



---

# Оглавление

Отзывы о книге.....	18
<b>Введение.....</b>	<b>22</b>
Для кого эта книга.....	22
Что нужно знать.....	23
Чему вы научитесь.....	23
<b>Предисловие.....</b>	<b>25</b>
От издательства.....	26

## ЧАСТЬ I ГЛУБОКОЕ ОБУЧЕНИЕ НА ПРАКТИКЕ

<b>Глава 1. Путешествие в мир глубокого обучения.....</b>	<b>28</b>
Глубокое обучение для всех.....	28
Нейронные сети: краткая история.....	30
Кто мы.....	33
Как изучать глубокое обучение.....	35
Ваши проекты и мышление.....	37
ПО: PyTorch, fastai и Jupyter (почему это важно).....	38
Ваша первая модель.....	40
Настройка сервера глубокого обучения на GPU.....	40
Запуск первого блокнота.....	42
Что такое машинное обучение.....	46
Что такое нейронная сеть.....	50
Немного терминологии глубокого обучения.....	51
Характерные для ML ограничения.....	52
Как работает наш распознаватель изображений.....	54
Чему научился наш распознаватель изображений.....	61

---

Распознаватели изображений для других задач .....	64
Обобщение терминов .....	68
Глубокое обучение подходит не только для классификации изображений ....	70
Контрольные и тестовые выборки.....	77
Создавайте тестовую выборку обдуманно.....	79
Момент выбора собственного приключения .....	83
Вопросник.....	83
Дополнительные задания .....	85
<b>Глава 2. От модели к продакшену .....</b>	<b>86</b>
Практика глубокого обучения.....	86
Начало проекта .....	87
Текущий уровень глубокого обучения .....	89
Подход Drivetrain .....	93
Сбор данных .....	95
От данных к DataLoaders.....	100
Аугментация данных .....	105
Обучение модели и ее использование для чистки данных.....	105
Превращение модели в онлайн-приложение.....	109
Использование модели для вывода .....	109
Создание в блокноте приложения на основе модели .....	110
Превращение блокнота в реальное приложение.....	113
Развертывание приложения .....	114
Как избежать катастрофы.....	117
Непредвиденные последствия и петли обратной связи.....	120
Записывайте!.....	121
Вопросник.....	122
Дополнительные задания .....	124
<b>Глава 3. Этика данных .....</b>	<b>125</b>
Ключевые примеры этики данных .....	126
Баги и оказание помощи: неисправный алгоритм, распределявший медицинские льготы .....	127
Петли обратной связи: рекомендательная система YouTube.....	127
Предвзятость: «Арест» профессора Латаньи Суини.....	128
Почему это важно? .....	129

Тесное взаимодействие процессов ML и дизайна продукта.....	132
Темы этики данных.....	134
Защита прав и ответственность.....	134
Петли обратной связи.....	135
Необъективность.....	138
Дезинформация.....	150
Выявление этических проблем и их решение.....	152
Анализ проекта.....	152
Какие процессы нужно реализовать.....	153
Сила разнообразия.....	155
Справедливость, ответственность и прозрачность.....	157
Роль политики.....	159
Эффективность регулирования.....	159
Права и политика.....	161
Автомобили: исторический прецедент.....	161
Резюме.....	162
Вопросник.....	163
Дополнительные задания.....	164
Глубокое обучение на практике: итог!.....	165

## **ЧАСТЬ II**

### **ПОНИМАНИЕ ПРИЛОЖЕНИЙ НА БАЗЕ FASTAI**

<b>Глава 4. Обучение классификатора цифр: взгляд изнутри .....</b>	<b>168</b>
Пиксели: основа компьютерного зрения.....	168
Первая попытка: сходство пикселей.....	172
Массивы NumPy и тензоры PyTorch.....	178
Вычисление метрик с помощью бродкастинга (Broadcasting).....	180
Стохастический градиентный спуск.....	184
Вычисление градиентов.....	189
Определение шагов скорости обучения.....	191
Сквозной пример SGD.....	193
Подведение итогов темы градиентного спуска.....	198
Функция потерь MNIST.....	199
Сигмоида.....	205

