Part I

深度学习基础

Chapter 1

什么是深度学习

Chapter 2

神经网络的数学基础

2.1 初识神经网络

神经网络的核心组件是**层**（layer），它是一种数据处理模块，你可以将它看成数据过滤器。具体来说，层从输入数据中提取表示——我们期望这种表示有助于解决手头的问题。大多数深度学习都是将简单的层链接起来，从而实现渐进式的**数据蒸馏**（data distillation）。深度学习模型就像是数据处理的筛子，包含一系列越来越精细的数据过滤器（即层）。

要想训练网络，我们还需要选择**编译**（compile）步骤的三个参数。

- 损失函数（loss function）：网络如何衡量在训练数据上的性能，即网络如何朝着正确的方向前进。
- 优化器（optimizer）：基于训练数据和损失函数来更新网络的机制。
- 在训练和测试过程中需要监控的指标（metric）。

2.2 神经网络的数据表示

2.2.1 关键属性

张量是由以下三个关键属性来定义的：

- 轴的个数（阶）。例如，3D 张量有 3 个轴，矩阵有 2 个轴。这在 Numpy 等 Python 库中也叫张量的 ndim。
- 形状。这是一个整数元组，表示张量沿每个轴的维度大小（元素个数）。例如，前面矩阵示例的形状为 (3, 5)，3D 张量示例的形状为 (3, 3, 5)。向量的形状只包含一个元素，比如 (5,)，而标量的形状为空，即 ()。
- 数据类型（在 Python 库中通常叫作 dtype）。这是张量中所包含数据的类型，例如，张量的类型可以是 float32、uint8、float64 等。在极少数情况下，你可能会遇到字符（char）张量。注意，Numpy（以及大多数其他库）中不存在字符串张量，因为张量存储在预先分配的连续内存段中，而字符串的长度是可变的，无法用这种方式存储。

2.2.2 数据批量的概念

通常来说，深度学习中所有数据张量的第一个轴（0 轴，因为索引从 0 开始）都是样本轴（samples axis，有时也叫样本维度）。

此外，深度学习模型不会同时处理整个数据集，而是将数据拆分成小批量。

2.2.3 现实世界中的数据张量

我们用几个你未来会遇到的示例来具体介绍数据张量。你需要处理的数据几乎总是以下类别之一。

- 向量数据：2D 张量，形状为 (samples, features)。
- 时间序列数据或序列数据：3D 张量，形状为 (samples, timesteps, features)。
- 图像：4D 张量，形状为 (samples, height, width, channels) 或 (samples, channels, height, width)。
- 视频：5D 张量，形状为 (samples, frames, height, width, channels) 或 (samples, frames, channels, height, width)。

图像数据

图像通常具有三个维度：高度、宽度和颜色深度。虽然灰度图像（比如 MNIST 数字图像）只有一个颜色通道，因此可以保存在 2D 张量中，但按照惯例，图像张量始终都是 3D 张量，灰度图像的彩色通道只有一维。

图像张量的形状有两种约定：**通道在后**（channels-last）的约定（在 TensorFlow 中使用）和**通道在前**（channels-first）的约定（在 Theano 中使用）。Google 的 TensorFlow 机器学习框架将颜色深度轴放在最后：(samples, height, width, color_depth)。与此相反，Theano将图像深度轴放在批量轴之后：(samples, color_depth, height, width)。Keras 框架同时支持这两种格式。

Chapter 3

Introduction to Keras and TensorFlow

3.1 神经网络剖析

3.1.1 层：深度学习的基础组件

神经网络的基本数据结构是层。层是一个数据处理模块，将一个或多个输入张量转换为一个或多个输出张量。有些层是无状态的，但大多数的层是有状态的，即层的权重。权重是利用随机梯度下降学到的一个或多个张量，其中包含网络的知识。不同的张量格式与不同的数据处理类型需要用到不同的层。例如，简单的向量数据保存在形状为 (samples, features) 的 2D 张量中，通常用密集连接层[densely connected layer，也叫全连接层（fully connected layer）或密集层（dense layer），对应于 Keras 的 Dense 类]来处理。序列数据保存在形状为 (samples, timesteps, features) 的 3D 张量中，通常用循环层（recurrent layer，比如 Keras 的 LSTM 层）来处理。图像数据保存在 4D 张量中，通常用二维卷积层（Keras 的 Conv2D）来处理。

