модель.....	441
10.3.8.2. Несбалансированность классов и длинный хвост.....	441
10.4. Робастная логистическая регрессия*	443
10.4.1. Смесевая модель правдоподобия.....	443
10.4.2. Дважды смягченная потеря	444
10.5. Байесовская логистическая регрессия*	447
10.5.1. Аппроксимация Лапласа	447
10.5.2. Аппроксимация апостериорного прогнозного распределения	449
10.5.2.1. Аппроксимация Монте-Карло.....	451
10.5.2.2. Пробит-аппроксимация	451
10.6. Упражнения.....	452
Глава 11. Линейная регрессия	455
11.1. Введение	455
11.2. Линейная регрессия по методу наименьших квадратов	455
11.2.1. Терминология.....	455
11.2.2. Оценивание по методу наименьших квадратов.....	457
11.2.2.1. Обыкновенный метод наименьших квадратов	457
11.2.2.2. Геометрическая интерпретация метода наименьших квадратов	458
11.2.2.3. Алгоритмические проблемы	460
11.2.2.4. Метод взвешенных наименьших квадратов	461
11.2.3. Другие подходы к вычислению MLE	461
11.2.3.1. Нахождение смещения и углового коэффициента по отдельности.....	461
11.2.3.2. Простая линейная регрессия (одномерные входные данные)	462
11.2.3.3. Частная регрессия	462
11.2.3.4. Рекурсивное вычисление MLE.....	462
11.2.3.5. Вывод MLE с порождающей точки зрения	464
11.2.3.6. Вывод MLE для σ^2	465
11.2.4. Измерение степени согласия оценки	465
11.2.4.1. Графики невязок.....	465
11.2.4.2. Точность предсказания и R^2	466
11.3. Гребневая регрессия	467
11.3.1. Вычисление оценки MAP.....	467
11.3.1.1. Решение с использованием QR-разложения.....	468
11.3.1.2. Решение с использованием сингулярного разложения	469
11.3.2. Связь между гребневой регрессией и PCA.....	469
11.3.3. Выбор силы регуляризатора	471
11.4. Регрессия lasso	471
11.4.1. Оценка MAP с априорным распределением Лапласа (ℓ_1 -регуляризация).....	472
11.4.2. Почему ℓ_1 -регуляризация дает разреженные решения?	473

11.4.3. Жесткие и мягкие пороги	474
11.4.4. Путь регуляризации	476
11.4.5. Сравнение методов наименьших квадратов, lasso, гребневой регрессии и выбора подмножеств	478
11.4.6. Согласованность выбора переменных	479
11.4.7. Групповое lasso	481
11.4.7.1. Приложения	481
11.4.7.2. Штрафование по норме ℓ_2	482
11.4.7.3. Штрафование по норме ℓ_∞	482
11.4.7.4. Пример	483
11.4.8. Эластичная сеть (комбинация гребневой регрессии и lasso)	484
11.4.9. Алгоритмы оптимизации	485
11.4.9.1. Покоординатный спуск	485
11.4.9.2. Спроецированный градиентный спуск	486
11.4.9.3. Проксимальный градиентный спуск	486
11.4.9.4. LARS	486
11.5. Регрессионные сплайны*	487
11.5.1. В-сплайны в качестве базисных функций	488
11.5.2. Обучение линейно модели с помощью сплайнового базиса	489
11.5.3. Сглаживающие сплайны	490
11.5.4. Обобщенные аддитивные модели	490
11.6. Робастная линейная регрессия*	491
11.6.1. Правдоподобие Лапласа	491
11.6.1.1. Вычисление MLE методами линейного программирования	492
11.6.2. t-правдоподобие Стьюдента	493
11.6.3. Функция потерь Хьюбера	493
11.6.4. RANSAC	494
11.7. Байесовская линейная регрессия*	494
11.7.1. Априорные распределения	494
11.7.2. Апостериорные распределения	495
11.7.3. Пример	495
11.7.4. Вычисление апостериорного прогнозного распределения	497
11.7.5. Преимущество центрирования	498
11.7.6. Мультиколлинеарность	499
11.7.7. Автоматическое определение релевантности (ARD)*	501
11.8. Упражнения	502
Глава 12. Обобщенные линейные модели*	505
12.1. Введение	505
12.2. Примеры	506
12.2.1. Линейная регрессия	506
12.2.2. Биномиальная регрессия	506
12.2.3. Регрессия Пуассона	507
12.3. GLM с неканоническими функциями связи	508
12.4. Оценка максимального правдоподобия	509
12.5. Рабочий пример: предсказание обращений за страховыми выплатами	510

Часть III. ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ	513
Глава 13. Нейронные сети для структурированных данных	514
13.1. Введение	514
13.2. Многослойные перцептроны (МСП)	516
13.2.1. Задача XOR	516
13.2.2. Дифференцируемые МСП	517
13.2.3. Функции активации	518
13.2.4. Примеры моделей	519
13.2.4.1. МСП для классификации двумерных данных по двум категориям	519
13.2.4.2. МСП для классификации изображений	520
13.2.4.3. МСП для классификации текстов	522
13.2.4.4. МСП для гетероскедастической регрессии	523
13.2.5. Важность глубины	524
13.2.6. Революция глубокого обучения	525
13.2.7. Связи с биологией	526
13.3. Обратное распространение	529
13.3.1. Прямой и обратный режим дифференцирования	530
13.3.2. Дифференцирование в обратном режиме для многослойных перцептронов	531
13.3.3. Произведение вектора на якобиан для типичных слоев	533
13.3.3.1. Слой перекрестной энтропии	533
13.3.3.2. Поэлементная нелинейность	534
13.3.3.3. Линейный слой	535
13.3.3.4. Соберем все вместе	536
13.3.4. Графы вычислений	536
13.4. Обучение нейронных сетей	538
13.4.1. Настройка скорости обучения	539
13.4.2. Исчезающие и взрывные градиенты	539
13.4.3. Функции активации без насыщения	540
13.4.3.1. ReLU	542
13.4.3.2. ReLU без насыщения	542
13.4.3.3. Другие варианты	543
13.4.4. Остаточные связи	544
13.4.5. Инициализация параметров	545
13.4.5.1. Эвристические схемы инициализации	545
13.4.5.2. Инициализации, управляемые данными	546
13.4.6. Параллельное обучение	546
13.5. Регуляризация	548
13.5.1. Ранняя остановка	548
13.5.2. Уменьшение весов	548
13.5.3. Разреженные ГНС	548
13.5.4. Прореживание	549
13.5.5. Байесовские нейронные сети	551
13.5.6. Эффекты регуляризации, порождаемые стохастическим градиентным спуском*	551

13.6. Другие виды сетей прямого распространения*	553
13.6.1. Сети радиально-базисных функций	553
13.6.1.1. RBF-сеть для регрессии	554
13.6.1.2. RBF-сеть для классификации	554
13.6.2. Смесь экспертов	555
13.6.2.1. Смесь линейных экспертов	558
13.6.2.2. Сети на основе смеси моделей разной плотности	558
13.6.2.3. Иерархические смеси экспертов	559
13.7. Упражнения	559

Глава 14. Нейронные сети для изображений 561

14.1. Введение	561
14.2. Наиболее употребительные слои	563
14.2.1. Сверточные слои	563
14.2.1.1. Свертка в одномерном случае	563
14.2.1.2. Свертка в двумерном случае	564
14.2.1.3. Свертка как умножение матрицы на вектор	565
14.2.1.4. Граничные условия и дополнение	566
14.2.1.5. Свертка с шагом	568
14.2.1.6. Несколько входных и выходных каналов	568
14.2.1.7. Свертка 1×1 (поточечная)	569
14.2.2. Пулинговые слои	569
14.2.3. Соберем все вместе	571
14.2.4. Слои нормировки	571
14.2.4.1. Пакетная нормировка	572
14.2.4.2. Другие виды слоя нормировки	573
14.2.4.3. Сети без нормировки	575
14.3. Распространенные архитектуры классификации изображений	575
14.3.1. LeNet	575
14.3.2. AlexNet	577
14.3.3. GoogLeNet	578
14.3.4. ResNet	579
14.3.5. DenseNet	581
14.3.6. Поиск архитектуры нейронной сети	581
14.4. Другие формы свертки*	582
14.4.1. Дырявая свертка	582
14.4.2. Транспонированная свертка	583
14.4.3. Пространственная раздельная свертка	584
14.5. Решение других дискриминантных задач компьютерного зрения с помощью СНС*	585
14.5.1. Аннотирование изображений	586
14.5.2. Определение объектов	586
14.5.3. Сегментация экземпляров	588
14.5.4. Семантическая сегментация	589
14.5.5. Оценивание позы человека	590
14.6. Генерирование изображений посредством инвертирования СНС*	591

14.6.1. Преобразование обученного классификатора в порождающую модель.....	592
14.6.2. Априорные распределения изображений.....	592
14.6.2.1. Гауссово априорное распределения	593
14.6.2.2. Априорное распределение на основе полной вариации	594
14.6.3. Визуализация признаков, обученных с помощью СНС	595
14.6.4. Deep Dream.....	595
14.6.5. Нейронный перенос стиля	597
14.6.5.1. Как это работает	598
14.6.5.2. Ускорение метода	600
Глава 15. Нейронные сети для последовательностей.....	602
15.1. Введение	602
15.2. Рекуррентные нейронные сети (РНС).....	602
15.2.1. Vec2Seq (генерирование последовательностей)	602
15.2.1.1. Модели	603
15.2.1.2. Приложения.....	604
15.2.2. Seq2Vec (классификация последовательностей).....	606
15.2.3. Seq2Seq (трансляция последовательностей)	607
15.2.3.1. Выровненный случай.....	607
15.2.3.2. Невыровненный случай	608
15.2.4. Принуждение со стороны учителя	609
15.2.5. Обратное распространение во времени	610
15.2.6. Исчезающие и взрывные градиенты	612
15.2.7. Вентильная и долгосрочная память	612
15.2.7.1. Управляемые рекуррентные блоки (GRU).....	612
15.2.7.2. Долгая краткосрочная память (LSTM)	613
15.2.8. Лучевой поиск	615
15.3. Одномерные СНС	617
15.3.1. Применение одномерных СНС для классификации последовательностей	618
15.3.2. Применение каузальных одномерных СНС для генерирования последовательностей	618
15.4. Модель внимания.....	620
15.4.1. Механизм внимания как мягкий поиск в словаре	620
15.4.2. Ядерная регрессия как непараметрическое внимание	622
15.4.3. Параметрическое внимание	623
15.4.4. Модель Seq2Seq с вниманием	624
15.4.5. Модель Seq2Vec с вниманием (классификация текста).....	626
15.4.6. Модель Seq+Seq2Vec с вниманием (классификация пар предложений).....	627
15.4.7. Жесткое внимание по сравнению с мягким	629
15.5. Трансформеры.....	629
15.5.1. Самовнимание	630
15.5.2. Многоголовое внимание	632
15.5.3. Позиционное кодирование	632
15.5.4. Соберем все вместе.....	634

15.5.5. Сравнение трансформеров, СНС и РНС.....	636
15.5.6. Применение трансформеров для изображений*.....	636
15.5.7. Другие варианты трансформеров*.....	638
15.6. Эффективные трансформеры*.....	639
15.6.1. Фиксированные необучаемые локализованные паттерны внимания.....	639
15.6.2. Обучаемые паттерны разреженного внимания.....	640
15.6.3. Методы с добавлением памяти и рекуррентные методы.....	640
15.6.4. Низкоранговые и ядерные методы.....	640
15.7. Языковые модели и обучение представлений без учителя.....	643
15.7.1. ELMo.....	643
15.7.2. BERT.....	644
15.7.2.1. Замаскированная языковая модель.....	645
15.7.2.2. Задача предсказания следующего предложения.....	645
15.7.2.3. Дообучение BERT для приложений NLP.....	647
15.7.3. GPT.....	649
15.7.3.1. Приложения GPT.....	649
15.7.4. T5.....	649
15.7.5. Обсуждение.....	650

Часть IV. НЕПАРАМЕТРИЧЕСКИЕ МОДЕЛИ..... 652

Глава 16. Методы на основе эталонов..... 653

16.1. Классификация методом К ближайших соседей (KNN).....	653
16.1.1. Пример.....	654
16.1.2. Проклятие размерности.....	655
16.1.3. Снижение требований к скорости и памяти.....	656
16.1.4. Распознавание открытого множества.....	657
16.1.4.1. Онлайнное обучение, обнаружение посторонних и распознавание открытого множества.....	657
16.1.4.2. Другие задачи открытого мира.....	658
16.2. Обучение метрик.....	658
16.2.1. Линейные и выпуклые методы.....	659
16.2.1.1. Метод ближайших соседей с большим зазором.....	659
16.2.1.2. Анализ компонентов соседства.....	660
16.2.1.3. Анализ латентных совпадений.....	660
16.2.2. Глубокое обучение метрики.....	661
16.2.3. Потери классификации.....	662
16.2.4. Потери ранжирования.....	662
16.2.4.1. Попарная (сопоставительная) потеря и сиамские сети.....	663
16.2.4.2. Триpletная потеря.....	663
16.2.4.3. N-парная потеря.....	664
16.2.5. Ускорение оптимизации потери ранжирования.....	665
16.2.5.1. Методы на основе расширения.....	665
16.2.5.2. Методы на основе представителей.....	665
16.2.5.3. Оптимизация верхней границы.....	666

16.2.6. Другие приемы глубокого обучения метрики.....	668
16.3. Ядерные оценки плотности.....	669
16.3.1. Ядра плотности	669
16.3.2. Оконная оценка плотности Парзена	670
16.3.3. Как выбирать полосу пропускания	672
16.3.4. От KDE к KNN-классификации	672
16.3.5. Ядерная регрессия	673
16.3.5.1. Оценка среднего Надарая–Ватсона.....	673
16.3.5.2. Оценка дисперсии.....	675
16.3.5.3. Локально взвешенная регрессия.....	675
Глава 17. Ядерные методы*	676
17.1. Ядра Мерсера.....	676
17.1.1. Теорема Мерсера	678
17.1.2. Некоторые популярные ядра Мерсера.....	678
17.1.2.1. Стационарные ядра для вещественных векторов	678
17.1.2.2. Создание новых ядер из существующих.....	681
17.1.2.3. Комбинирование ядер с помощью сложения и умножения	682
17.1.2.4. Ядра для структурированных входов	683
17.2. Гауссовы процессы	683
17.2.1. Незашумленные наблюдения.....	684
17.2.2. Зашумленные наблюдения.....	685
17.2.3. Сравнение с ядерной регрессией	686
17.2.4. Пространство весов и пространство функций.....	687
17.2.5. Численные проблемы	688
17.2.6. Оценивание параметров ядра.....	688
17.2.6.1. Эмпирическая байесовская оценка	689
17.2.6.2. Байесовский вывод.....	691
17.2.7. Применение гауссовых процессов для классификации	692
17.2.8. Связи с глубоким обучением.....	694
17.2.9. Масштабирование ГП на большие наборы данных	694
17.2.9.1. Разреженные аппроксимации	694
17.2.9.2. Распараллеливание с использованием структуры ядерной матрицы.....	694
17.2.9.3. Аппроксимация случайными признаками.....	695
17.3. Метод опорных векторов	696
17.3.1. Классификаторы с широким зазором.....	697
17.3.2. Двойственная задача	699
17.3.3. Классификаторы с мягким зазором	701
17.3.4. Ядерный трюк.....	702
17.3.5. Преобразование выходов SVM в вероятности.....	703
17.3.6. Связь с логистической регрессией	704
17.3.7. Многоклассовая классификация с применением SVM.....	705
17.3.8. Как выбирать регуляризатор C	706
17.3.9. Ядерная гребневая регрессия.....	707
17.3.10. Применение SVM для регрессии.....	708
17.4. Метод разреженных векторов	711

17.4.1. Метод релевантных векторов.....	711
17.4.2. Сравнение разреженных и плотных ядерных методов	711
17.5. Упражнения	715