SGD и мини-пакеты .....	206
Собрать все вместе .....	208
Создание оптимизатора .....	211
Добавление нелинейности .....	213
Углубляемся .....	217
Сводка терминов .....	218
Вопросник.....	220
Дополнительные задания .....	221
<b>Глава 5. Классификация изображений .....</b>	<b>222</b>
От собак и кошек к породам домашних животных.....	222
Подготовка размера.....	226
Проверка и отладка DataBlock .....	229
Перекрестная энтропия .....	231
Активации и метки.....	232
Softmax .....	233
Логарифмическая функция правдоподобия .....	236
Применение логарифма.....	238
Интерпретация модели .....	241
Улучшение модели.....	242
Поиск скорости обучения .....	242
Разморозка и перенос обучения.....	245
Дискриминативные скорости обучения.....	248
Выбор количества эпох .....	250
Углубленные архитектуры.....	251
Резюме.....	253
Вопросник.....	254
Дополнительные задания .....	255
<b>Глава 6. Другие задачи компьютерного зрения .....</b>	<b>256</b>
Классификация по нескольким меткам .....	256
Данные.....	257
Построение DataBlock .....	259
Бинарная перекрестная энтропия .....	264
Регрессия .....	269

---

Сборка данных.....	270
Обучение модели .....	273
Резюме.....	275
Вопросник.....	276
Дополнительные задания .....	277
<b>Глава 7. Обучение современной модели .....</b>	<b>278</b>
Imagenette .....	278
Нормализация .....	280
Прогрессивное изменение размера .....	282
Аугментация во время тестирования .....	284
Mixup .....	286
Сглаживание меток.....	289
Резюме.....	292
Вопросник.....	292
Дополнительные задания .....	293
<b>Глава 8. Коллаборативная фильтрация .....</b>	<b>294</b>
Первый взгляд на данные.....	295
Обучение скрытых факторов.....	297
Создание DataLoaders .....	299
Коллаборативная фильтрация с нуля .....	302
Сокращение весов .....	306
Создание собственного модуля вложений .....	307
Интерпретация вложений и смещений.....	309
Использование fastai.collab .....	311
Расстояние между вложениями.....	312
Бутстрэппинг модели коллаборативной фильтрации.....	312
Глубокое обучение для коллаборативной фильтрации .....	314
Резюме.....	317
Вопросник.....	317
Дополнительные задания .....	319
<b>Глава 9. Табличное моделирование .....</b>	<b>320</b>
Категориальные вложения .....	320
За гранью глубокого обучения .....	326

---

Датасет.....	328
Соревнования Kaggle.....	328
Знакомство с данными.....	330
Деревья решений.....	331
Обработка дат.....	333
Использование TabularPandas и TabularProc.....	334
Создание дерева решений.....	337
Категориальные переменные.....	342
Случайные леса.....	343
Создание случайного леса.....	344
Ошибка Out-of-Bag.....	346
Интерпретация модели.....	347
Дисперсия деревьев для уверенного прогнозирования.....	348
Важность признаков.....	349
Удаление переменных с низкой важностью.....	350
Удаление лишних признаков.....	351
Частичная зависимость.....	354
Утечка данных.....	357
Интерпретатор деревьев.....	358
Экстраполяция и нейронные сети.....	360
Проблема экстраполяции.....	361
Поиск несоответствующих области данных.....	362
Использование нейронной сети.....	365
Ансамблирование.....	369
Бустинг.....	370
Совмещение вложений с другими методами.....	371
Резюме.....	372
Вопросник.....	373
Дополнительные задания.....	375
<b>Глава 10. Погружение в NLP: рекуррентные нейронные сети.....</b>	<b>376</b>
Предварительная обработка текста.....	378
Токенизация.....	379
Токенизация слов с помощью fastai.....	380
Токенизация подслов.....	384

Нумеризация с помощью fastai.....	385
Разделение текстов на пакеты.....	387
Обучение классификатора текста.....	390
Создание языковой модели с помощью DataBlock.....	391
Тонкая настройка языковой модели .....	392
Сохранение и загрузка моделей .....	393
Генерация текста.....	395
Создание DataLoaders классификатора .....	395
Тонкая настройка классификатора .....	398
Дезинформация и языковые модели .....	399
Резюме.....	402
Вопросник.....	403
Дополнительные задания .....	404
<b>Глава 11. Преобразование данных с помощью Mid-Level API .....</b>	<b>405</b>
Знакомство с многослойным API .....	405
Преобразования.....	406
Написание собственного преобразования.....	408
Pipeline.....	409
TfmdLists и датасеты: преобразованные коллекции.....	410
TfmdLists .....	410
Datasets.....	412
Использование промежуточного API: SiamesePair .....	414
Резюме.....	419
Вопросник.....	419
Дополнительные задания .....	420
Сферы применения fastai: обобщение .....	420

