Chapter 1

神经网络的复习

1.1 神经网络的学习

1.1.1 损失函数

我们来介绍一下 Softmax 函数和交叉熵误差。首先，Softmax 函数可由下式表示：

$$y_k = \frac{\exp s_k}{\sum_{i=1}^{n} \exp s_i} \tag{1.1}$$

此时，交叉熵误差可由下式表示：

$$L = -\sum_k t_k \log y_k \tag{1.2}$$

这里，t_k 是对应于第 k 个类别的监督标签。监督标签以 one-hot 向量的形式表示，比如 $\boldsymbol{t} = (0, 0, 1)$。

另外，在考虑了 mini-batch 处理的情况下，交叉熵误差可以由下式表示：

$$L = -\frac{1}{N} \sum_n \sum_k t_{nk} \log y_{nk} \tag{1.3}$$

这里假设数据有 N 笔，t_{nk} 表示第 n 笔数据的第 k 维元素的值，y_{nk} 表示神经网络的输出，t_{nk} 表示监督标签。通过这样的平均化，无论 mini-batch 的大小如何，都始终可以获得一致的指标。

1.1.2 梯度与导数

严格地说，这里使用的“梯度”一词与数学中的“梯度”是不同的。数学中的梯度仅限于关于向量的导数。而在深度学习领域，一般也会定义关于矩阵和张量的导数，称为“梯度”。

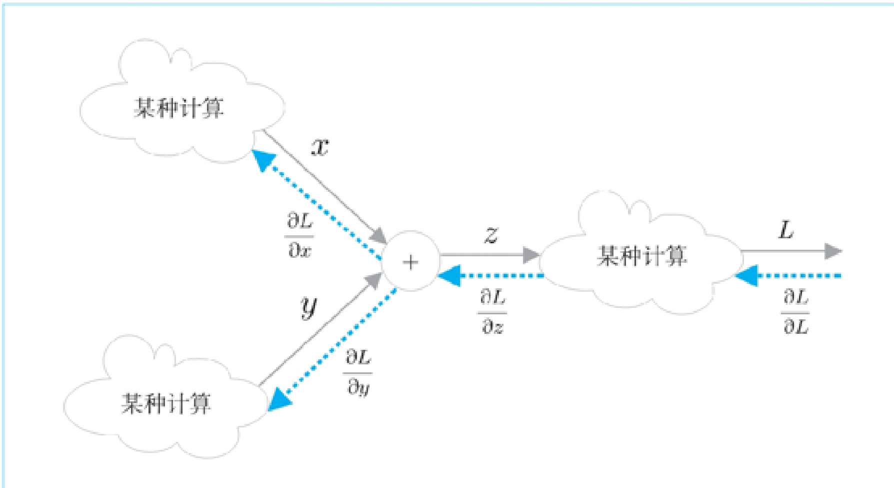

图 1.1: 计算图的反向传播

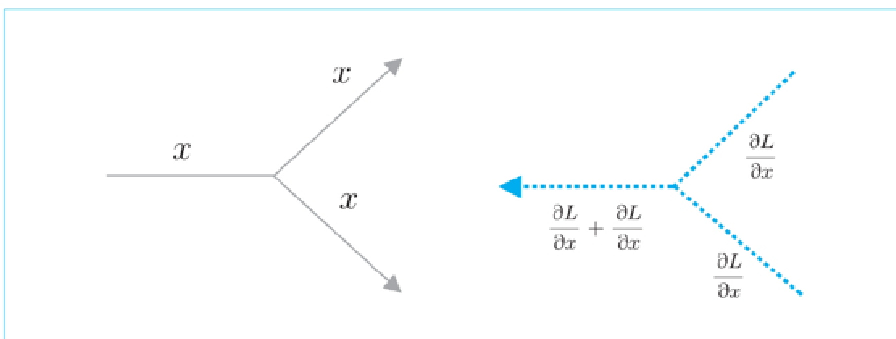

图 1.2: 分支节点的正向传播（左图）和反向传播（右图）

1.1.3 链式法则

链式法则的重要之处在于，无论我们要处理的函数有多复杂（无论复合了多少个函数），都可以根据它们各自的导数来求复合函数的导数。也就是说，只要能够计算各个函数的局部的导数，就能基于它们的积计算最终的整体的导数。