Глава 18. Деревья, леса, бэггинг и бустинг 716

18.1. Деревья классификации и регрессии.....	716
18.1.1. Определение модели.....	716
18.1.2. Обучение модели	717
18.1.3. Регуляризация	719
18.1.4. Обработка отсутствующих входных признаков	720
18.1.5. Плюсы и минусы	720
18.2. Ансамблевое обучение	721
18.2.1. Стековое обобщение	722
18.2.2. Ансамблевое обучение не то же, что байесовское усреднение моделей	722
18.3. Бэггинг	723
18.4. Случайные леса.....	724
18.5. Бустинг.....	725
18.5.1. Прямое поэтапное аддитивное моделирование.....	726
18.5.2. Квадратичная потеря и бустинг наименьших квадратов.....	727
18.5.3. Экспоненциальная потеря и AdaBoost	727
18.5.4. LogitBoost	731
18.5.5. Градиентный бустинг	732
18.5.5.1. Градиентный бустинг деревьев.....	734
18.5.5.2. XGBoost	734
18.6. Интерпретация ансамблей деревьев	736
18.6.1. Важность признаков.....	736
18.6.2. Графики частичной зависимости.....	738

Часть V. ЗА ПРЕДЕЛАМИ ОБУЧЕНИЯ С УЧИТЕЛЕМ 739

Глава 19. Обучение при меньшем числе помеченных примеров 740

19.1. Приращение данных.....	740
19.1.1. Примеры.....	740
19.1.2. Теоретическое обоснование.....	741
19.2. Перенос обучения	742
19.2.1. Дообучение	742
19.2.2. Адаптеры.....	744
19.2.3. Предобучение с учителем.....	745
19.2.4. Предобучение без учителя (самостоятельное обучение)	746
19.2.4.1. Задачи подстановки.....	747
19.2.4.2. Замещающие задачи.....	748
19.2.4.3. Сопоставительные задачи.....	748
19.2.4.4. SimCLR.....	748
19.2.4.5. CLIP	751

19.2.5. Адаптация домена	752
19.3. Обучение с частичным привлечением учителя	753
19.3.1. Самообучение и псевдопометка	754
19.3.2. Минимизация энтропии.....	755
19.3.2.1. Кластерное допущение.....	756
19.3.2.2. Взаимная информация между входом и выходом.....	757
19.3.3. Совместное обучение	758
19.3.4. Распространение меток на графах.....	759
19.3.5. Регуляризация по согласованности	760
19.3.6. Глубокие порождающие модели*	762
19.3.6.1. Вариационные автокодировщики	763
19.3.6.2. Порождающие состязательные сети.....	765
19.3.6.3. Нормализующие потоки	766
19.3.7. Сочетание самостоятельного обучения и обучения с частичным привлечением учителя	767
19.4. Активное обучение	768
19.4.1. Подход на основе теории принятия решений.....	769
19.4.2. Теоретико-информационный подход	769
19.4.3. Пакетное активное обучение	770
19.5. Метаобучение	770
19.5.1. Метаобучение, не зависящее от модели (MAML).....	771
19.6. Обучение на малом числе примеров	772
19.6.1. Сопоставляющие сети	773
19.7. Обучение со слабым учителем	774
19.8. Упражнения.....	775
Глава 20. Понижение размерности	776
20.1. Метод главных компонент	776
20.1.1. Примеры.....	777
20.1.2. Вывод алгоритма	779
20.1.2.1. Базовый случай	779
20.1.2.2. Оптимальный вектор весов максимизирует дисперсию спроецированных данных	780
20.1.2.3. Шаг индукции	781
20.1.3. Вычислительные трудности	782
20.1.3.1. Ковариационная матрица и корреляционная матрица	782
20.1.3.2. Работа с данными высокой размерности	783
20.1.3.3. Вычисление PCA с использованием SVD	783
20.1.4. Выбор числа латентных измерений	784
20.1.4.1. Ошибка реконструкции.....	784
20.1.4.2. Графики каменистой осыпи	785
20.1.4.3. Правдоподобие профиля.....	785
20.2. Факторный анализ*	787
20.2.1. Порождающая модель.....	787
20.2.2. Вероятностный PCA.....	789
20.2.3. EM-алгоритм для ФА/PPCA	790
20.2.3.1. EM-алгоритм для ФА.....	791

20.2.3.2. EM-алгоритм для (P)PCA	791
20.2.3.3. Преимущества	792
20.2.4. Неидентифицируемость параметров	794
20.2.5. Нелинейный факторный анализ	795
20.2.6. Смеси факторных анализаторов	795
20.2.7. Факторный анализ экспоненциального семейства	797
20.2.7.1. Пример: бинарный PCA	798
20.2.7.2. Пример: категориальный PCA	798
20.2.8. Модели факторного анализа для парных данных	799
20.2.8.1. PCA с учителем	799
20.2.8.2. Метод частичных наименьших квадратов	800
20.2.8.3. Канонический корреляционный анализ	801
20.3. Автокодировщики	802
20.3.1. Автокодировщики с сужением	802
20.3.2. Шумоподавляющие автокодировщики	804
20.3.3. Сжимающие автокодировщики	806
20.3.4. Разреженные автокодировщики	806
20.3.5. Вариационные автокодировщики	808
20.3.5.1. Обучение VAE	809
20.3.5.2. Перепараметризация	809
20.3.5.3. Сравнение VAE с автокодировщиками	811
20.4. Обучение многообразий*	813
20.4.1. Что такое многообразие?	813
20.4.2. Гипотеза многообразия	814
20.4.3. Подходы к обучению многообразий	815
20.4.4. Многомерное шкалирование	816
20.4.4.1. Классическое ММШ	816
20.4.4.2. Метрическое ММШ	817
20.4.4.3. Неметрическое ММШ	818
20.4.4.4. Отображение Саммона	818
20.4.5. Isomap	819
20.4.6. Ядерный PCA	820
20.4.7. Максимальное раскрытие дисперсии	822
20.4.8. Локально линейное погружение	823
20.4.9. Лапласовы собственные отображения	824
20.4.9.1. Использование собственных векторов лапласиана графа для вычисления погружений	824
20.4.9.2. Что такое лапласиан графа?	825
20.4.10. t-SNE	827
20.4.10.1. Стохастическое погружение соседей	827
20.4.10.2. Симметричное SNE	829
20.4.10.3. SNE с t-распределением	829
20.4.10.4. Выбор линейного масштаба	830
20.4.10.5. Вычислительные проблемы	831
20.4.10.6. UMAP	831
20.5. Погружения слов	832
20.5.1. Латентно-семантический анализ и индексирование	832

20.5.1.1. Латентно-семантическое индексирование	832
20.5.1.2. Латентно-семантический анализ	833
20.5.1.3. Поточечная взаимная информация	834
20.5.2. Word2vec	835
20.5.2.1. Модель Word2vec CBOW	835
20.5.2.2. Скип-граммная модель Word2vec	835
20.5.2.3. Отрицательная выборка	836
20.5.3. GloVe	837
20.5.4. Аналогичные слова	838
20.5.5. Модель погружений слов RAND-WALK	839
20.5.6. Контекстуальные погружения слов	840
20.6. Упражнения	840
Глава 21. Кластеризация	843
21.1. Введение	843
21.1.1. Оценивание выхода методов кластеризации	843
21.1.1.1. Чистота	844
21.1.1.2. Индекс Рэнда	844
21.1.1.3. Взаимная информация	845
21.2. Иерархическая агломеративная кластеризация	846
21.2.1. Алгоритм	847
21.2.1.1. Одиночная связь	848
21.2.1.2. Полная связь	848
21.2.1.3. Средняя связь	849
21.2.2. Пример	849
21.2.3. Расширения	850
21.3. Кластеризация методом К средних	851
21.3.1. Алгоритм	851
21.3.2. Примеры	852
21.3.2.1. Кластеризация точек на плоскости	852
21.3.2.2. Кластеризация временных рядов экспрессии генов дрожжей	852
21.3.3. Векторное квантование	853
21.3.4. Алгоритм K-means++	854
21.3.5. Алгоритм K медоидов	855
21.3.6. Способы ускорения	856
21.3.7. Выбор числа кластеров K	857
21.3.7.1. Минимизация искажения	857
21.3.7.2. Максимизация маргинального правдоподобия	857
21.3.7.3. Силуэтный коэффициент	858
21.3.7.4. Инкрементное увеличение количества компонент смеси	860
21.3.7.5. Методы разреженного оценивания	860
21.4. Кластеризация с помощью смесевых моделей	860
21.4.1. Смеси гауссовых распределений	860
21.4.1.1. Метод К средних – частный случай EM-алгоритма	861
21.4.1.2. Неидентифицируемость и переключение метки	861
21.4.1.3. Байесовский выбор модели	864

21.4.2. Смеси распределений Бернулли.....	865
21.5. Спектральная кластеризация*	865
21.5.1. Нормализованные разрезы.....	866
21.5.2. Собственные векторы лапласиана графа кодируют кластеризацию	866
21.5.3. Пример	867
21.5.4. Связь с другими методами	868
21.5.4.1. Связь с kPCA	868
21.5.4.2. Связь с анализом случайного блуждания	868
21.6. Бикластеризация*	869
21.6.1. Базовая бикластеризация.....	869
21.6.2. Модели вложенного разбиения (Crosscat)	870
Глава 22. Рекомендательные системы	873
22.1. Явная обратная связь	873
22.1.1. Наборы данных	874
22.1.2. Коллаборативная фильтрация	874
22.1.3. Матричная факторизация	875
22.1.3.1. Вероятностная матричная факторизация	876
22.1.3.2. Пример: Netflix	876
22.1.3.3. Пример: MovieLens.....	877
22.1.4. Автокодировщики	878
22.2. Неявная обратная связь	879
22.2.1. Байесовское персонализированное ранжирование	880
22.2.2. Машины факторизации	881
22.2.3. Нейронная матричная факторизация	882
22.3. Использование побочной информации	882
22.4. Компромисс между исследованием и использованием.....	884
Глава 23. Погружения графов*	885
23.1. Введение	885
23.2. Погружение графа как задача о кодировщике и декодере	887
23.3. Поверхностные погружения графов	889
23.3.1. Обучение погружений без учителя	889
23.3.2. На основе расстояния: евклидовы методы	890
23.3.3. На основе расстояния: неевклидовы методы.....	890
23.3.4. На основе внешнего произведения: методы матричной факторизации.....	891
23.3.5. На основе внешнего произведения: скип-граммные методы.....	892
23.3.6. Обучение погружений с учителем	894
23.3.6.1. Распространение меток.....	894
23.4. Графовые нейронные сети.....	895
23.4.1. Графовые нейронные сети передачи сообщений.....	895
23.4.2. Спектральные свертки графов.....	897
23.4.3. Пространственные свертки графов	897
23.4.3.1. Выборочные пространственные методы.....	898

23.4.3.2. Пространственные методы на основе механизма внимания	898
23.4.3.3. Геометрические пространственные методы.....	899
23.4.4. Неевклидовы графовые свертки	899
23.5. Глубокие погружения графов	900
23.5.1. Обучение погружений без учителя.....	900
23.5.1.1. Структурное погружение с помощью глубокой сети	900
23.5.1.2. Вариационные графовые автокодировщики	901
23.5.1.3. Итеративное порождающее моделирование графов (Graphite).....	902
23.5.1.4. Методы на основе сопоставительных потерь	902
23.5.2. Обучение погружений с частичным привлечением учителя	903
23.5.2.1. SemiEmb	903
23.5.2.2. Planetoid.....	903
23.6. Приложения	904
23.6.1. Приложения без учителя	904
23.6.1.1. Реконструкция графа	904
23.6.1.2. Предсказание связей.....	905
23.6.1.3. Кластеризация	906
23.6.1.4. Визуализация	906
23.6.2. Приложения с учителем.....	907
23.6.2.1. Классификация вершин	907
23.6.2.2. Классификация графов.....	907

Приложение А. Обозначения.....

A.1. Введение	909
A.2. Общепринятые математические символы.....	909
A.3. Функции	910
A.3.1. Функции с одним аргументом	910
A.3.2. Функции двух аргументов	910
A.3.3. Функции более двух аргументов.....	911
A.4. Линейная алгебра	911
A.4.1. Общие обозначения.....	911
A.4.2. Векторы.....	911
A.4.3. Матрицы	912
A.4.4. Матричное исчисление	912
A.5. Оптимизация	913
A.6. Вероятность	913
A.7. Теория информации.....	914
A.8. Статистика и машинное обучение.....	915
A.8.1. Обучение с учителем	915
A.8.2. Обучение без учителя и порождающие модели	915
A.8.3. Байесовский вывод	916
A.9. Аббревиатуры.....	916

Библиография

Предметный указатель.....

Предисловие

В 2012 году я опубликовал 1200-страничную книгу под названием «Machine Learning: A Probabilistic Perspective», в которой сложившаяся на тот момент дисциплина машинного обучения (МО) достаточно полно рассматривалась сквозь объединяющую призму вероятностного моделирования. Книга была хорошо принята и получила премию де Грута (<https://bayesian.org/project/degroot-prize/>) в 2013 году.

2012-й также принято считать началом «революции глубокого обучения». Термином «глубокое обучение» называют раздел МО, основанный на нейронных сетях с большим количеством слоев (отсюда и слово «глубокий»). Хотя базовая технология к тому времени существовала уже много лет, именно в 2012 году в работе [KSH12] глубокие нейронные сети (ГНС) выиграли конкурс по классификации изображений ImageNet с таким отрывом, что это привлекло внимание профессионального сообщества. Примерно в то же время был достигнут прогресс в таких трудных задачах, как распознавание речи (см., например, [Cir+10; Cir+11; Hin+12]). Эти прорывы стали возможными благодаря достижениям в развитии оборудования (в частности, перепрофилированию быстрых графических процессоров, GPU, с видеоигр на МО), технологиях сбора данных (в частности, применению краудсорсинговых инструментов типа «Механического турка» от Amazon для сбора больших размеченных наборов данных, как в ImageNet), а также различным новым алгоритмическим идеям; некоторые из них мы рассмотрим в этой книге.

Начиная с 2012 года, область глубокого обучения развивалась лавинообразно, новые результаты появлялись со все возрастающей скоростью. Столь же лавинообразно рос и интерес к этой области, подогреваемый коммерческим успехом технологии. Поэтому в 2018 году я решил написать второе издание своей книги и попытаться подвести какой-то итог.

К марту 2020 года черновой вариант второго издания разросся до 1600 страниц, а еще много тем оставались неосвещенными. В результате издательство MIT Press порекомендовало мне разбить книгу на два тома. Но тут разразилась пандемия COVID-19. Я решил на время отвлечься от написания книги и поучаствовать в разработке алгоритма оценки рисков для разрабатываемого Google приложения, уведомляющего о подверженности заражению [MKS21], а также в других проектах, связанных с прогнозированием [Wah+21]. Однако к осени 2020 года я решил вернуться к работе над книгой.

Чтобы наверстать упущенное, я попросил нескольких коллег помочь мне в написании различных разделов (см. благодарности ниже). В результате появились две новые книги «Вероятностное машинное обучение: введение», которую вы сейчас читаете, и «Вероятностное машинное обучение: дополнительные вопросы», которая является ее продолжением [Mur22]. Надеюсь, что в совокупности они дают довольно полное представление о состоянии дел в МО в 2021 году – сквозь ту же объединяющую призму вероятностного

моделирования и байесовской теории принятия решений, которая использовалась в книге 2012 года.

Почти весь материал первого издания сохранен, но теперь он более-менее равномерно распределен между двумя новыми книгами. Кроме того, в каждую книгу включено много дополнительных тем из области глубокого обучения и других разделов теории, например порождающие модели, вариационный вывод и обучение с подкреплением.

Чтобы сделать эту вводную книгу более замкнутой и полезной студентам, я добавил ряд сведений по базовым дисциплинам, в частности оптимизации и линейной алгебре, которых за скудостью места не было в издании 2012 года.

Материал повышенной сложности, который можно опустить в курсе вводного уровня, помечен звездочкой * в названии раздела или главы.

В конце некоторых глав имеются упражнения. Решения упражнений можно найти в сети. Дополнительные учебные материалы (например, рисунки и слайды) см. на сайте книги [probml.ai](https://probml.github.io/pml-book/) (<https://probml.github.io/pml-book/>).