## **ЧАСТЬ III ОСНОВЫ ГЛУБОКОГО ОБУЧЕНИЯ**

<b>Глава 12. Языковая модель с нуля .....</b>	<b>422</b>
Данные .....	422
Первая языковая модель с нуля.....	424
Языковая модель в PyTorch.....	425
Первая рекуррентная нейронная сеть.....	428



---

Улучшение RNN .....	430
Управление состоянием RNN .....	430
Создание дополнительного сигнала .....	433
Многослойные RNN .....	436
Модель .....	437
Взрывающиеся или исчезающие активации .....	438
LSTM .....	439
Создание LSTM с нуля .....	440
Обучение языковой модели с помощью LSTM .....	442
Регуляризация LSTM .....	444
Dropout .....	444
Регуляризация активаций и регуляризация временных активаций .....	447
Обучение регуляризованной LSTM со связанными весами .....	447
Резюме .....	449
Вопросник .....	450
Дополнительные задания .....	452
<b>Глава 13. Сверточные нейронные сети .....</b>	<b>453</b>
Магия сверток .....	453
Отображение ядра свертки .....	457
Свертки в PyTorch .....	459
Штрихи и заполнение .....	461
Понимание сверточных уравнений .....	462
Первая сверточная нейронная сеть .....	465
Создание CNN .....	465
Разъяснение арифметики сверток .....	468
Рецептивные поля .....	469
О Twitter .....	471
Цветные изображения .....	473
Повышение стабильности обучения .....	476
Базовая модель .....	477
Увеличение размера пакета .....	480
Обучение 1 cycle .....	480
Пакетная нормализация .....	485
Резюме .....	489

---

Вопросник.....	489
Дополнительные задания .....	491
<b>Глава 14. ResNet.....</b>	<b>492</b>
Возвращение к Imagenette .....	492
Построение современной CNN: ResNet .....	496
Пропускающие соединения .....	496
Актуальная ResNet .....	503
Зауженные слои .....	506
Резюме.....	508
Вопросник.....	509
Дополнительные задания .....	510
<b>Глава 15. Архитектуры приложений .....</b>	<b>511</b>
Компьютерное зрение .....	511
cnn_learner.....	511
unet_learner.....	513
Сиамская сеть.....	516
Обработка естественного языка .....	518
Табличные модели .....	519
Резюме.....	520
Вопросник.....	522
Дополнительные задания .....	523
<b>Глава 16. Процесс обучения .....</b>	<b>524</b>
Создание базовой модели .....	524
Универсальный оптимизатор .....	526
Импульс.....	527
RMSProp .....	530
Adam.....	531
Раздельное сокращение весов .....	532
Обратные вызовы.....	533
Создание обратного вызова.....	536
Упорядочивание обратных вызовов и исключения.....	539
Резюме.....	540
Вопросник.....	541

Дополнительные задания .....	542
Основы глубокого обучения: итог .....	542

## **ЧАСТЬ IV ГЛУБОКОЕ ОБУЧЕНИЕ С ЧИСТОГО ЛИСТА**

<b>Глава 17. Продвинутые основы нейронной сети .....</b>	<b>544</b>
Создание слоя нейронной сети с нуля .....	544
Моделирование нейрона .....	544
Матричное умножение .....	546
Поэлементная арифметика .....	547
Уширение (broadcasting) .....	549
Соглашение Эйнштейна .....	553
Прямой и обратный проход .....	555
Определение и инициализация слоя .....	555
Градиенты и обратный проход .....	560
Рефакторинг модели .....	563
Переходим в PyTorch .....	564
Резюме .....	567
Вопросник .....	568
Дополнительные задания .....	570
<b>Глава 18. Интерпретация CNN с помощью CAM .....</b>	<b>571</b>
CAM и хуки .....	571
CAM градиентов .....	575
Резюме .....	577
Вопросник .....	577
Дополнительные задания .....	578
<b>Глава 19. Класс Learner с нуля .....</b>	<b>579</b>
Данные .....	579
Dataset .....	581
Module и Parameter .....	584
Простая CNN .....	587
Функция потерь .....	588

Learner .....	590
Обратные вызовы.....	591
Планирование скорости обучения .....	593
Резюме.....	595
Вопросник.....	595
Дополнительные задания .....	597
<b>Глава 20. Подведем итог .....</b>	<b>598</b>

