ОГЛАВЛЕНИЕ

9

СОВЕРШЕНСТВОВАНИЕ ГЛУБОКИХ СЕТЕЙ

В главе 6 мы детально исследовали отдельные искусственные нейроны. В главе 7 объединили их в сеть, организовав прямое распространение некоторых входных данных x через сеть для получения некоторого прогноза $\hat{y}$. А совсем недавно, в главе 8, мы узнали, как количественно оценить ошибки сети (сравнить оценку $\hat{y}$ с истинным значением y с помощью функции стоимости), а также как минимизировать эти ошибки (настроить параметры сети w и b с применением оптимизаторов — стохастического градиентного спуска и обратного распространения).

Теперь мы рассмотрим типичные препятствия, встречающиеся на пути создания высококачественных нейронных сетей, и методы их преодоления. Мы применим эти идеи при разработке нашей первой глубокой нейронной сети[1]. Объединив увеличенную глубину сети с новыми приемами, мы посмотрим, можно ли превзойти более простые и мелкие архитектуры из предыдущих глав в точности классификации рукописных цифр.

ИНИЦИАЛИЗАЦИЯ ВЕСОВ

В главе 8 мы познакомились с эффектом насыщения нейронов (см. рис. 8.1), когда очень низкие или очень высокие значения z ухудшают способность нейрона к обучению. Там же было предложено решение в виде функции стоимости на основе перекрестной энтропии. Перекрестная энтропия существенно ослабляет влияние эффекта насыщения нейронов, но, объединив ее с вдумчивой *инициализацией весов*, можно еще больше уменьшить вероятность появления эффекта насыщения. Как отмечалось в сноске в главе 1, применение усовершенствованных методов

[1] Как рассказывалось в главе 4, *глубокой* называется нейронная сеть, состоящая по крайней мере из трех скрытых слоев.

инициализации весов обеспечило существенный скачок в развитии сферы глубокого обучения: это одно из знаковых теоретических достижений, сделанных между LeNet-5 (см. рис. 1.11) и AlexNet (см. рис. 1.17), которые значительно расширили диапазон задач, решаемых с помощью искусственных нейронных сетей. В этом разделе мы поэкспериментируем с несколькими подходами к инициализации весов, чтобы вы смогли понять, насколько они эффективны.

Описывая порядок обучения нейронной сети в главе 8, мы упоминали, что параметры w и b инициализируются случайными значениями, из-за чего начальная аппроксимация сети оказывается далекой от цели и, соответственно, имеет высокую начальную стоимость C. Вообще говоря, нет большой нужды подробно рассматривать эту проблему, потому что за кулисами библиотека Keras создает модели TensorFlow, которые инициализируются вполне разумными значениями w и b. Тем не менее мы обсудим ее, чтобы вы не только знали о существовании еще одного метода предотвращения насыщения нейронов, но и восполнили пробел в вашем понимании особенностей обучения нейронных сетей. Библиотека Keras берет на себя всю тяжелую работу по выбору начальных значений, и это главное ее преимущество; но вы должны понимать, что возможно, а иногда необходимо изменить эти значения по умолчанию, чтобы добиться лучшего соответствия вашей задаче.

Чтобы чтение этого раздела проходило более увлекательно, мы советуем использовать наш блокнот Jupyter *weight_initialization.ipynb*. Как показано в следующем фрагменте кода, блокнот зависит от NumPy (библиотека числовых операций), matplotlib (библиотека для построения графиков) и нескольких методов Keras, которые мы подробно рассмотрим в этом разделе.

```
import numpy as np
import matplotlib.pyplot as plt
from keras import Sequential
from keras.layers import Dense, Activation
from keras.initializers import Zeros, RandomNormal
from keras.initializers import glorot_normal, glorot_uniform
```

В этом блокноте мы моделируем 784-пиксельные значения, которые служат входными данными для одного полносвязанного слоя искусственных нейронов. Такая размерность входных данных, конечно же, была выбрана по образу и подобию наших любимых цифр из коллекции MNIST (рис. 5.3). Для полносвязанного слоя мы выбрали достаточно большое число нейронов (256), чтобы потом, когда мы будем строить графики, у нас имелось достаточно данных:

```
n_input = 784
n_dense = 256
```

Основной темой этого раздела является инициализация параметров сети w и b. Прежде чем начать передавать обучающие данные в сеть, желательно определить разумные начальные значения параметров. Тому есть две причины.

1. Большие значения w и b, как правило, приводят к большим значениям z и, соответственно, к появлению эффекта насыщения нейронов (см. график на рис. 8.1).
2. Большие значения параметров означают, что сеть имеет четкое представление о том, как входные данные x связаны с истинными значениями y, но было бы странно строить любые предположения до обучения на фактических данных.

Нулевые значения параметров, с другой стороны, предполагают весьма расплывчатое представление о взаимосвязях между x и y. Проводя аналогию со сказкой «Три медведя», наша цель — уподобиться Машеньке и начать обучение со сбалансированными и *пригодными к обучению* исходными значениями параметров. По этой причине, проектируя архитектуру нейронной сети, мы использовали метод `Zeros()` для инициализации нейронов полносвязанного слоя $b = 0$:

```
b_init = Zeros()
```

Продолжая рассуждения, начатые в предыдущем абзаце, можно подумать, что веса w сети тоже следует инициализировать нулями. Но в действительности это привело бы к катастрофе: при одинаковых значениях весов и смещений многие нейроны в сети будут одинаково интерпретировать одни и те же входные данные x, мешая стохастическому градиентному спуску (SGD) определять индивидуальные настройки параметров, способствующие снижению стоимости C. Продуктивнее было бы инициализировать веса разными значениями из некоторого диапазона, чтобы каждый нейрон обрабатывал входные данные x уникально и давал алгоритму SGD широкий выбор начальных точек для аппроксимации y. По теории вероятностей, начальные результаты некоторых нейронов могут внести верный вклад в разумное отображение x в y. Сначала этот вклад будет слабым, но у SGD появится шанс поэкспериментировать с ним и определить, может ли он способствовать снижению стоимости C между прогнозируемым $\hat{y}$ и целевым значением y.