Еще одно существенное изменение – использование Python вместо Matlab во всех примерах программ. (В будущем мы, возможно, создадим версию кода на Julia.) В новом коде используются такие стандартные Python-библиотеки, как NumPy (<https://numpy.org/>), Scikit-learn (<https://scikit-learn.org/stable/>), JAX (<https://github.com/google/jax>), PyTorch (<https://pytorch.org/>), TensorFlow (<https://www.tensorflow.org/>), PyMC3 (<https://docs.pymc.io/en/v3/>) и др. О том, как использовать код, см. на странице по адресу <https://probml.github.io/pml-book/>.

Благодарности

Я выражаю благодарность следующим людям, помогавшим мне при написании книги:

- Зико Колтеру (Zico Kolter), помогавшему писать части главы 7 («Линейная алгебра»);
- Фредерику Кунстнеру (Frederik Kunstner), Си И Менгу (Si Yi Meng), Аарону Мишкину (Aaron Mishkin), Шарану Васвани (Sharan Vaswani) и Марку Шмидту (Mark Schmidt), помогавшим писать части главы 8 («Оптимизация»);
- Матью Блонделю (Mathieu Blondel), помогавшему писать раздел 13.3 («Обратное распространение»);
- Кшиштофу Хоромански (Krzysztof Choromanski), помогавшему писать раздел 15.6 («Эффективные преобразователи»*);
- Колину Раффелю (Colin Raffel), помогавшему писать раздел 19.2 («Перенос обучения») и раздел 19.3 («Обучение с частичным привлечением учителя»);
- Брайану Пероцци (Bryan Perozzi), Сами Абу Эль Хайджа (Sami Abu-El-Haija) и Инес Чами (Ines Chami), помогавшим писать главу 23 («Погружения графов»*);
- Джону Фэнсу (John Fearn) и Питеру Черно (Peter Cern) за внимательную вычитку книги;

- многочисленных членов сообщества github за поиск опечаток и других ошибок (см. перечень таковых по адресу <https://github.com/probml/pml-book/issues?q=is:issue>);
- четырем анонимным рецензентам, выбранным MIT Press;
- Махмуду Солиману (Mahmoud Soliman), который написал весь код, связующий latex, colab, github и пр., и поведал мне о GCP и TPU;
- всей когорте студентов, работавших над кодом книги на Google Summer of Code 2021 года; это Алейна Кара (Aleyna Kara), Срикаар Джилугу (Srikar Jilugu), Дришти Патель (Drishti Patel), Минь Лиан Анг (Ming Liang Ang), Жерардо Дюран-Мартен (Gerardo Durán-Martín) (см. перечень их трудов по адресу <https://probml.github.io/pml-book/gsoc2021.html>);
- многочисленным членам сообщества github за предложенный ими код (см. <https://github.com/probml/pyprobml#acknowledgements>);
- авторам работ [Zha+20], [Gér17] и [Mar18], разрешившим мне использовать или модифицировать части открытого исходного кода, включенного в их замечательные книги;
- моему начальнику в Google, Дугу Эку (Doug Eck), который позволил мне тратить принадлежащее компании время на написание этой книги;
- своей жене Маргарет, позволившей мне тратить принадлежащее семье время на эту книгу.

Об обложке

На обложке изображена нейронная сеть (глава 13), которая используется для отнесения рукописной цифры x к одному из десяти классов меток $y \in \{0, 1, \dots, 9\}$. Гистограмма справа – вывод модели, соответствующий условному распределению вероятности $p(y|x)$.

Кэвин Патрик Мэрфи
Паль-Альто, Калифорния
август 2021

Глава 1

Введение

1.1. Что такое машинное обучение?

Популярное определение **машинного обучения**, или **МО**, принадлежащее Тому Митчеллу [Mit97], звучит следующим образом:

Говорят, что компьютерная программа обучается на опыте E относительно некоторого класса задач T и меры качества P , если ее качество на задачах, принадлежащих T , измеренное в соответствии с P , улучшается с увеличением опыта E .

Таким образом, существует много видов машинного обучения, зависящих от природы задачи T , решению которой мы хотим обучить систему, природы меры качества P , используемой для оценки работы системы, и природы обучающего сигнала, или опыта E , на котором обучается система.

В этой книге мы будем рассматривать наиболее распространенные типы МО, но с **вероятностной точки зрения**. Грубо говоря, это означает, что все неизвестные переменные (например, предсказания будущего значения некоторой величины, скажем температуры на завтра или параметров некоторой модели) рассматриваются как **случайные величины**, имеющие **распределение вероятностей**, которое описывает взвешенное множество возможных значений переменной (см. краткое введение в основы теории вероятностей в главе 2).

Есть две основные причины, чтобы отдать предпочтение вероятностному подходу. Во-первых, он оптимален для **принятия решений в условиях неопределенности**, это будет объяснено в разделе 5.1. Во-вторых, вероятностное моделирование – язык, используемый в большинстве других областей науки и техники, так что мы получаем единую систему понятий. Шакир Мохамед (научный сотрудник компании DeepMind)¹ говорил:

Почти все машинное обучение можно представить в вероятностных терминах, так что вероятностный подход является фундаментальным. Такое представление, конечно, не единственное. Но именно оно позволяет

¹ Источник: слайд 2 на странице <https://bit.ly/3пуHyPn>.

связать машинное обучение с другими областями вычислительных наук, будь то стохастическая оптимизация, теория управления, исследование операций, эконометрика, теория информации, статистическая физика или биостатистика. Уже по этой причине умение рассуждать в вероятностных терминах обязательно.

1.2. ОБУЧЕНИЕ С УЧИТЕЛЕМ

Самая распространенная форма МО – **обучение с учителем**. В этом случае задача T заключается в том, чтобы обучиться отображению f множества входов $x \in \mathcal{X}$ в множество выходов $y \in \mathcal{Y}$. Входы x называются также **признаками**, **ковариатами** или **предикторами**; часто это числовой вектор фиксированной размерности, например рост и вес человека или пиксели изображения. В таком случае $\mathcal{X} = \mathbb{R}^D$, где D – размерность вектора (т. е. количество входных признаков). Выход y называется также **меткой**, **целью** или **откликом**¹. Опыт E задается в виде множества N пар вход–выход $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, которое называется **обучающим набором** (а N – **размером выборки**). Мера качества P зависит от типа предсказываемого выхода, мы обсудим это ниже.

1.2.1. Классификация

В задачах **классификации** пространство выходов \mathcal{C} неупорядочено, а метки называются **классами**, $\mathcal{Y} = \{1, 2, \dots, C\}$. Проблема предсказания метки класса заданного входа называется также **распознаванием образов**. (Если имеется всего два класса, часто обозначаемых $y \in \{0, 1\}$ или $y \in \{-1, +1\}$, то говорят о **бинарной классификации**.)

1.2.1.1. Пример: классификация ирисов

В качестве примера рассмотрим проблему классификации ирисов с отношением к трем видам: щетинистый (Setosa), разноцветный (Versicolor) и виргинский (Virginica). На рис. 1.1 показано по одному примеру из каждого класса.

В задаче **классификации изображений** пространством входов \mathcal{X} является множество изображений, имеющее очень высокую размерность: для цветного изображения с $C = 3$ каналами (например, в формате RGB) и $D_1 \times D_2$ пикселями имеем $\mathcal{X} = \mathbb{R}^D$, где $D = C \times D_1 \times D_2$. (На практике яркость пикселя представлена целым числом, обычно в диапазоне $\{0, 1, \dots, 255\}$, но для простоты обозначений мы предполагаем, что входы – вещественные числа.) Обучиться отображению $f: \mathcal{X} \rightarrow \mathcal{Y}$ изображений в метки очень трудно, как легко видеть из рис. 1.2. Однако эту задачу все же можно решить с помощью

¹ Иногда (например, в Python пакете statsmodels [[https://www.statsmodels.org/devel/](https://www.statsmodels.org/devel/endog_exog.html) endog_exog.html]) входы x называются **экзогенными переменными**, а выходы y – **эндогенными переменными**.

функций специального вида, например **сверточной нейронной сети** (СНС) (convolutional neural network – CNN), которую мы будем обсуждать в разделе 14.1.

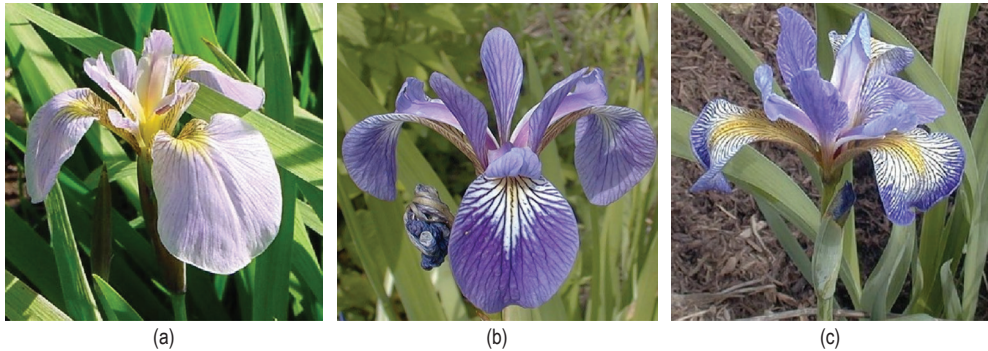


Рис. 1.1 ❖ Три типа ирисов: щетинистый (Setosa), разноцветный (Versicolor) и виргинский (Virginica). Печатается с разрешения Денниса Крэмба и компании SIGMA

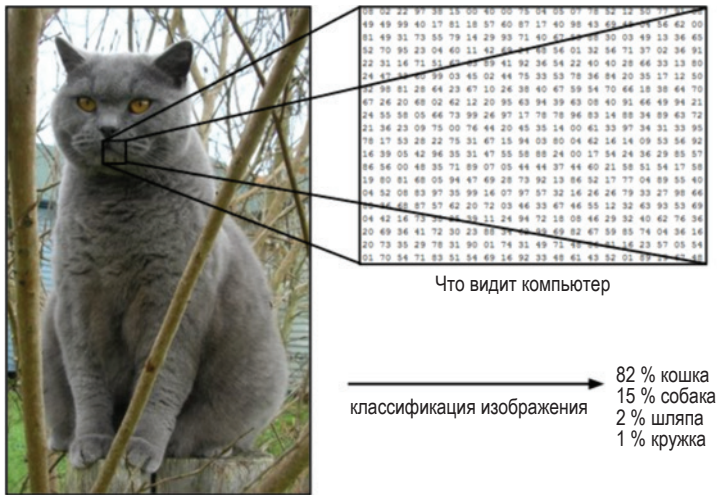


Рис. 1.2 ❖ Иллюстрация к проблеме классификации изображений. Взято с сайта <https://cs231n.github.io/>. Печатается с разрешения Андрея Карпатого

К счастью для нас, ботаники уже выделили 4 простых, но весьма информативных числовых признака – длину чашелистика, ширину чашелистика, длину лепестка и ширину лепестка, – с помощью которых можно различить три вида ирисов. В этом разделе мы для простоты будем использовать именно такое пространство входов куда более низкой размерности: $\mathcal{X} = \mathbb{R}^4$. **Набор данных Iris** представляет собой совокупность 150 помеченных примеров ирисов, по 50 каждого вида, описываемых этими четырьмя признаками. Он широко ис-

пользуется в качестве примера, потому что невелик и прост для понимания. (Далее в книге нам встретятся наборы данных побольше и посложнее.)

Небольшие наборы признаков часто хранят в виде **матрицы плана** размера $N \times D$, в которой строки представляют примеры, в столбцы – признаки, см. пример в табл. 1.1¹.

Таблица 1.1. Подмножество матрицы плана для ирисов

Индекс	sl	sw	pl	pw	Метка
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
	...				
50	7.0	3.2	4.7	1.4	Versicolor
	...				
149	5.9	3.0	5.1	1.8	Virginica

Признаки: длина чашелистика (sl), ширина чашелистика (sw), длина лепестка (pl), ширина лепестка (pw). В каждом классе по 50 примеров.

Набор данных Iris – пример **табличных данных**. Если размер входных данных переменный (например, последовательности слов или социальные сети), в отличие от векторов фиксированной длины, то данные обычно хранятся в формате, отличном от матрицы плана. Однако такие данные часто преобразуются в представление с признаками фиксированного размера (этот процесс называется **фичеризацией**), т. е. неявно создается матрица плана для последующей обработки. Пример такого рода мы приведем в разделе 1.5.4.1, когда будем обсуждать «представление» последовательностей в виде «мешка слов».

1.2.1.2. Разведочный анализ данных

Прежде чем применять к какой-то задаче МО, полезно провести **разведочный анализ данных** и выяснить, есть ли очевидные закономерности (их наличие может подсказать, какой метод выбрать) или очевидные проблемы (например, зашумленные метки или выбросы).

Для табличных данных с небольшим числом признаков часто строят **график парных отношений**, на котором панель (i, j) содержит диаграмму рассеяния величин i и j , а диагональные элементы (i, i) показывают маргинальную плотность величины i ; дополнительно все графики могут быть окрашены цветом, кодирующим метку класса, см. пример на рис. 1.3.

Для данных более высокой размерности зачастую сначала выполняют **понижение размерности**, а затем визуализируют данные в двух или трех измерениях. Методы понижения размерности мы будем обсуждать в главе 20.

¹ В этой конкретной матрице плана $N = 150$ строк и $D = 4$ столбцов, т. е. она высокая и узкая, поскольку $N \gg D$. Встречаются также наборы данных (например, в геномике), в которых признаков больше, чем примеров, т. е. $D \gg N$; для них матрица плана низкая и широкая. Термин **«большие данные»** обычно означает, что N велико, а термин **«широкие данные»** – что D велико (по сравнению с N).

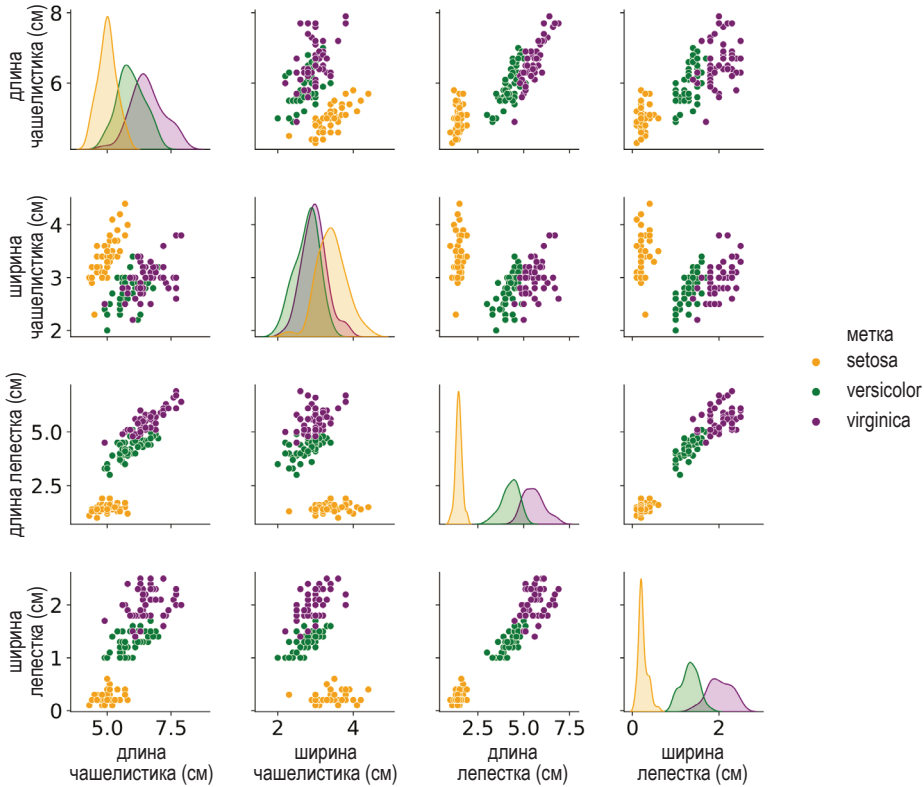


Рис. 1.3 ❖ Визуализация набора данных Iris в виде попарной диаграммы рассеяния. На главной диагонали приведены графики маргинального распределения¹ каждого признака для каждого класса. Элементы вне диагонали содержат диаграммы рассеяния всех возможных пар признаков. Построено программой по адресу figures.problml.ai/book1/1.3

1.2.1.3. Обучение классификатора

Из рис. 1.3 видно, что класс Setosa довольно легко отличить от двух остальных. Например, можно создать такое **решающее правило**:

$$f(x, \theta) = \begin{cases} \text{Setosa, если длина лепестка} < 2.45 \\ \text{Versicolor или Virginica в противном случае} \end{cases} \quad (1.1)$$

Это очень простой пример классификатора; мы разбили пространство входов на две области, разделенные одномерной **решающей границей** $x_{\text{длина лепестка}} = 2.45$. Точки слева от этой границы классифицируются как Setosa, а точки справа – как Versicolor или Virginica.