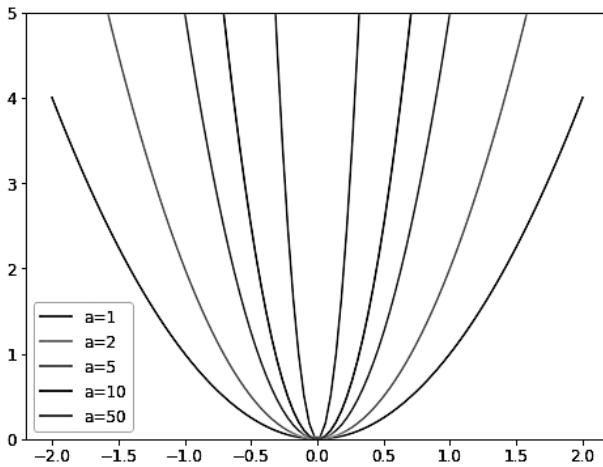
## ПРИЛОЖЕНИЯ

<b>Приложение А. Создание блога.....</b>	<b>602</b>
Блогинг на GitHub Pages.....	602
Создание репозитория .....	603
Настройка домашней страницы .....	604
Создание публикаций .....	606
Синхронизация GitHub и компьютера.....	608
Блогинг из Jupyter .....	610
<b>Приложение Б. Схема подготовки проекта по аналитике данных.....</b>	<b>611</b>
Специалисты по данным.....	612
Стратегия.....	613
Данные .....	615
Аналитика .....	616
Реализация .....	616
Обслуживание .....	617
Ограничения.....	618
Об авторах.....	619
Благодарности .....	620
Об обложке .....	622

## Сокращение весов

Сокращение весов, иначе называемое *регуляризацией L2*, представляет собой добавление к функции потерь, возведенной в квадрат, суммы всех весов. Зачем нам это делать? Потому что при вычислении градиентов это будет добавлять им вклад, способствующий максимальному сокращению весов.

Почему это должно предотвратить переобучение? Суть в том, что чем больше коэффициенты, тем круче спуски, получаемые в функции потерь. Если взять простой пример параболы  $y = a * (x^{**2})$ , то чем больше  $a$ , тем более *узкой* получается парабола:



Поэтому если позволить модели обучать высокие параметры, то она подстроит все точки данных обучающей выборки с помощью сверхсложной функции, имеющей резкие изменения, что приведет к переобучению.

Препятствуя чрезмерному увеличению весов, мы замедлим обучение модели, но она окажется в состоянии, способствующем лучшей обобщаемости. Если коротко вернуться к теории, то сокращение весов (или просто  $wd$ ) — это параметр, контролирующий сумму квадратов, добавляемых нами к потерям (предполагая, что  $parameters$  является тензором всех параметров):

```
loss_with_wd = loss + wd * (parameters**2).sum()
```

Тем не менее на практике будет очень неэффективно (а может, и численно нестабильно) вычислять настолько большую сумму и добавлять ее к потерям. Вы можете припомнить из курса высшей математики, что производная  $p^{**2}$  по отношению к  $p$  равна  $2*p$ . Поэтому добавление такой большой суммы к потерям эквивалентно следующему:

```
parameters.grad += wd * 2 * parameters
```

В реальности, так как `wd` является выбираемым нами параметром, мы можем его удвоить, исключив из данного уравнения `*2`. Для использования сокращения весов в `fastai` нужно передать `wd` в вызов `fit` или `fit_one_cycle` (можно передать в оба):

```
model = DotProductBias(n_users, n_movies, 50)
learn = Learner(dls, model, loss_func=MSELossFlat())
learn.fit_one_cycle(5, 5e-3, wd=0.1)
```

epoch	train_loss	valid_loss	time
0	0.972090	0.962366	00:13
1	0.875591	0.885106	00:13
2	0.723798	0.839880	00:13
3	0.586002	0.823225	00:13
4	0.490980	0.823060	00:13

Намного лучше!