分支节点

严格来说，分支节点并没有节点，只有两根分开的线。此时，相同的值被复制并分叉。因此，分支节点也称为复制节点。如 Figure 1.2 所示，它的反向传播是上游传来的梯度之和。

Repeat 节点

分支节点有两个分支，但也可以扩展为 N 个分支（副本），这里称为 Repeat 节点。现在，我们尝试用计算图绘制一个 Repeat 节点（图 Figure 1.3）。

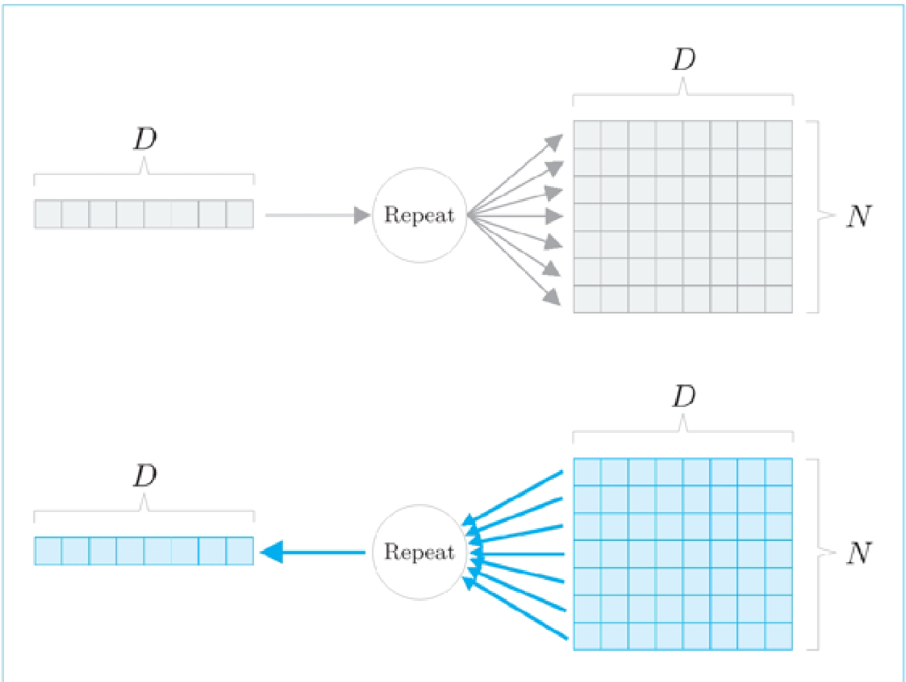

图 1.3: Repeat 节点的正向传播（上图）和反向传播（下图）

Sum 节点

Sum 节点是通用的加法节点。这里考虑对一个 $N \times D$ 的数组沿第 0 个轴求和。此时，Sum 节点的正向传播和反向传播如图 Figure 1.4 所示。有趣的是，Sum 节点和 Repeat 节点存在逆向关系。所谓逆向关系，是指 Sum 节点的正向传播相当于 Repeat 节点的反向传播，Sum 节点的反向传播相当于 Repeat 节点的正向传播。

MatMul 节点

考虑 $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{W}$ 这个计算，这里，$\boldsymbol{x}$、 $\boldsymbol{W}$、$\boldsymbol{y}$ 的形状分别是 $1 \times D$、$D \times H$、$1 \times H$。此时，可以按如下方式求得关于 $\boldsymbol{x}$ 的第 i 个元素的导数 $\frac{\partial L}{\partial x_i}$。

$$\frac{\partial L}{\partial x_i} = \sum_j \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial x_i} \tag{1.4}$$

利用 $\frac{\partial y_j}{\partial x_i} = W_{ij}$，将其代入 Equation 1.4：

$$\frac{\partial L}{\partial x_i} = \sum_j \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial x_i} = \sum_j \frac{\partial L}{\partial y_j} W_{ij} \tag{1.5}$$