¹ Ранее в русскоязычной литературе термин *marginal distribution* переводился как «частное распределение», сейчас более употребителен перевод «маргинальное распределение». – Прим. перев.

Мы видим, что это правило абсолютно точно классифицирует примеры Setosa, но не примеры Virginica и Versicolor. Для улучшения качества мы можем рекурсивно разбить те области пространства, в которых классификатор допускает ошибки. Например, можно добавить еще одно решающее правило, применяемое к входным данным, которые не прошли первого теста, и проверить, меньше ли ширина лепестка, чем 1.75 см (в таком случае мы предсказываем класс Versicolor), или больше (и тогда мы предсказываем Virginica). Эти вложенные правила можно организовать в виде древовидной структуры, называемой **решающим деревом**, как показано на рис. 1.4а. Это дерево индуцирует двумерную **решающую поверхность**, показанную на рис. 1.4б.

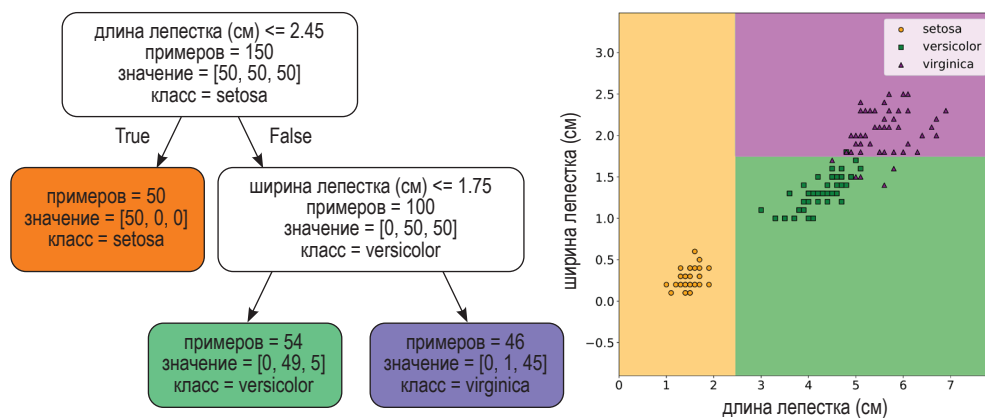


Рис. 1.4 ❖ Пример решающего дерева глубины 2 применительно к набору данных Iris с использованием только признаков «длина лепестка» и «ширина лепестка». Листовые узлы окрашены в соответствии с предсказанным классом. В каждом узле указано количество обучающих примеров, переданных от корня в этот узел; показано, сколько значений из каждого класса попадает в узел. Этот вектор счетчиков можно нормировать и получить распределение меток класса в каждом узле. Затем мы можем выбрать мажоритарный класс. На основе рис. 6.1 и 6.2 из работы [Gér19]. Построено программой по адресу figures.probml.ai/book1/1.4

Мы можем представить это дерево, сохранив для каждого внутреннего узла индекс использованного признака и соответствующее пороговое значение. Все эти **параметры** будем обозначать θ . Как выполнить обучение этих параметров, мы обсудим в разделе 18.1.

1.2.1.4. Минимизация эмпирического риска

Цель обучения с учителем – автоматическое построение моделей классификации типа показанной на рис. 1.4а, так чтобы они надежно предсказывали метки для любых подаваемых на вход данных. Часто для измерения качества модели на такой задаче применяется **частота неправильной классификации** на обучающем наборе:

$$\mathcal{L}(\theta) \triangleq \frac{1}{N} \sum_{n=1}^N \mathbb{I}(y_n \neq f(\mathbf{x}_n; \theta)), \quad (1.2)$$

где $\mathbb{I}(e)$ – бинарная **индикаторная функция**, которая возвращает 1, если условие e истинно, и 0 в противном случае, т. е.:

$$\mathbb{I}(e) = \begin{cases} 1, & \text{если } e \text{ равно true} \\ 0, & \text{если } e \text{ равно false} \end{cases}. \quad (1.3)$$

Здесь предполагается, что все ошибки равноценны. Однако бывают случаи, когда одни ошибки обходятся дороже, чем другие. Предположим, к примеру, что мы ищем съедобные растения в лесу и нашли какие-то ирисы. Еще предположим, что луковицы ирисов *Setosa* и *Versicolor* вкусные, а *Virginica* ядовитые. В таком случае можно было бы использовать асимметричную **функцию потерь** $\ell(y, \hat{y})$, показанную в табл. 1.2.

Таблица 1.2. Гипотетическая асимметричная матрица потерь для классификации набора данных *Iris*

		Оценка		
		Setosa	Versicolor	Virginica
Истина	Setosa	0	1	1
	Versicolor	1	0	1
	Virginica	10	10	0

Тогда можно определить **эмпирический риск** как среднюю потерю предиктора на обучающем наборе:

$$\mathcal{L}(\theta) \triangleq \frac{1}{N} \sum_{n=1}^N \ell(y_n, f(\mathbf{x}_n; \theta)). \quad (1.4)$$

Мы видим, что частота неправильной классификации (1.2) равна эмпирическому риску, если для сравнения истинной метки с предсказанной использовать **бинарную функцию потерь** (zero-one loss):

$$\ell_{01}(y, \hat{y}) = \mathbb{I}(y \neq \hat{y}). \quad (1.5)$$

Детали см. в разделе 5.1.

Один из способов поставить задачу о **подгонке** или **обучении модели** – найти параметры, доставляющие минимум эмпирическому риску на обучающем наборе:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^N \ell(y_n, f(\mathbf{x}_n; \theta)). \quad (1.6)$$

Такая постановка называется **минимизацией эмпирического риска**.

Однако наша истинная цель состоит в том, чтобы минимизировать ожидаемую потерю на будущих данных, которые не предъявлялись модели. То

есть мы хотим добиться **обобщаемости**, а не просто хорошего качества на обучающем наборе. Мы обсудим этот важный момент в разделе 1.2.3.

1.2.1.5. Неопределенность

[Надлежит избегать] ложной уверенности, проистекающей из неведения вероятностной природы мира, из желания видеть черное и белое там, где правильнее было бы видеть серое.

— Иммануил Кант в пересказе Марии Конниковой [Kon20]

Во многих случаях мы не можем точно предсказать, какой выход воспоследует из известного входа, поскольку не знаем, как устроено отображение входа в выход (это называется **эпистемической неопределенностью** или **неопределенностью модели**), и (или) вследствие внутренне присущей (неустранимой) стохастичности отображения (это называется **алеаторной неопределенностью** или **неопределенностью данных**).

Представление неопределенности в предсказании может играть важную роль в некоторых приложениях. Например, вернемся к примеру с отравленным цветком, матрица потерь которого показана в табл. 1.2. Если мы с высокой вероятностью предсказываем, что это ирис *Virginica*, то не должны есть такую луковицу. С другой стороны, мы могли бы заняться **сбором информации**, скажем выполнить диагностический тест, чтобы уменьшить неопределенность. Дополнительные сведения о принятии оптимальных решений в условиях неопределенности см. в разделе 5.1.

Мы можем описать неопределенность с помощью следующего **условного распределения вероятностей**:

$$p(y = c | \mathbf{x}; \boldsymbol{\theta}) = f_c(\mathbf{x}; \boldsymbol{\theta}), \quad (1.7)$$

где $f: \mathcal{X} \rightarrow [0, 1]^C$ – отображение распределения вероятностей C возможных выходных меток. Поскольку $f_c(\mathbf{x}; \boldsymbol{\theta})$ возвращает вероятность метки класса c , мы требуем, чтобы для каждого c было $0 \leq f_c \leq 1$ и $\sum_{c=1}^C f_c = 1$. Чтобы избежать этого ограничения, часто требуют вместо этого, чтобы модель возвращала ненормированные логарифмы вероятностей. Затем мы преобразуем их в вероятности с помощью **функции softmax**, определенной следующим образом:

$$\mathcal{S}(\mathbf{a}) \triangleq \left[\frac{e^{a_1}}{\sum_{c'=1}^C e^{a_{c'}}}, \dots, \frac{e^{a_C}}{\sum_{c'=1}^C e^{a_{c'}}} \right]. \quad (1.8)$$

Эта функция отображает \mathbb{R}^C в $[0, 1]^C$ и удовлетворяет ограничениям $0 \leq \mathcal{S}(\mathbf{a})_c \leq 1$ и $\sum_{c=1}^C \mathcal{S}(\mathbf{a})_c = 1$. Входные данные softmax, $\mathbf{a} = f(\mathbf{x}; \boldsymbol{\theta})$, называются **логитами**. Подробнее см. раздел 2.5.2. Таким образом, мы определяем общую модель в виде:

$$p(y = c | \mathbf{x}; \boldsymbol{\theta}) = \mathcal{S}_c(\mathbf{x}; \boldsymbol{\theta}). \quad (1.9)$$

В распространенном частном случае этого определения f – **аффинная функция** вида

$$f(\mathbf{x}; \boldsymbol{\theta}) = b + \mathbf{w}^\top \mathbf{x} = b + w_1 x_1 + w_2 x_2 + \dots + w_D x_D, \quad (1.10)$$

где $\boldsymbol{\theta} = (b, \mathbf{w})$ – параметры модели. Такая модель называется **логистической регрессией** и будет подробно обсуждаться в главе 10.

В статистике параметры \mathbf{w} обычно называются **коэффициентами регрессии** (и обозначаются β), а b – **свободным членом**. В МО параметры \mathbf{w} называются **весами**, а b – **смещением**. Эта терминология заимствована из электротехники, где функция f рассматривается как электрическая цепь, которая принимает на входе \mathbf{x} и возвращает $f(\mathbf{x})$. Входы подаются цепи по «проводам», имеющим веса \mathbf{w} . Цепь вычисляет сумму входов и прибавляет постоянное смещение b . (Это употребление термина «смещение» не следует путать со статистическим понятием смещения, обсуждаемым в разделе 4.7.6.1.)

Чтобы не загромождать обозначения, принято включать смещение b в состав весов \mathbf{w} , определив векторы $\tilde{\mathbf{w}} = [b, w_1, \dots, w_D]$ и $\tilde{\mathbf{x}} = [1, x_1, \dots, x_D]$, так что:

$$\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}} = b + \mathbf{w}^\top \mathbf{x}. \quad (1.11)$$

Тем самым мы преобразуем аффинную функцию в **линейную функцию**. Обычно мы будем предполагать, что это уже сделано, поэтому функцию предсказания можно записать в виде:

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}. \quad (1.12)$$

1.2.1.6. Оценка максимального правдоподобия

При обучении вероятностных моделей часто используют отрицательную логарифмическую вероятность в качестве функции потерь:

$$\ell(y; f(\mathbf{x}; \boldsymbol{\theta})) = -\log p(y|f(\mathbf{x}; \boldsymbol{\theta})). \quad (1.13)$$

Причины этого будут раскрыты в разделе 5.1.6.1, но интуиция подсказывает, что хорошей (с низкими потерями) является модель, которая назначает высокую вероятность истинному выходу y , соответствующему входу \mathbf{x} . Средняя отрицательная логарифмическая вероятность обучающего набора равна

$$\text{NLL}(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^N \log p(y_n | f(\mathbf{x}_n; \boldsymbol{\theta})). \quad (1.14)$$

Эта величина и называется **отрицательным логарифмическим правдоподобием**. Минимизировав ее, мы найдем **оценку максимального правдоподобия** (maximum likelihood estimate – **MLE**):

$$\hat{\boldsymbol{\theta}}_{\text{mle}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \text{NLL}(\boldsymbol{\theta}). \quad (1.15)$$

Ниже мы увидим, что это очень распространенный способ подгонки моделей к данным.

1.2.2. Регрессия

Теперь предположим, что требуется предсказать вещественную величину $y \in \mathbb{R}$, а не метку класса $y \in \{1, \dots, C\}$; такая задача называется **регрессией**. Так, в случае набора Iris y может быть степенью токсичности луковицы цветка или средней высотой растения.

Регрессия очень похожа на классификацию. Но, поскольку выходом является вещественное значение, нам нужна другая функция потерь. Для регрессии чаще всего используют **квадратичную потерю**, или ℓ_2 -**потерю**:

$$\ell_2(y; \hat{y}) = (y - \hat{y})^2. \quad (1.16)$$

Эта функция штрафует за большие **невязки** $y - \hat{y}$ сильнее, чем за малые¹. Эмпирический риск при использовании квадратичной потери равен **средне-квадратической ошибке (СКО, англ. MSE)**:

$$\text{MSE}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N (y_n - f(\mathbf{x}_n; \boldsymbol{\theta}))^2. \quad (1.17)$$

Вспоминая обсуждение в разделе 1.2.1.5, мы должны также смоделировать неопределенность предсказания. В задачах регрессии принято предполагать, что распределение выходов **гауссово**, или **нормальное**. В разделе 2.6 мы объясним, что это распределение определено формулой

$$\mathcal{N}(y|\mu, \sigma^2) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y-\mu)^2}, \quad (1.18)$$

где μ – среднее, σ^2 – дисперсия, а $\sqrt{2\pi\sigma^2}$ – нормировочная постоянная, которая необходима, чтоб интеграл плотности был равен 1. В контексте регрессии среднее может зависеть от входов: $\mu = f(\mathbf{x}_n; \boldsymbol{\theta})$. Поэтому мы получаем такое условное распределение вероятностей:

$$p(y|\mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}(y|f(\mathbf{x}; \boldsymbol{\theta}); \sigma^2). \quad (1.19)$$

Если предположить, что дисперсия σ^2 фиксирована (для простоты), то отрицательное логарифмическое правдоподобие принимает вид:

$$\text{NLL}(\boldsymbol{\theta}) = -\sum_{n=1}^N \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp \left(-\frac{1}{2\sigma^2} (y_n - f(\mathbf{x}_n; \boldsymbol{\theta}))^2 \right) \right] \quad (1.20)$$

$$= \frac{N}{2\sigma^2} \text{MSE}(\boldsymbol{\theta}) + \text{const}. \quad (1.21)$$

¹ Если в данных имеются выбросы, то квадратичный штраф может оказаться слишком суровым. В таких случаях лучше использовать более **робастную** ℓ_1 -потерю. Подробнее см. раздел 11.6.

Мы видим, что NLL пропорциональна MSE. Поэтому вычисление оценки максимального правдоподобия параметров сводится к минимизации квадратичной ошибки, что представляется разумным подходом к подгонке модели.

1.2.2.1. Линейная регрессия

В качестве примера модели регрессии рассмотрим одномерные данные на рис. 1.5а. Их можно аппроксимировать простой моделью **линейной регрессии** вида

$$f(x; \theta) = b + wx, \quad (1.22)$$

где w – **угловой коэффициент**, b – **смещение**, а $\theta = (w, b)$ – параметры модели. Варьируя θ , мы можем уменьшать сумму квадратов ошибок, представленных вертикальными отрезками на рис. 1.5b, и в итоге найдем решение с наименьшей СКО:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \operatorname{MSE}(\theta). \quad (1.23)$$

Детали см. в разделе 11.2.2.1.