层兼容性（layer compatibility）具体指的是每一层只接受特定形状的输入张量，并返回特定形状的输出张量。

3.2 First steps with TensorFlow

3.2.1 Constant tensors and variables

A significant difference between NumPy arrays and TensorFlow tensors is that TensorFlow tensors aren't assignable: they're constant.

To train a model, we'll need to update its state, which is a set of tensors. If tensors aren't assignable, how do we do it? That's where variables come in. tf.Variable is the class meant to manage modifiable state in TensorFlow.

Similarly, assign_add() and assign_sub() are efficient equivalents of += and -=.

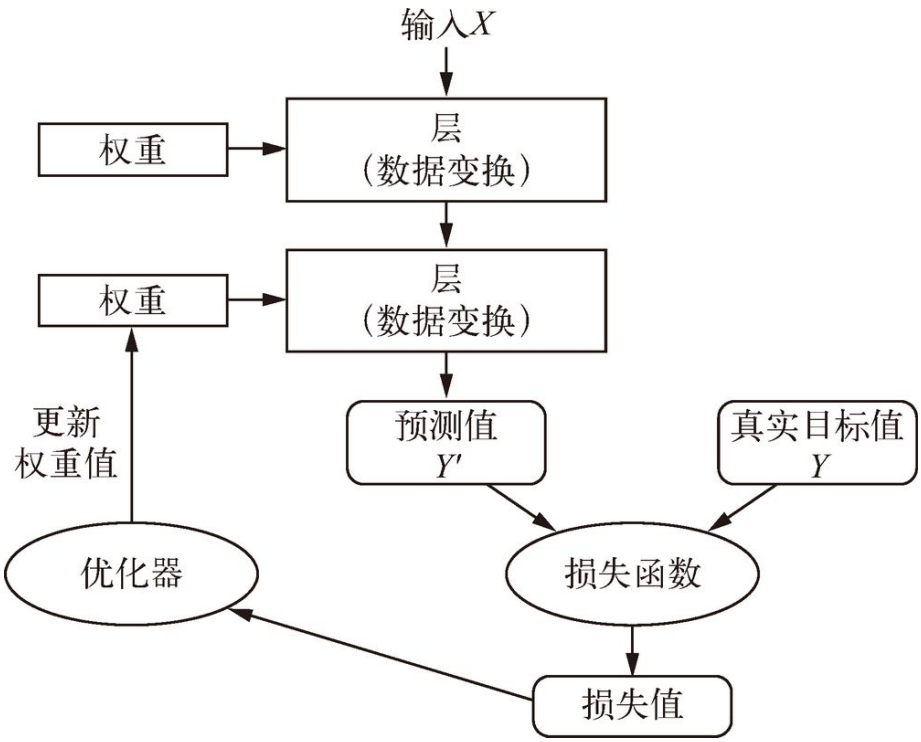

图 3.1: 多个层链接在一起组成了网络，将输入数据映射为预测值。然后损失函数将这些预测值与目标进行比较，得到损失值，用于衡量网络预测值与预期结果的匹配程度。优化器使用这个损失值来更新网络的权重。

THE BASE LAYER CLASS IN KERAS

A simple API should have a single abstraction around which everything is centered. In Keras, that's the Layer class. Everything in Keras is either a Layer or something that closely interacts with a Layer.

A Layer is an object that encapsulates some state (weights) and some computation (a forward pass). The weights are typically defined in a build() (although they could also be created in the constructor, __init__()), and the computation is defined in the call() method.

Part II

深度学习实战