Как отмечалось ранее (например, при обсуждении рис. 7.5 и 8.9), подавляющее большинство параметров в типичной сети составляют веса, а на смещения приходится относительно небольшая их часть. Поэтому допустимо (в действительности это наиболее распространенная практика) инициализировать смещения нулями, а веса — случайными значениями, *близкими* к нулю. Один из самых простых способов сгенерировать случайные значения, близкие к нулю, — выбирать их из стандартного нормального распределения[1], как в листинге 9.1.

Листинг 9.1. Инициализация весов значениями, выбираемыми из стандартного нормального распределения

```
w_init = RandomNormal(stddev=1.0)
```

[1] Нормальное распределение также называют распределением Гаусса или, в разговорной речи, «колоколообразной кривой» из-за соответствующей формы графика. *Стандартное нормальное* распределение — это нормальное распределение со средним значением 0 и стандартным отклонением 1.

Чтобы оценить влияние выбранного способа инициализации весов, в листинге 9.2 создается архитектура нейронной сети с единственным полносвязанным слоем сигмоидных нейронов.

Листинг 9.2. Архитектура с единственным полносвязанным слоем сигмоидных нейронов

```
model = Sequential()
model.add(Dense(n_dense,
                input_dim=n_input,
                kernel_initializer=w_init,
                bias_initializer=b_init))
model.add(Activation('sigmoid'))
```

Как и во всех предыдущих примерах, модель создается с помощью `Sequential()`. Затем используется метод `add()` для создания одного полносвязанного слоя со следующими параметрами:

- 256 нейронов (`n_dense`);
- 784 входа (`n_input`);
- в `kernel_initializer` передается массив `w_init` для инициализации весов сети желаемыми значениями — в данном случае выбранными из стандартного нормального распределения;
- в `bias_initializer` передается массив `b_init` для инициализации смещений нулевыми значениями.

Чтобы упростить корректировку параметров далее в этом разделе, отдельно добавляем в слой сигмоидную функцию активации вызовом `Activation('sigmoid')`.

После настройки сети вызывается метод `random()` из библиотеки NumPy, чтобы сгенерировать 784 «значения пикселов» — случайные числа с плавающей точкой из диапазона [0.0, 1.0):

```
x = np.random.random((1,n_input))
```

Далее вызывается метод `predict()` для прямого распространения x через один слой и получения активаций `a`:

```
a = model.predict(x)
```

В заключение генерируется гистограмма, иллюстрирующая распределение активаций[1]:

```
_ = plt.hist(np.transpose(a))
```

[1] Для тех, кому интересно, что означает символ подчеркивания (`_ =`), поясняем, что этот прием помогает сохранить блокнот Jupyter более аккуратным и просто вывести график, без создания объекта, хранящего этот график.

Полученные нами результаты показаны на рис. 9.1. Ваши результаты будут немного отличаться от наших из-за метода `random()`, который использовался для получения входных значений, но в целом они должны выглядеть примерно похожими.

Как и ожидалось (см. рис. 6.9), активации a на выходе слоя сигмоидных нейронов ограничены диапазоном от 0 до 1. Однако в их распределении отмечается нежелательная закономерность — большая их часть сосредоточена по краям диапазона: они примыкают либо к 0, либо к 1. Это указывает на то, что при использовании нормального распределения для инициализации весов w слоя наши искусственные нейроны склонны производить большие значения z. Это нежелательно по двум причинам, упомянутым выше в этом разделе:

1. Подавляющее большинство нейронов в слое подвержено эффекту насыщения.
2. Нейроны выражают твердое мнение о том, как x влияет на y *еще до* обучения на данных.

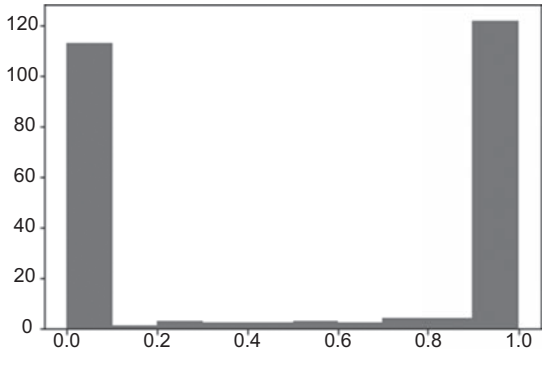

Рис. 9.1. Гистограмма распределения активаций на выходе слоя сигмоидных нейронов с весами, инициализированными с использованием стандартного нормального распределения

К счастью, эту проблему легко устранить, инициализировав веса сети значениями, выбранными из альтернативных распределений.

РАСПРЕДЕЛЕНИЯ КСАВЬЕ ГЛОРО

В сфере глубокого обучения большой популярностью для выборки начальных значений весов пользуются распределения, разработанные Ксавье Глоро (Xavier Glorot) и Йошуа Бенжио[1], портрет которого представлен на рис. 1.10.

[1] *Glorot X. & Bengio Y.* (2010). «Understanding the difficulty of training deep feedforward neural networks». Proceedings of Machine Learning Research, 9, 249–56.