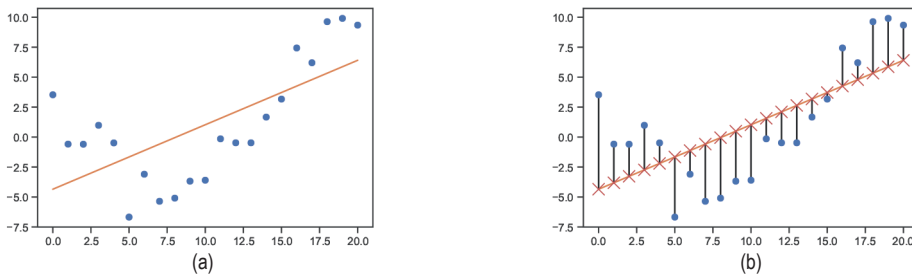


Рис. 1.5 ❖ (а) Линейная регрессия для одномерных данных. (б) Вертикальные отрезки обозначают невязки между наблюдаемым (синий кружочек) и предсказанным (красный крестик) выходным значением. Цель регрессии методом наименьших квадратов – выбрать линию, которая минимизирует сумму квадратов невязок. Сгенерировано программой по адресу figures.probl.ai/book1/1.5

При наличии нескольких входных признаков мы можем написать

$$f(x; \theta) = b + w_1x_1 + \dots + w_Dx_D = b + \mathbf{w}^T \mathbf{x}, \quad (1.24)$$

где $\theta = (w, b)$. Это называется **многофакторной линейной регрессией**.

Например, рассмотрим задачу предсказания температуры как функции двумерного местоположения в комнате. На рис. 1.6а показаны результаты линейной модели вида

$$f(x; \theta) = b + w_1x_1 + w_2x_2. \quad (1.25)$$

Мы можем обобщить эту модель на $D > 2$ входных признаков (скажем, добавить время суток), но тогда ее будет труднее визуализировать.

1.2.2.2. Полиномиальная регрессия

Линейная модель на рис. 1.5а, очевидно, не очень хорошо аппроксимирует данные. Мы можем улучшить дело, воспользовавшись **моделью полиномиальной регрессии** степени D . Она имеет вид $f(x; \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(x)$, где $\boldsymbol{\phi}(x)$ – вектор признаков, построенный по входным данным и имеющий вид

$$\boldsymbol{\phi}(x) = [1; x; x^2; \dots; x^D]. \quad (1.26)$$

Это простой пример **предобработки признаков**, или **конструирования признаков**. На рис. 1.7а видно, что при $D = 2$ результаты аппроксимации гораздо лучше. Можно и дальше увеличивать D , а вместе с ним и количество параметров модели, вплоть до $D = N - 1$, когда число параметров сравняется с числом точек и интерполяция будет идеальной. В такой модели СКО будет равно 0, как показано на рис. 1.7с. Однако интуитивно понятно, что результирующая функция вряд ли окажется хорошим предиктором для будущих данных, потому что она слишком «заковыристая». Мы обсудим этот вопрос подробнее в разделе 1.2.3.

Мы можем также применить полиномиальную регрессию к многомерным входным данным. Например, на рис. 1.6b показаны предсказания модели температуры, когда входные данные аппроксимируются квадратичной функцией:

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2. \quad (1.27)$$

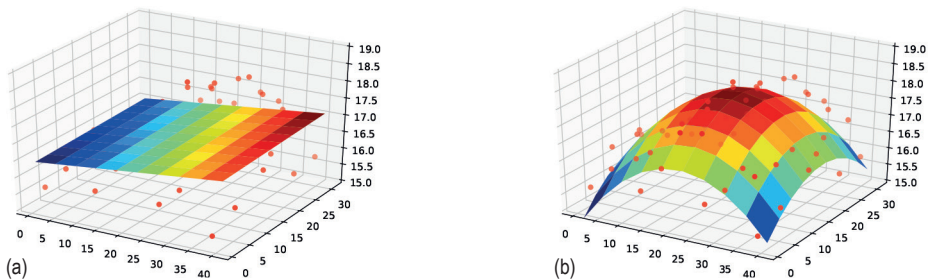


Рис. 1.6 ❖ Линейная и полиномиальная регрессия применительно к двумерным данным. По вертикальной оси отложена температура, а по горизонтальным – местоположение в комнате. Данные были собраны с помощью беспроводных однокристалльных датчиков в лаборатории Intel в Беркли, штат Калифорния (используются с разрешения Ромейна Тибо). (а) Аппроксимирующая плоскость вида $\hat{f}(x) = w_0 + w_1x_1 + w_2x_2$. (б) Данные о температуре аппроксимированы квадратичной функцией вида $\hat{f}(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2$.

Построено программой по адресу figures.probl.ai/book1/1.6

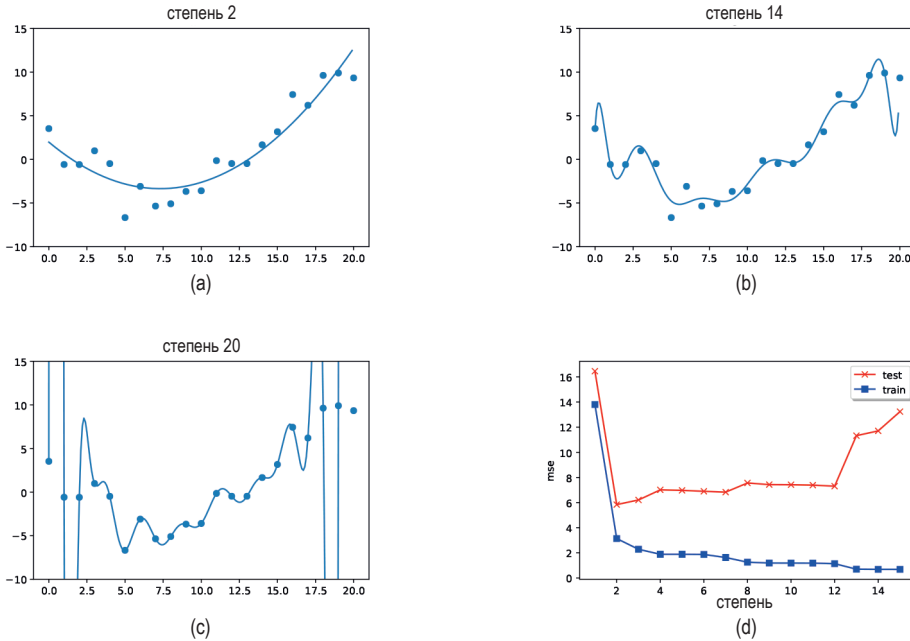


Рис. 1.7 ❖ (а–с) Аппроксимация 21 точки полиномами степени 2, 14 и 20 (данные те же, что на рис. 1.5). (д) Зависимость СКО от степени. Построено программой по адресу figures.problml.ai/book1/1.7

Квадратичная модель лучше аппроксимирует данные, чем линейная, показанная на рис. 1.6а, потому что улавливает тот факт, что в центре комнаты теплее. Можно также добавить перекрестные произведения типа x_1x_2 , чтобы уловить эффекты взаимодействия. Детали см. в разделе 1.5.3.2.

Отметим, что во всех перечисленных выше моделях функция-предиктор линейно зависит от параметров \mathbf{w} , хотя и нелинейно от входных данных \mathbf{x} . Это важно, потому что линейная модель индуцирует среднеквадратическую функцию потерь $MSE(\theta)$, имеющую единственный глобальный оптимум, как будет объяснено в разделе 11.2.2.1.

1.2.2.3. Глубокие нейронные сети

В разделе 1.2.2.2 мы вручную задали преобразование входных признаков – полиномиальное разложение $\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2, \dots]$. Можно создать гораздо более мощные модели, обучившись производить такое нелинейное **выделение признаков** автоматически. Если считать, что $\phi(\mathbf{x})$ имеет собственный набор параметров, скажем \mathbf{V} , то полная модель будет иметь вид:

$$f(\mathbf{x}; \mathbf{w}; \mathbf{V}) = \mathbf{w}^T \phi(\mathbf{x}; \mathbf{V}). \quad (1.28)$$

Мы можем рекурсивно разложить функцию выделения признаков $\phi(\mathbf{x}; \mathbf{V})$ в композицию более простых функций. В результате получается модель, состоящая из L вложенных функций:

$$f(\mathbf{x}; \boldsymbol{\theta}) = f_L(f_{L-1}(\dots(f_1(\mathbf{x}))\dots)), \quad (1.29)$$

где $f_\ell(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta}_\ell)$ – функция слоя ℓ . Последний слой линейный и имеет вид $f_L(\mathbf{x}) = \mathbf{w}^\top f_{1:L-1}(\mathbf{x})$, где $f_{1:L-1}(\mathbf{x})$ – обученная функция выделения признаков (экстрактор). Эта идея – ключ к **глубоким нейронным сетям (ГНС)**, у которых есть такие широко известные варианты, как **сверточные нейронные сети (СНС)** для изображений и **рекуррентные нейронные сети (РНС)** для последовательностей. Подробнее см. часть III.

1.2.3. Переобучение и обобщаемость

Мы можем переписать эмпирический риск в формуле (1.4) в эквивалентной, но более удобной форме:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{train}}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{train}}} \ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta})), \quad (1.30)$$

где $|\mathcal{D}_{\text{train}}|$ – размер обучающего набора $\mathcal{D}_{\text{train}}$. Эта форма полезна, потому что из нее явно видно, на каком наборе данных вычисляется функция потерь.

Если модель достаточно гибкая, то можно свести потери на обучающем наборе к нулю (в предположении, что метки незашумленные), просто запомнив правильный выход для каждого входа. Так, модель на рис. 1.7с идеально интерполирует обучающие данные (не считая одной точки справа). Но нас-то интересует точность предсказания на новых данных, которые могут и не быть частью обучающего набора. Если модель точно аппроксимирует обучающие данные, но при этом слишком сложна, то говорят, что она **переобучена**.

Чтобы выяснить, является ли модель переобученной, предположим (пока), что имеется доступ к истинному (но неизвестному) распределению $p^*(\mathbf{x}, \mathbf{y})$, использованному для порождения обучающего набора. Тогда вместо вычисления эмпирического риска мы вычисляем теоретически ожидаемую потерю, или **риск на генеральной совокупности**:

$$\mathcal{L}(\boldsymbol{\theta}; p^*) \triangleq \mathbb{E}_{p^*(\mathbf{x}, \mathbf{y})}[\ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta}))]. \quad (1.31)$$

Разность $\mathcal{L}(\boldsymbol{\theta}; p^*) - \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{train}})$ называется **разрывом обобщения**. Если разрыв обобщения для модели велик (т. е. эмпирический риск низкий, а риск на генеральной совокупности высокий), то можно заподозрить переобучение. На практике распределение p^* неизвестно. Однако мы можем разбить имеющиеся данные на два подмножества: обучающий набор и **тестовый набор**. И аппроксимировать риск на генеральной совокупности **риском на тестовом наборе**:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{test}}) \triangleq \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{test}}} \ell(\mathbf{y}, f(\mathbf{x}; \boldsymbol{\theta})). \quad (1.32)$$

Так, на рис. 1.7d построен график ошибки на обучающем и на тестовом наборе для полиномиальной регрессии в виде функции от степени D . Мы видим, что ошибка на обучающем наборе стремится к 0 при усложнении модели. Однако ошибка на тестовом наборе имеет характерную U-образную форму: в левой части, где $D = 1$, модель недообучена, а справа, где $D \gg 1$, переобучена. А вот при $D = 2$ сложность модели как раз такая, «как надо».

Как правильно выбрать сложность модели? Если для оценивания различных моделей использовать обучающий набор, то мы всегда будем выбирать самую сложную модель, потому что у нее наибольшее число **степеней свободы**, а значит, наименьшая потеря. Поэтому выбирать следует модель с минимальной потерей на тестовом наборе.

На практике данные следует разбивать на три множества: обучающий набор, тестовый и **контрольный набор**; последний используется для выбора модели, а тестовый только для оценки качества на будущих данных (риска на генеральной совокупности). То есть тестовый набор не используется ни для обучения модели, ни для ее выбора. Дальнейшие детали см. в разделе 4.5.4.

1.2.4. Теорема об отсутствии бесплатных завтраков

Все модели неверны, но некоторые из них полезны.

— Джордж Бок [BD87, стр. 424]¹

В литературе описано множество самых разных моделей, поэтому естественно спросить, какая из них наилучшая. К сожалению, не существует одной модели, которая была бы оптимальна для всех задач, — иногда этот факт называют **теоремой об отсутствии бесплатных завтраков** [Wol96]. Причина в том, что набор допущений (иногда называемый **индуктивным смещением**), который хорошо работает в одной предметной области, может давать никуда не годные результаты в другой. При выборе подходящей модели лучше всего опираться на знание предметной области и (или) на метод проб и ошибок (т. е. применять такие способы выбора модели, как перекрестная проверка (раздел 4.5.4) или байесовские методы (раздел 5.2.2)). Поэтому так важно иметь в своем арсенале много моделей и алгоритмических методов.

1.3. ОБУЧЕНИЕ БЕЗ УЧИТЕЛЯ

В обучении с учителем предполагается, что с каждым входным примером x в обучающем наборе ассоциировано множество выходных меток y , а наша цель — обучиться отображению входа в выход. Хотя это полезно, а иногда трудно, по сути дела, обучение с учителем — всего лишь «напыщенная аппроксимация кривой» [Pea18].

¹ Джордж Бок — профессор статистики Висконсинского университета, ныне на пенсии.

Гораздо более интересная задача – попытаться «извлечь смысл» из данных, а не просто обучить отображение. То есть у нас есть только наблюдаемые «входы» $\mathcal{D} = \{\mathbf{x}_n : n = 1 : N\}$, но никаких соответствующих им «выходов» \mathbf{y}_n . Такая постановка задачи называется **обучением без учителя**.

С вероятностной точки зрения, задачу обучения без учителя можно рассматривать как обучение безусловной модели вида $p(\mathbf{x})$, которая умеет порождать новые данные \mathbf{x} , тогда как обучение с учителем – это обучение условной модели $p(\mathbf{y}|\mathbf{x})$, задающей распределение выходов при условии входов¹.

В обучении без учителя нет нужды собирать большие помеченные наборы данных для обучения, что может оказаться делом долгим и дорогостоящим (представьте, что вы просите врачей пометить медицинские изображения).

Нет также нужды обучаться разбиению мира на категории, зачастую произвольные. Подумайте, например, о задаче пометить на видео действие, скажем «питье» или «прихлебывание». Когда оно начинается: когда человек поднимает стакан или когда стакан касается губ? Люди зачастую не могут прийти к единой точке зрения по таким вопросам [Idr+17], а значит, задача поставлена некорректно. Поэтому странно было бы ожидать, что машина сможет обучиться такому отображению².

Наконец, обучение без учителя вынуждает модель «объяснять» входы высокой размерности, а не только выходы низкой размерности. Это позволяет обучать более развитые модели «устройства мира».

Джеффри Хинтон, знаменитый профессор МО из Торонтского университета, писал:

Когда мы учимся видеть, никто не подсказывает нам правильные ответы – мы просто смотрим. Бывает, мама говорит «это собака», но информации в этих словах очень мало. Хорошо, если таким способом удастся получить хоть несколько битов информации – пусть даже один бит в секунду. В зрительной системе мозга порядка 10^{14} нейронных связей. А наша жизнь продолжается всего 10^9 с. Так что обучаться со скоростью один бит в секунду бесполезно. Нужно быстрее, порядка 10^5 бит в секунду. И есть только одно место, где можно добыть так много информации: сами входные данные.

— Джеффри Хинтон (цитируется по [Gor06])

1.3.1. Кластеризация

Простой пример обучения без учителя – проблема нахождения **кластеров** в данных. Цель – разбить множество входов на области, содержащие «похо-

¹ У статистиков принято использовать букву \mathbf{x} для обозначения экзогенных переменных, которые не моделируются, а просто являются входными данными. Поэтому безусловная модель обозначалась бы $p(\mathbf{y})$, а не $p(\mathbf{x})$.

² Более разумный подход – попытаться уловить распределение вероятностей меток, предоставленных «толпой» аннотаторов (см., например, [Dum+18; Ago+19]). При этом учитывается тот факт, что данному входу может соответствовать несколько «правильных» меток – просто в силу неоднозначности задачи.