## Создание собственного модуля вложений

До сих пор мы использовали `Embedding`, не обращая внимания на то, как он фактически работает. Давайте воссоздадим `DotProductBias` без применения этого класса. Нам понадобится матрица случайным образом инициализированных весов для каждого из вложений. При этом нужно быть осторожными. В главе 4 мы писали, что оптимизаторы требуют возможности получения всех параметров модуля из его метода `parameters`. Тем не менее это происходит не полностью автоматически. Если мы просто добавим тензор к `Module` в качестве атрибута, то в `parameters` он включен не будет:

```
class T(Module):
    def __init__(self): self.a = torch.ones(3)
```

```
L(T()).parameters()
(#0) []
```

Чтобы сообщить `Module` о своем желании рассматривать тензор как параметр, нужно обернуть его в класс `nn.Parameter`. Этот класс не привносит никакую функциональность (за исключением автоматического вызова `requires_grad`). Он просто используется как «маркер», показывающий, что нужно включить в `parameters`:

```
class T(Module):
    def __init__(self): self.a = nn.Parameter(torch.ones(3))
```

```
L(T().parameters())
(#1) [Parameter containing:
tensor([1., 1., 1.], requires_grad=True)]
```

Все модули PyTorch задействуют `nn.Parameter` для всех обучаемых параметров, почему нам и не требовалось специально использовать эту обертку до текущего момента:

```
class T(Module):
    def __init__(self): self.a = nn.Linear(1, 3, bias=False)

t = T()
L(t.parameters())

(#1) [Parameter containing:
tensor([[ -0.9595],
        [-0.8490],
         0.8159]], requires_grad=True)]

type(t.a.weight)

torch.nn.parameter.Parameter
```

Мы можем создать тензор в качестве параметра, используя случайную инициализацию:

```
def create_params(size):
    return nn.Parameter(torch.zeros(*size).normal_(0, 0.01))
```

Используем это для повторного создания `DotProductBias`, но без `Embedding`:

```
class DotProductBias(Module):
    def __init__(self, n_users, n_movies, n_factors, y_range=(0,5.5)):
        self.user_factors = create_params([n_users, n_factors])
        self.user_bias = create_params([n_users])
        self.movie_factors = create_params([n_movies, n_factors])
        self.movie_bias = create_params([n_movies])
        self.y_range = y_range

    def forward(self, x):
        users = self.user_factors[x[:,0]]
        movies = self.movie_factors[x[:,1]]
        res = (users*movies).sum(dim=1)
        res += self.user_bias[x[:,0]] + self.movie_bias[x[:,1]]
        return sigmoid_range(res, *self.y_range)
```

Еще раз проведем обучение, чтобы проверить, получим ли мы такой же результат, что и в предыдущем разделе:

```
model = DotProductBias(n_users, n_movies, 50)
learn = Learner(dls, model, loss_func=MSELossFlat())
learn.fit_one_cycle(5, 5e-3, wd=0.1)
```

epoch	train_loss	valid_loss	time
0	0.962146	0.936952	00:14
1	0.858084	0.884951	00:14
2	0.740883	0.838549	00:14
3	0.592497	0.823599	00:14
4	0.473570	0.824263	00:14

А теперь посмотрим, чему наша модель научилась.

## Интерпретация вложений и смещений

Наша модель уже достаточно эффективна в своей способности предоставлять рекомендации для наших пользователей, но при этом очень интересно узнать, какие же параметры она обнаружила. Проще всего интерпретировать смещения. Вот фильмы с наименьшими значениями в векторе смещений:

```
movie_bias = learn.model.movie_bias.squeeze()
idxs = movie_bias.argsort()[:5]
[dls.classes['title'][i] for i in idxs]

['Children of the Corn: The Gathering (1996)',
 'Lawnmower Man 2: Beyond Cyberspace (1996)',
 'Beautician and the Beast, The (1997)',
 'Crow: City of Angels, The (1996)',
 'Home Alone 3 (1997)']
```

Подумайте о том, что это значит. Здесь говорится, что, несмотря на высокое соответствие пользователя латентным факторам приведенных фильмов (которые, как мы вскоре увидим, представляют уровень экшена, возрастную категорию фильма и т. д.), сам фильм, как правило, пользователю не нравится. Мы могли бы просто отсортировать фильмы непосредственно по их средней оценке, но, глядя на обученное смещение, становится понятно кое-что более интересное. Это говорит нам не только о том, что фильм относится к категории тех, которые пользователи не любят смотреть, но и о том, что пользователи не хотели бы смотреть такой фильм, даже если он относится к жанру, который их интересует. А вот аналогичный результат, но уже для фильмов с самым высоким смещением:

```
idxs = movie_bias.argsort(descending=True)[:5]
[dls.classes['title'][i] for i in idxs]

['L.A. Confidential (1997)',
 'Titanic (1997)',
 'Silence of the Lambs, The (1991)',
 'Shawshank Redemption, The (1994)',
 'Star Wars (1977)']
```