жие» точки. Рассмотрим двумерный вариант набора данных Iris. На рис. 1.8а показаны точки без меток класса. Интуитивно понятно, что в данных имеется по меньшей мере два кластера, в левом нижнем и в правом верхнем углу. Кроме того, если предположить, что «хороший» кластер должен быть относительно компактным, то точки в правом верхнем углу хочется разбить на два подкластера (по крайней мере). Результирующее разбиение на три кластера показано на рис. 1.8б. (Отметим, что никто не говорит, сколько должно быть кластеров; мы сами должны выбрать компромисс между сложностью модели и качеством аппроксимации ею данных. Как это сделать, обсуждается в разделе 21.3.7.)

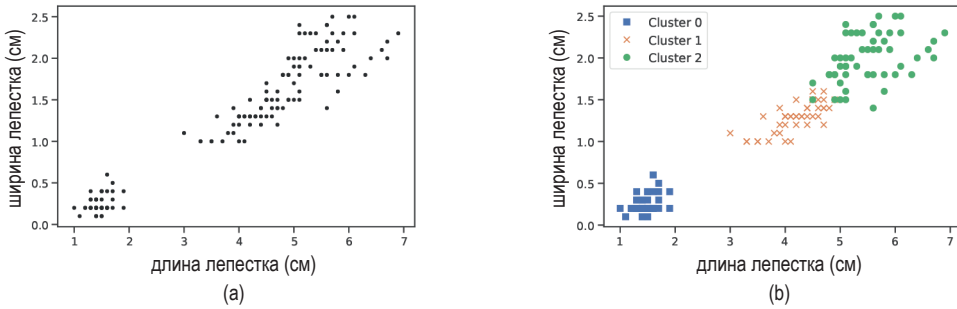


Рис. 1.8 ❖ (а) Диаграмма рассеяния признаков, относящихся к лепесткам, в наборе данных об ирисах. (б) Результат кластеризации без учителя при $K = 3$. Построено программой по адресу figures.problm.ai/book1/1.8

1.3.2. Обнаружение латентных «факторов изменчивости»

При работе с данными высокой размерности часто бывает полезно понизить размерность путем проецирования на подпространство меньшей размерности, которое улавливает основные особенности данных. Один из подходов к этой проблеме – предположить, что все наблюдаемые многомерные выходы $\mathbf{x}_n \in \mathbb{R}^D$ порождены множеством скрытых, т. е. ненаблюдаемых **латентных факторов** низкой размерности $\mathbf{z}_n \in \mathbb{R}^K$. Мы можем наглядно представить модель в виде $\mathbf{z}_n \rightarrow \mathbf{x}_n$, где стрелка обозначает причинно-следственную связь. Поскольку латентные факторы \mathbf{z}_n нам неизвестны, часто предполагается простая модель априорной вероятности для $p(\mathbf{z}_n)$, например гауссова, согласно которой каждый фактор является случайным K -мерным вектором. Если данные вещественные, то можно также использовать гауссово правдоподобие.

Простейший пример – использование линейной модели $p(\mathbf{x}_n | \mathbf{z}_n; \theta) = \mathcal{N}(\mathbf{x}_n | \mathbf{W}\mathbf{z}_n + \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Результирующая модель называется **факторным анализом** (ФА). Он похож на линейную регрессию с тем отличием, что мы наблюдаем только выходы \mathbf{x}_n , а не входы \mathbf{z}_n . В частном случае, когда $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, это сводится к модели вероятностного **метода главных компонент** (principal components analysis — **PCA**), который мы будем рассматривать в разделе 20.1.

На рис. 1.9 показано, как этот метод позволяет найти двумерное линейное подпространство, будучи применен к простым трехмерным данным.

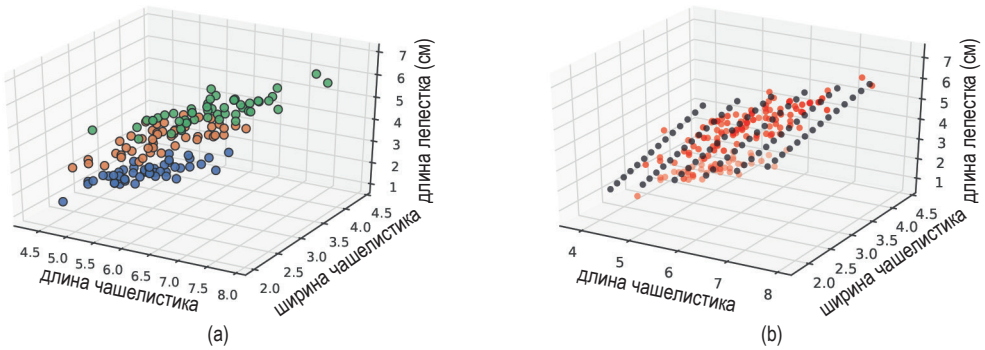


Рис. 1.9 ❖ (а) Диаграмма рассеяния признаков данных об ирисах (первые три признака). Цвет точек соответствует классам. (б) Мы аппроксимируем трехмерные данные двумерным линейным подпространством, применяя метод РСА. Метки классов игнорируются. Красные точки – оригинальные данные, черные точки сгенерированы по модели с использованием $\hat{\mathbf{x}} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu}$, где \mathbf{z} – латентные точки на выведенном двумерном линейном многообразии. Построено программой по адресу figures.problml.ai/book1/1.9

Конечно, предположение о линейности отображения \mathbf{z}_n в \mathbf{x}_n чрезмерно ограничительно. Однако мы можем создать нелинейные обобщения, определив $p(\mathbf{x}_n|\mathbf{z}_n; \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_n|f(\mathbf{z}_n; \boldsymbol{\theta}), \sigma^2\mathbf{I})$, где $f(\mathbf{z}; \boldsymbol{\theta})$ – нелинейная модель, например глубокая нейронная сеть. Обучить такую модель (т. е. оценить параметры $\boldsymbol{\theta}$) оказывается гораздо труднее, потому что необходимо выводить как входы нейронной сети, так и параметры модели. Но существуют различные приближенные методы, например вариационные автокодировщики (см. раздел 20.3.5).

1.3.3. Самостоятельное обучение

Недавно приобрел популярность новый подход к обучению без учителя – **самостоятельное обучение** (self-supervised learning). В этом случае мы создаем замещающие задачи обучения с учителем по непомеченным данным. Например, можно попробовать обучиться предсказанию цветного изображения по полутоновому или замаскировать некоторые слова в предложении и попытаться предсказать их по окружающему контексту. Мы надеемся, что получившийся в результате предиктор $\hat{\mathbf{x}}_1 = f(\mathbf{x}_2; \boldsymbol{\theta})$, где \mathbf{x}_2 – наблюдаемый вход, а $\hat{\mathbf{x}}_1$ – предсказанный выход, сможет обучиться полезным признакам на основе данных, а затем использовать их в последующих стандартных задачах обучения с учителем. Тем самым мы обходим трудную проблему вывода «истинных латентных факторов» \mathbf{z} , стоящих за наблюдаемыми данными, а вместо этого полагаемся на стандартные методы обучения с учителем. Мы подробнее обсудим этот подход в разделе 19.2.

1.3.4. Оценка обучения без учителя

Хотя идея обучения без учителя кажется весьма соблазнительной, очень трудно оценить качество такого метода обучения, потому что результат не с чем сравнивать [ТОВ16]. Наиболее распространенный способ оценки моделей, обученных без учителя, – измерение вероятности, назначенной моделью не предъявлявшимся ранее тестовым примерам. Это можно сделать, вычислив (безусловное) отрицательное логарифмическое правдоподобие данных:

$$\mathcal{L}(\theta; \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}|\theta). \quad (1.33)$$

Тем самым мы сводим проблему обучения без учителя к проблеме **оценивания плотности**. Идея в том, что хорошую модель не должны «удивлять» примеры фактических данных (т. е. она назначает им высокую вероятность). Кроме того, сумма вероятностей должна быть равна 1.0, а значит, если модель назначает высокую вероятность тем областям пространства данных, из которых взяты примеры, то областям, откуда нет данных, неявно назначается низкая вероятность. Таким образом, модель обучилась улавливать **типичные паттерны** в данных. Это можно использовать в алгоритме **сжатия данных**.

К сожалению, оценить плотность трудно, особенно если размерность велика. Кроме того, модель, которая назначает высокую вероятность данным, может не обучиться полезным паттернам (ведь она могла бы просто запомнить все обучающие примеры).

Альтернативный способ оценивания – использовать представление, полученное в результате обучения без учителя, в качестве признаков или подать на вход следующему далее методу обучения с учителем. Если метод обучения без учителя выявил полезные паттерны, то, возможно, ими удастся воспользоваться для обучения с учителем на гораздо меньшем объеме помеченных данных, чем при работе с оригинальными признаками. Например, в разделе 1.2.1.1 мы видели, что четыре вручную определенных признака ирисов содержат большую часть информации, необходимой для классификации. И таким образом сумели обучить почти идеальный классификатор всего на 150 примерах. Если бы на вход подавались исходные пиксели, то для достижения сравнимого качества потребовалось бы куда больше примеров (см. раздел 14.1). Следовательно, мы сможем повысить **выборочную эффективность** обучения (т. е. уменьшить количество помеченных примеров, необходимых для получения приемлемого качества), если предварительно обучимся хорошему представлению.

Повышенная выборочная эффективность – полезная метрика оценки, но во многих приложениях, особенно научных, цель обучения без учителя – *понять данные*, а не улучшить качество на какой-то задаче предсказания. Для этого нужны модели, **допускающие интерпретацию**, но при этом способные генерировать или «объяснять» большинство наблюдаемых в данных паттернов. Перефразируя Платона, можно сказать, что цель состоит в том, чтобы выяс-

нить, как «природа делится в местах сочленений»¹. Конечно, чтобы оценить, правильно ли мы распознали истинную структуру, стоящую за некоторым набором данных, зачастую необходимо ставить эксперименты и, стало быть, взаимодействовать с миром. Мы продолжим обсуждение этой темы в разделе 1.4.

1.4. ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ

Помимо обучения с учителем и без учителя, есть еще и третий вид МО – **обучение с подкреплением** (ОП, англ. **RL**). В проблемах этого класса система или **агент** должна обучиться взаимодействию со своим окружением. Это можно представить в виде **политики** $a = \pi(x)$, определяющей, какое действие предпринять в ответ на любой возможный вход x (действие выводится из состояния окружающей среды).

Например, рассмотрим агента, который обучается играть в видеоигру, скажем Atari Space Invaders (см. рис. 1.10а). В этом случае входом x является изображение (или последовательность прошлых изображений), а выходом – направление движения (влево или вправо) и признак пуска ракеты. В качестве более сложного примера рассмотрим робота, обучающегося ходить (см. рис. 1.10b). Здесь вход x – множество положений и углов сочленения для всех конечностей, а выход – множество сигналов, подаваемых на приводы или электрические двигатели.

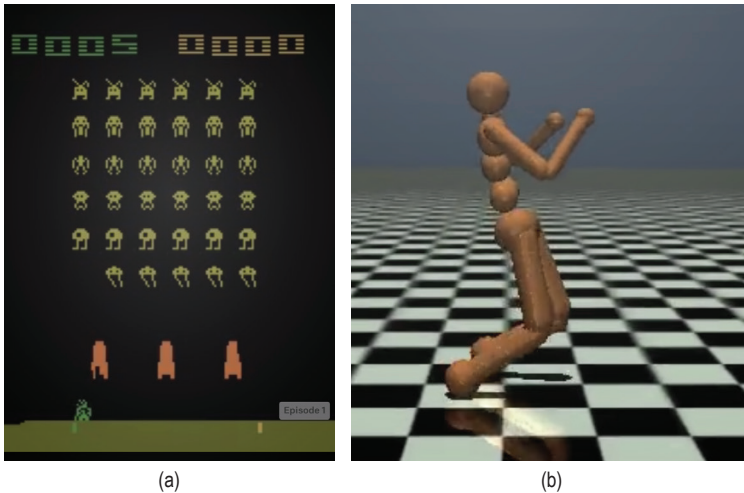


Рис. 1.10 ❖ Пример некоторых задач управления. (а) Игра Space Invaders компании Atari. Взято с <https://gym.openai.com/envs/SpaceInvaders-v0/>. (б) Управление роботом-гуманоидом в эмуляторе MuJoCo; цель – добиться максимальной скорости ходьбы без падений. Взято с <https://gym.openai.com/envs/Humanoid-v2/>

¹ Отсылка к следующему месту из диалога Платона «Федр»: «Другой [род] состоит в уменьи делить предмет на виды – и делить, как водится, почленно, так чтобы, подобно плохому повару, не раздробить ни одной части». – *Прим. перев.*

Отличие от обучения с учителем заключается в том, что системе никто не говорит, какое действие наилучшее (т. е. какой выход следует сгенерировать для данного входа). Вместо этого система время от времени получает сигнал **вознаграждения** (или наказания) в ответ на предпринятые ей действия. Это скорее напоминает **обучение с критиком**, который периодически поднимает или опускает большой палец, а не с учителем, который на каждом шаге говорит, что нужно делать.

В последнее время популярность ОП заметно выросла благодаря широкой применимости (ведь в качестве сигнала вознаграждения, который агент пытается оптимизировать, можно выбирать любую интересующую метрику). Но по разным причинам заставить ОП работать труднее, чем в случае обучения с учителем или без него. Основная трудность заключается в том, что сигнал вознаграждения может подаваться лишь изредка (например, только когда агент в конечном итоге достигает желаемого состояния), и даже в этих случаях может быть неясно, какое из многочисленных действий привело к успеху. (Примером могут служить шахматы, когда единственный сигнал – выигрыш или проигрыш – подается в конце партии.)

Чтобы компенсировать скудость информации, поступающей от сигнала вознаграждения, часто используют другие источники информации, например проводимые экспертами демонстрации, которые можно использовать, как в обучении с учителем, или неразмеченные данные, которые могут использоваться системой обучения без учителя, чтобы выявить структуру окружающей среды. Благодаря таким уловкам становится реально обучиться на ограниченном числе испытаний (раундов взаимодействия с окружающей средой). Как заметил Ян Лекун в лекции на конференции NIPS¹ в 2016 году, «если представить интеллект в виде торта, то обучение без учителя будет шоколадной прослойкой, обучение с учителем – глазурью, а обучение с подкреплением – вишенкой». Эта мысль иллюстрируется на рис. 1.11.

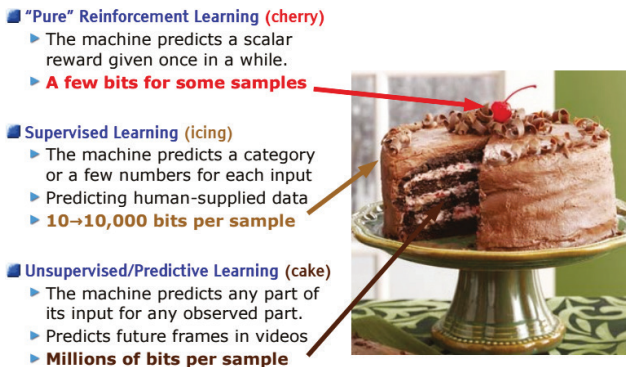


Рис. 1.11 ❖ Три типа машинного обучения, представленные в виде слоев шоколадного торта. Этот рисунок (взятый с <https://bit.ly/2m65Vs1>) был использован в лекции Яна Лекунa на конференции NIPS'16 и печатается с его разрешения

¹ NIPS означает Neural Information Processing Systems (Системы обработки нейронной информации). Это одна из самых авторитетных конференций по машинному обучению. Недавно она была переименована в **NeurIPS**.

Дополнительную информацию об ОП можно найти во втором томе этой книги, [Mur22].

1.5. ДАННЫЕ

Предмет машинного обучения – построение моделей данных с помощью различных алгоритмов. Хотя мы акцентируем внимание на аспектах моделирования и алгоритмов, необходимо отметить, что природа и качество обучающих данных также играют важную роль в успехе любой обученной модели.

В этом разделе мы кратко опишем некоторые используемые в этой книге наборы данных, содержащие изображения и текст. Мы также немного поговорим о предобработке данных.

1.5.1. Некоторые широко известные наборы изображений

В этом разделе приводится краткое обсуждение используемых в книге наборов данных, содержащих изображения.

1.5.1.1. Небольшие наборы изображений

Один из самых простых и широко распространенных наборов данных – **MNIST** [LeC+98; YB19]¹. Этот набор содержит 60 000 обучающих и 10 000 тестовых полутоновых изображений рукописных цифр из 10 категорий, каждое размером 28×28 пикселей. Каждый пиксель представлен целым числом в диапазоне $\{0, 1, \dots, 255\}$; обычно они приводятся к диапазону $[0, 1]$, чтобы представить яркость пикселей. Дополнительно можно преобразовать изображение в бинарное, воспользовавшись методом бинаризации (см. рис. 1.12а).

Набор MNIST настолько широко используется в сообществе МО, что знаменитый ученый Джеффри Хинтон назвал его «дрозофилой машинного обучения», поскольку если некоторый метод не работает на MNIST, то вряд ли он будет хорошо работать на более трудных наборах данных. Однако в наши дни классификация MNIST считается «слишком легкой» задачей, поскольку различить большинство пар цифр можно всего по одному пикселю. Были предложены различные расширения.

В работе [Coh+17] предложен набор **EMNIST** (extended MNIST), включающий также строчные и заглавные буквы (см. рис. 1.12b). Этот набор данных гораздо труднее MNIST, поскольку классов 62 и некоторые из них неоднозначны (например, сравните цифру 1 со строчной буквой l).

¹ Акроним MNIST означает Modified National Institute of Standards. Слово «modified» употребляется, потому что изображения были подвергнуты предварительной обработке с целью поместить цифры примерно в центр.

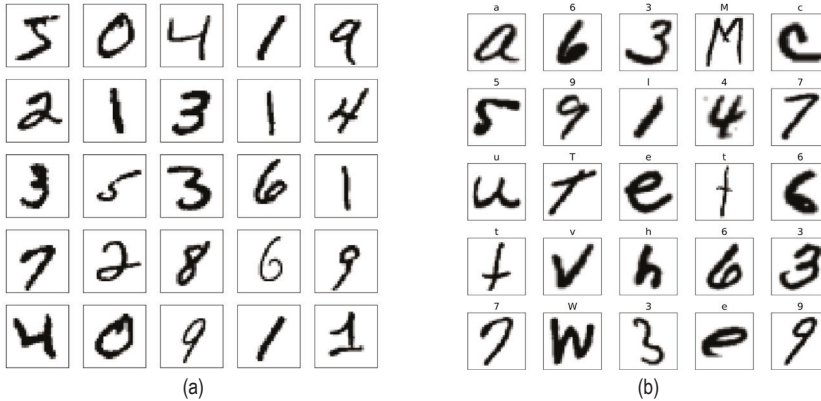


Рис. 1.12 ❖ (a) Визуализация набора данных MNIST. Размер каждого изображения 28×28 . Всего имеется 60k обучающих и 10k тестовых примеров. Показаны первые 25 изображений из обучающего набора. Построено программой по адресу figures.problml.ai/book1/1.12. (b) Визуализация набора данных EMNIST. Имеется 697 932 обучающих и 116 323 тестовых примеров, размер каждого 28×28 . Имеется 62 класса (a–z, A–Z, 0–9). Показаны первые 25 изображений из обучающего набора. Построено программой по адресу figures.problml.ai/book1/1.12

В работе [XRV17] предложен набор **Fashion-MNIST**. Его размер и форма изображений такие же, как в MNIST, но изображены предметы одежды, а не рукописные цифры (см. рис. 1.13a).

Что касается небольших цветных изображений, то наиболее известен набор данных **CIFAR** [КН09]¹. Он содержит 60 000 изображений хорошо знакомых объектов из 10 или 100 классов, примеры см. на рис. 1.13b.

1.5.1.2. ImageNet

Набольшие наборы данных хороши для апробирования идей, но важно также тестировать методы на более крупных наборах – с точки зрения как размера изображений, так и количества помеченных примеров. Самым употребительным из таких наборов является **ImageNet** [Rus+15]. Он содержит порядка 14 млн изображений размера $256 \times 256 \times 3$, иллюстрирующих различные объекты из 20 000 классов, примеры см. на рис. 1.14a.

Набор данных ImageNet был положен в основу конкурса ImageNet Large Scale Visual Recognition Challenge (**ILSVRC**), проводившегося с 2010 по 2018 год. В нем использовалось подмножество, содержащее 1.3 млн изображений, принадлежащих 1000 классам. За эти годы сообщество добилось значительного прогресса, как показывает рис. 1.14b. В частности, в 2015 году СНС впервые превзошла человека (или, по крайней мере, одного человека, Андрея Карпатого) в задаче классификации изображений из набора ImageNet. Заметим, что это не означает, что СНС видят лучше человека (см., например, работу

¹ CIFAR означает Canadian Institute For Advanced Research. Эта организация финансировала разметку набора данных, за основу которого взят набор данных TinyImages по адресу <http://groups.csail.mit.edu/vision/TinyImages/>, созданный Антонио Торралба. Детали см. в [КН09].

[YL21], где описаны некоторые типичные ошибки). Скорее всего, это отражает тот факт, что в наборе данных много нюансов **детальной классификации**, например различий между «тигром» и «тигровой кошкой», которые человеку просто непонятны. С другой стороны, достаточно гибкие СНС могут обучиться любым паттернам, в том числе случайным меткам [Zha+17a].



Рис. 1.13 ❖ (a) Визуализация набора данных Fashion-MNIST [XRV17]. Этот набор данных имеет такой же размер, как MNIST, но труднее для классификации. Всего имеется 10 классов: футболка/топ, брюки, пуловер, платье, куртка, сандалии, рубашка, кроссовки, сумка, полусапоги. Показаны первые 25 изображений из обучающего набора. Построено программой по адресу figures.problml.ai/book1/1.13. (b) Несколько изображений из набора данных CIFAR-10 [KH09]. Все изображения имеют размер $32 \times 32 \times 3$, где последнее измерение (3) означает формат RGB. Набор содержит 50k обучающих и 10k тестовых примеров. Имеется 10 классов: самолет, автомобиль, птица, кошка, собака, лягушка, лошадь, судно и грузовик. Показаны первые 25 изображений из обучающего набора. Построено программой по адресу figures.problml.ai/book1/1.13

Хотя набор ImageNet гораздо труднее MNIST и CIFAR в роли эталона классификации, он тоже дошел почти до точки «насыщения» [Beu+20]. Тем не менее качество работы того или иного метода на наборе ImageNet зачастую оказывается на удивление хорошим предиктором качества на других, никак не связанных задачах классификации изображений (см., например, [Res+19]), поэтому он по-прежнему широко используется.

1.5.2. Некоторые широко известные наборы текстовых данных

Машинное обучение часто применяется к текстам для решения самых разных задач. Эта область называется **обработкой естественного языка** (natural language processing – NLP) (детали см., например, в [JM20]). Ниже даны краткие описания нескольких наборов текстовых данных, используемых в этой книге.

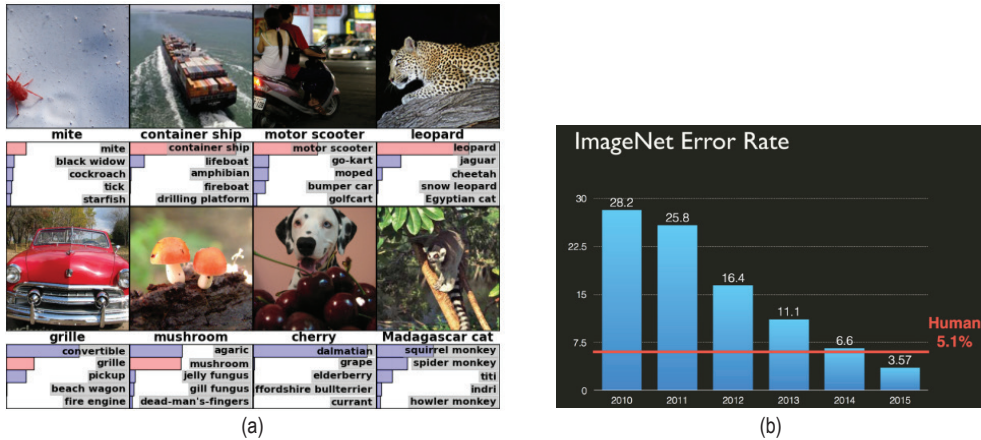


Рис. 1.14 ❖ (а) Примеры изображений из набора данных ImageNet [Rus+15]. Это подмножество состоит из 1.3М цветных обучающих изображений размером 256×256 пикселей. Всего имеется 1000 меток, каждому изображению сопоставляется одна метка, а задача заключается в том, чтобы гарантировать попадание правильной метки в пять предсказаний с наибольшей вероятностью. Под каждым изображением показана истинная метка и распределение вероятностей первых пяти предсказанных меток. Если истинная метка оказывается среди первых пяти, то соответствующий ей столбик окрашен красным. Предсказания сгенерированы сверточной нейронной сетью (СНС) AlexNet (раздел 14.3.2). Заимствовано из работы [KSH12] (рис. 4). Печатается с разрешения Алекса Крижевски. (б) Изменение коэффициента неправильной классификации (непопадание в первые 5) в конкурсах ImageNet со временем. Печатается с разрешения Андрея Карпатого

1.5.2.1. Классификация текста

Простой задачей NLP является классификация текста, ее можно использовать для **классификации спама**, **анализа эмоциональной окраски** (например, является ли обзор фильма или товара положительным или отрицательным) и т. д. Для оценки таких методов часто используется **набор данных IMDB**, содержащий обзоры фильмов (см. [Maa+11]) (IMDB означает Internet Movie Database). В нем 25 000 помеченных примеров для обучения и 25 000 для тестирования. Каждый пример сопровождается бинарной меткой: положительная или отрицательная оценка. Примеры предложений приведены в табл. 1.3.

Таблица 1.3. Примеры двух первых предложений из набора данных IMDB, содержащего обзоры фильмов

- | |
|---|
| 1. this film was just brilliant casting location scenery story direction everyone's really suited the part they played robert <UNK> is an amazing actor ... |
| 2. big hair big boobs bad music and a giant safety pin these are the words to best describe this terrible movie i love cheesy horror movies and i've seen hundreds... |

Первый пример помечен как положительный, второй как отрицательный (<UNK> означает неизвестную лексему).

1.5.2.2. Машинный перевод

Более трудная задача NLP – обучиться отображать предложение x на одном языке в «семантически эквивалентное» предложение y на другом языке; это называется **машинным переводом**. Для обучения таких моделей нужны соответственные пары (x, y) . По счастью, существует несколько таких наборов данных, например от канадского парламента (англо-французские пары) и Европейского союза (Europarl). Подмножество последнего, **набор данных WMT** (Workshop on Machine Translation), состоит из англо-немецких пар и широко используется в качестве эталона для проверки алгоритмов.

1.5.2.3. Другие задачи типа seq2seq

Обобщение машинного перевода – задача об обучении отображения одной последовательности x в другую последовательность y . Она называется **моделью seq2seq** и может рассматриваться как форма многомерной классификации (подробности см. в разделе 15.2.3). Эта постановка задачи очень общая, поэтому можно выделить много частных случаев: **реферирование документов, вопросно-ответные системы** и т. д. Например, в табл. 1.4 показано, как поставить задачу о вопросно-ответной системе как проблему seq2seq: входом является текст T и вопрос Q , а выходом – ответ A , представляющий собой множество слов, возможно, извлеченных из входного текста.

Таблица 1.4. Пары вопрос-ответ для примера текста из набора данных SQuAD

T: In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity . The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud . Short, intense periods of rain in scattered locations are called “showers”.
Q1: What causes precipitation to fall? A1: gravity
Q2: What is another main form of precipitation besides drizzle, rain, snow, sleet and hail? A2: graupe
Q3: Where do water droplets collide with ice crystals to form precipitation? A3: within a cloud

Каждый ответ представляет собой отрывок текста. Эту задачу можно решить с помощью разметки пар предложений. Входом является текстовый абзац T и вопрос Q . На выходе помечаются те слова T , которые отвечают на вопрос Q . Взято из работы [Raj+16]. Печатается с разрешения Перси Лианга.

1.5.2.4. Языковое моделирование

Под очень многоплановым термином «**языковое моделирование**» понимается задача создания безусловных порождающих моделей текстовых предложений, $p(x_1, \dots, x_T)$. Для нее требуются только входные предложения x , без ответственных «меток» y . Поэтому проблему можно рассматривать как одну из форм обучения без учителя, обсуждавшегося в разделе 1.3. Если языковая модель порождает в ответ на вход какой-то выход, как в случае seq2seq, то ее можно рассматривать как условную порождающую модель.

1.5.3. Предобработка дискретных входных данных

Во многих моделях МО предполагается, что данные представляют собой вещественные векторы признаков, $x \in \mathbb{R}^D$. Но иногда входные данные могут содержать дискретные признаки, например категориальные величины типа расы и пола или слова из некоторого словаря. Ниже мы обсудим некоторые способы предобработки таких данных с целью преобразования их в векторную форму. Эта операция часто применяется в различных моделях.

1.5.3.1. Унитарное кодирование

Категориальные признаки необходимо преобразовать в какую-то числовую шкалу, чтобы вычисление их взвешенных комбинаций имело смысл. Стандартный способ такого преобразования называется **унитарным кодированием**, или **индикаторным кодированием** (dummy encoding). Если переменная x может принимать K значений, то результатом ее унитарного кодирования будет one-hot(x) = $[\mathbb{I}(x = 1), \dots, \mathbb{I}(x = K)]$. Например, если имеется 3 цвета (скажем, красный, зеленый и синий), то соответствующие им унитарные векторы имеют вид one-hot(красный) = $[1, 0, 0]$, one-hot(зеленый) = $[0, 1, 0]$, one-hot(синий) = $[0, 0, 1]$.

1.5.3.2. Перекрестные произведения признаков

Линейная модель, в которой используется унитарное кодирование всех категориальных переменных, может уловить **основной эффект** каждой переменной, но не **эффекты взаимодействия** переменных. Например, предположим, что требуется предсказать топливную эффективность транспортного средства по двум категориальным переменным: типу (скажем, городской внедорожник, грузовик или семейный автомобиль) и стране происхождения (скажем, США или Япония). Если конкатенировать унитарные коды тернарных и бинарных признаков, то входные данные будут представлены в следующем виде:

$$\phi(x) = [1; \mathbb{I}(x_1 = S); \mathbb{I}(x_1 = T); \mathbb{I}(x_1 = F); \mathbb{I}(x_2 = U); \mathbb{I}(x_2 = J)], \quad (1.34)$$

где x_1 – тип, а x_2 – страна происхождения.

Эта модель не может уловить зависимости между признаками. Например, мы ожидаем от грузовиков меньшей топливной эффективности, но, быть может, американские грузовики даже менее эффективны, чем японские. Линейная модель (1.34) не может уловить этот факт, потому что вклад страны происхождения не зависит от типа автомобиля.

Это можно исправить, вычислив явные **перекрестные произведения признаков**. Например, мы можем определить новый составной признак с 3×2 возможными значениями, который будет улавливать взаимодействие между типом и страной происхождения. Тогда новая модель принимает вид:

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \quad (1.35)$$

$$\begin{aligned} &= w_0 + w_1 \mathbb{I}(x_1 = S) + w_2 \mathbb{I}(x_1 = T) + w_3 \mathbb{I}(x_1 = F) \\ &+ w_4 \mathbb{I}(x_2 = U) + w_5 \mathbb{I}(x_2 = J) \\ &+ w_6 \mathbb{I}(x_1 = S; x_2 = U) + w_7 \mathbb{I}(x_1 = T; x_2 = U) + w_8 \mathbb{I}(x_1 = F; x_2 = U) \\ &+ w_9 \mathbb{I}(x_1 = S; x_2 = J) + w_{10} \mathbb{I}(x_1 = T; x_2 = J) + w_{11} \mathbb{I}(x_1 = F; x_2 = J). \end{aligned} \quad (1.36)$$

Видно, что перекрестные произведения позволяют преобразовать исходный набор данных в **широкий формат** с гораздо большим числом столбцов.

1.5.4. Предобработка текстовых данных

В разделе 1.5.2 мы кратко обсудили классификацию текста и другие задачи NLP. Для подачи текста на вход классификатору нужно решить ряд вопросов. Во-первых, документы имеют переменную длину, а не являются векторами признаков фиксированной длины, как предполагается во многих моделях. Во-вторых, слова – это категориальные переменные, имеющие много значений (столько, сколько слов в словаре), поэтому размерность соответствующей унитарной кодировки будет очень велика, а в многомерном пространстве нет естественного понятия сходства. В-третьих, во время тестирования могут встретиться слова, которые не предъявлялись на этапе обучения (так называемые **внесловарные слова**, или OOV-слова). Некоторые решения этих проблем мы обсудим ниже, а дополнительные сведения можно найти, например, в работах [BKL10; MRS08; JM20].

1.5.4.1. Модель мешка слов

Простой подход к обработке текстовых документов переменной длины состоит в том, чтобы интерпретировать их как **мешок слов**, в котором порядок слов игнорируется. Чтобы преобразовать такой документ в вектор из фиксированного пространства входов, мы сначала преобразуем каждое слово в **лексему** из некоторого словаря.

Для уменьшения количества лексем применяются различные методы предобработки, например: отбрасывание знаков препинания, преобразование всех слов в нижний регистр, отбрасывание часто встречающихся, но не информативных слов типа «and» и «the» (это называется **исключением стоп-слов**), замена слов основной формой, например замена «running» и «runs» формой «run» (это называется **стеммингом**) и т. д. Детали см., например, в [BL12], а демонстрационный код на странице code.problml.ai/book1/text_preproc_torch.

Обозначим x_{nt} лексему в позиции t n -го документа. Если в словаре имеется D различных лексем, то n -й документ можно представить D -мерным вектором $\tilde{\mathbf{x}}_n$, где \tilde{x}_{nv} – количество вхождений слова v в документ n :

$$\tilde{x}_{nv} = \sum_{t=1}^T \mathbb{I}(x_{nt} = v), \quad (1.37)$$

где T – длина документа n . Теперь документы можно интерпретировать как векторы в пространстве \mathbb{R}^D . Это так называемая **модель векторного пространства** [SWY75; TP10].

Традиционно входные данные хранятся в матрице плана X размера $N \times D$, где D – число признаков. Но в контексте моделей векторного пространства принято представлять входные данные **терм-документной матрицей** размера $D \times N$, где TF_{ij} – частота термина i в документе j (см. рис. 1.15).

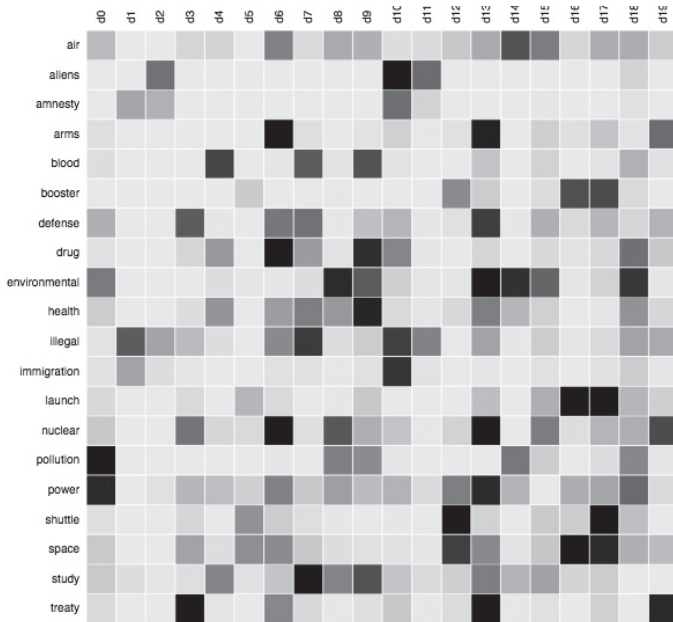


Рис. 1.15 ❖ Пример терм-документной матрицы, в которой исходные счетчики заменены значениями TF-IDF (см. раздел 1.5.4.2). Чем темнее ячейка, тем больше значение. Взято из статьи <https://bit.ly/2kByLQI>. Печатается с разрешения Кристофа Карла Клингга

1.5.4.2 TF-IDF

Одной из проблем, сопутствующих представлению документов векторами счетчиков слов, является тот факт, что часто встречающиеся слова могут оказывать несоразмерное их важности влияние, просто потому что счетчик вхождений для них больше, хотя никакого особого семантического смысла они не несут. Стандартное решение этой проблемы – заменить счетчики логарифмами, что уменьшает влияние слов, многократно встречающихся в одном документе.

А чтобы уменьшить влияние слов, которые вообще часто встречаются (во всех документах), мы вычисляем так называемую **обратную частоту документа**, которая определяется следующим образом: $IDF_i \triangleq \log N / (1 + DF_i)$, где DF_i – число документов, содержащих терм i . Оба преобразования можно объединить и вычислить матрицу **TF-IDF**:

$$\text{TFIDF}_{ij} = \log(\text{TF}_{ij} + 1) \times \text{IDF}_i \quad (1.38)$$

(дополнительно строки часто нормируются). Такая матрица дает семантически более осмысленное представление документов и используется в качестве входа во многих алгоритмах МО. Пример см. по адресу code.problml.ai/book1/tfidf_demo.

1.5.4.3. Погружения слов

Хотя преобразование в матрицу TF-IDF улучшает векторное представление слов, придавая больший вес «информативным» словам и меньший «неинформативным», оно не решает фундаментальную проблему, заключающуюся в том, что семантически родственные слова, например «man» и «woman», могут располагаться в векторном пространстве дальше друг от друга, чем семантически не связанные слова, например «man» и «banana». Поэтому предположение о том, что для точек, близких в пространстве входов, и выходы будут близки, неявно подразумеваемое в моделях логистической регрессии, неверно.

Стандартный способ решения этой проблемы – использование **погружений слов** (word embeddings), когда каждый разреженный унитарный вектор $\mathbf{x}_{nt} \in \{0, 1\}^V$ отображается в плотный вектор меньшей размерности, $\mathbf{e}_{nt} \in \mathbb{R}^K$ посредством преобразования $\mathbf{e}_{nt} = \mathbf{E}\mathbf{x}_{nt}$, где матрица \mathbf{E} обучена таким образом, что семантически родственные слова оказываются рядом. В разделе 20.5 мы увидим, что существует много способов обучиться такому погружению.

Имея матрицу погружений, мы можем представить текстовый документ переменной длины в виде **мешка погружений слов**. Затем этот мешок преобразуется в вектор фиксированной длины путем суммирования (или усреднения) погружений:

$$\bar{\mathbf{e}}_n = \sum_{t=1}^T \mathbf{e}_{nt} = \mathbf{E}\tilde{\mathbf{x}}_n, \quad (1.39)$$

где $\tilde{\mathbf{x}}_n$ – представление в виде мешка слов из формулы (1.37). Далее этот вектор можно использовать в классификаторе методом логистической регрессии, который мы кратко описали в разделе 1.2.1.5. Итоговая модель принимает вид:

$$p(y = c | \mathbf{x}_n; \boldsymbol{\theta}) = \mathcal{S}_c(\mathbf{W}\mathbf{E}\tilde{\mathbf{x}}_n). \quad (1.40)$$

Мы часто используем **предварительно обученную матрицу погружений \mathbf{E}** , и в этом случае модель линейна относительно \mathbf{W} , что упрощает оценивание параметров (см. главу 10). Смотрите также обсуждение контекстуального погружения слов в разделе 15.7.

1.5.4.4. Обработка новых слов

Во время тестирования модели могут встретиться совершенно новые слова, которые раньше не предъявлялись. Такие слова называются **внесловарными**

ми (out of vocabulary – **OOV**). Их появление неизбежно, потому что множество слов – **открытый класс**. Так, множество имен собственных (имен людей и названий мест) не ограничено.

Стандартная эвристика для решения этой проблемы – заменить все новые слова специальным символом **UNK** (unknown – неизвестно). Но при этом теряется информация. Например, встретив слово «athazagoraphobia», мы можем догадаться, что это «боязнь чего-то», поскольку производный от греческого слова суффикс «phobia» часто встречается в английском языке и означает «боязнь» (кстати говоря, «атазагорафобия» означает боязнь быть забытым или проигнорированным).

Мы могли бы работать на уровне символов, но тогда пришлось бы обучать модель, как группировать частые комбинации букв в слова. Лучше воспользоваться тем фактом, что у слов есть внутренняя структура, и брать в качестве входных данных **подслова**, или **части слов** [SHB16; Wu+16]; они часто создаются методом **кодирования пар байтов** [Gag94], который представляет собой вариант сжатия данных, когда для представления общих подстрок создаются новые символы.

1.5.5. Обработка отсутствующих данных

Иногда некоторые данные отсутствуют, т. е. части входа **x** или выхода **y** неизвестны. Если во время обучения выход неизвестен, то пример останется не помеченным; такие сценарии обучения с частичным привлечением учителя рассматриваются в разделе 19.3. Поэтому мы сосредоточимся на случае, когда отсутствуют некоторые входные признаки, – на этапе обучения, этапе тестирования или на обоих.

Для моделирования этой ситуации обозначим **M** матрицу бинарных переменных размера $N \times D$, в которой $M_{nd} = 1$, если признак d в примере n присутствует, и $M_{nd} = 0$ в противном случае. Пусть X_v – присутствующие части матрицы входных признаков, соответствующие элементам $M_{nd} = 0$, а X_h – отсутствующие части, соответствующие элементам $M_{nd} = 1$. Обозначим **Y** матрицу выходных меток и предположим, что она наблюдаема целиком. Если $p(\mathbf{M} | X_v, X_h, \mathbf{Y}) = p(\mathbf{M})$, то говорят, что данные **отсутствуют вполне случайно** (missing completely at random – **MCAR**), поскольку присутствие или отсутствие данных вообще не зависит от скрытых или наблюдаемых признаков. Если $p(\mathbf{M} | X_v, X_h, \mathbf{Y}) = p(\mathbf{M} | X_v, \mathbf{Y})$, то говорят, что данные **отсутствуют случайно** (missing at random – **MAR**), поскольку отсутствие не зависит от скрытых признаков, но может зависеть от наблюдаемых. Если ни то, ни другое условие не выполняется, то говорят, что данные **отсутствуют не случайно** (not missing at random – **NMAR**).

В случаях MCAR и MAR мы можем игнорировать механизм отсутствия, поскольку он ничего не говорит о скрытых признаках. Но в случае NMAR необходимо смоделировать **механизм отсутствия данных**, потому что само отсутствие информации может нести полезную информацию. Например, тот факт, что человек не дал ответа на деликатный вопрос в анкете (например, «Болеете ли вы ковидом?»), может дать информацию об истинном значе-

нии. Дополнительные сведения о моделях отсутствия данных см. в работах [LR87; Mar08].

В этой книге мы всегда будем придерживаться предположения MAR. Но даже в этом предположении мы не можем напрямую воспользоваться дискриминантной моделью, например ГНС, при наличии отсутствующих входных признаков, потому что вход x будет иметь неизвестные значения. Стандартная эвристика, применяемая в этом случае, называется **подстановкой среднего значения** – вместо отсутствующих значений включаются их эмпирические средние. Вообще, мы можем аппроксимировать входные данные порождающей моделью и использовать эту модель для **восполнения** отсутствующих значений. Подходящие для этой цели порождающие модели мы кратко обсудим в главе 20, а более полно во втором томе этой книги, [Mur22].

1.6. ОБСУЖДЕНИЕ

В этом разделе мы определим место МО и этой книги в более общем контексте.

1.6.1. Связь МО с другими дисциплинами

Есть несколько сообществ, работающих в связанных с МО областях, называемых по-разному. **Прогнозная аналитика** напоминает обучение с учителем (в особенности классификацию и регрессию), но в большей степени относится к бизнес-приложениям. **Добыча данных** охватывает машинное обучение с учителем и без учителя, но применяется в основном к структурированным данным, которые обычно хранятся в крупных коммерческих базах данных. В **науке о данных** (data science) применяются методы машинного обучения и статистики, но много внимания уделяется также другим вопросам, в частности интеграции данных, визуализации данных и работе со специалистами в предметной области, зачастую в итеративном цикле с обратной связью (см., например, [BS17]). Различие между этими дисциплинами преимущественно терминологическое¹.

МО также тесно связано с математической **статистикой**. Действительно, Джерри Фридман, известный профессор статистики в Стенфордском университете, писал²:

[Если бы статистика] с самого начала включала методологию вычислений в качестве основополагающего инструмента, а не просто была удобным способом приложения имеющихся у нас инструментов, то нужды во многих других дисциплинах, связанных с данными [например, МО], и не возникло бы – они просто были бы частью статистики.

— Джерри Фридман [Fri97b]

¹ См. полезный «Глоссарий МО» на страницу <https://developers.google.com/machine-learning/glossary/>.

² Цитируется по <https://brenocon.com/blog/2008/12/statistics-vs-machine-learning-fight/>.

Машинное обучение также связано с **искусственным интеллектом (ИИ)**. Исторически предполагалось, что мы сможем программировать «интеллект» вручную (см., например, [RN10; PM17]), но этот подход не выдержал столкновения с реальностью, в основном потому что явно закодировать все знания, необходимые таким системам, оказалось слишком трудно. Поэтому возродился интерес к МО как к средству помочь ИИ в приобретении собственных знаний. (На самом деле связи между МО и ИИ настолько тесные, что зачастую эти термины употребляют как синонимы, хотя это, пожалуй, только вводит в заблуждение [Pre21].)

1.6.2. Структура книги

Мы видели, что МО тесно связано со многими другими разделами математики, статистики, информатики и т. д. Поэтому трудно решить, с чего начать.

В этой книге мы выбрали один конкретный путь по этому ландшафту взаимосвязанных дисциплин, используя теорию вероятностей в качестве объединяющей призмы. В части I мы излагаем основы статистики, в частях II–IV рассматриваем обучение с учителем, а в части V – обучение без учителя. Дополнительные сведения об этих (и других) вопросах читатель найдет во втором томе этой книги, [Mur22],

В дополнение к книге на сайте <http://probml.ai> имеются Python-блокноты.

1.6.3. Подводные камни

В этой книге мы увидим, как можно использовать машинное обучение для создания систем, которые могут (или хотя бы пытаются) предсказывать выходы по заданным входам, а затем использовать предсказания для выбора действий с целью минимизировать ожидаемую потерю. При разработке таких систем бывает трудно спроектировать функцию потерь, которые точно отражает все наши предпочтения; это может приводить к **«манипулированию вознаграждением»** (reward hacking) – машина усердно оптимизирует заданную нами функцию вознаграждения, а потом обнаруживается, что функция не улавливает различных ограничений или предпочтений, которые мы забыли описать [Wei76; Amo+16; D'A+20]. (Это особенно важно, когда необходимо искать компромиссы между несколькими целями.)

Манипулирование вознаграждением поднимает различные вопросы в контексте **этики** и **безопасности ИИ** (см., например, [KR19; Lia20]). В работе [Rus19] предлагается решать эту проблему, отказавшись от задания функции вознаграждения, а заставив вместо этого машину выводить вознаграждение путем наблюдения за поведением человека; этот подход называется **обратным обучением с подкреплением**. Однако слишком точное подражание текущему или прошлому поведению человека может быть нежелательно, на него могут повлиять доступные для обучения данные (см., например, [Pau+20]).

Другой подход – рассматривать МО как инструмент для построения адаптивных компонентов, являющихся частями более крупной системы. Такую систему следует проектировать и регулировать так же, как мы проектируем и регулируем другие сложные полуавтономные продукты человеческой деятельности: самолеты, платформы для интернет-трейдинга или системы медицинской диагностики (см. [Jor19]). МО играет важную роль в таких системах, но необходимы дополнительные сдержки и противовесы, чтобы закодировать наши предшествующие знания и предпочтения и гарантировать, что система будет действовать так, как запланировано.