$\frac{\partial L}{\partial x_i}$ 由向量 $\frac{\partial L}{\partial \boldsymbol{y}}$ 和 $\boldsymbol{W}$ 的第 i 行向量的内积求得。从这个关系可以导出下式：

$$\frac{\partial L}{\partial \boldsymbol{x}} = \frac{\partial L}{\partial \boldsymbol{y}} \boldsymbol{W}^T \tag{1.6}$$

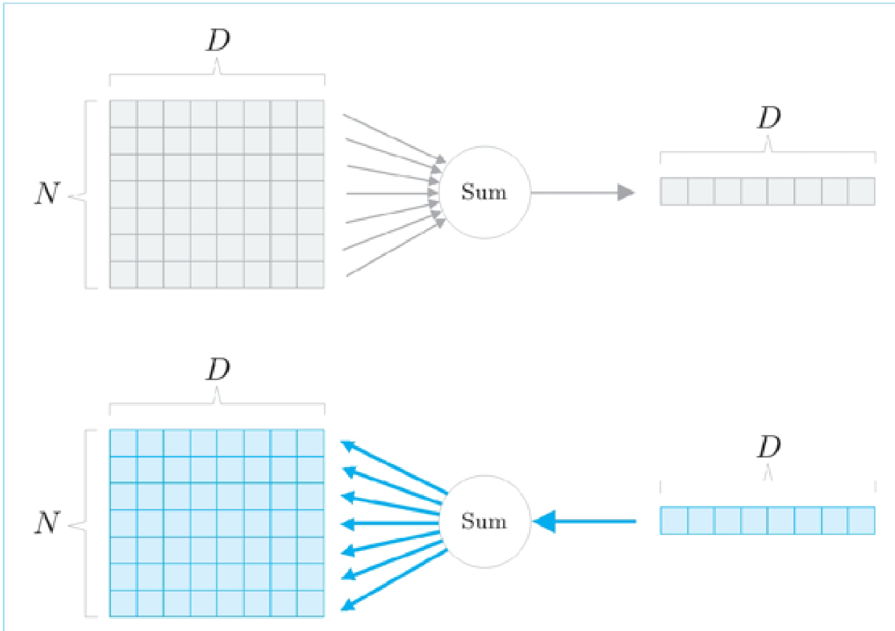

图 1.4: Sum 节点的正向传播（上图）和反向传播（下图）

和省略号一样，这里也可以进行基于 `grads[0] = dW` 的赋值。不同的是，在使用省略号的情况下会覆盖掉 NumPy 数组。这是浅复制（shallow copy）和深复制（deep copy）的差异。`grads[0] = dW` 的赋值相当于浅复制，`grads[0][...] = dW` 的覆盖相当于深复制。

浅复制中，`a` 的指向发生改变（指向的内存地址改变），而深复制中 `a` 的指向没有发生改变，只是指向的内存地址内的值改变（Figure 1.6）。

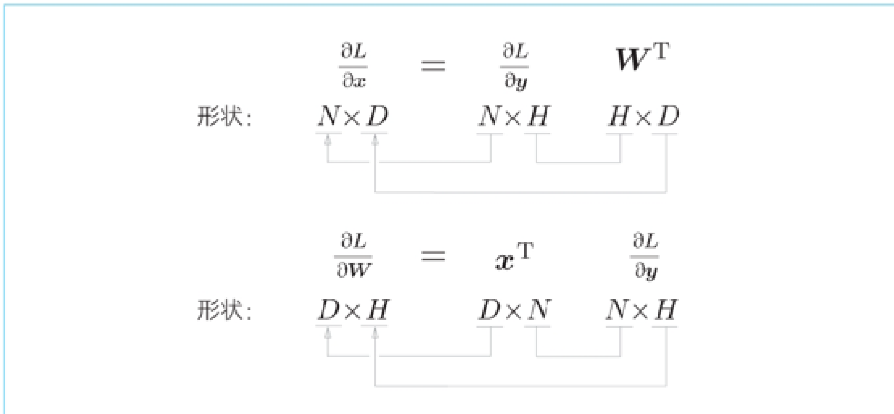

图 1.5: 通过确认矩阵形状，推导反向传播的数学式

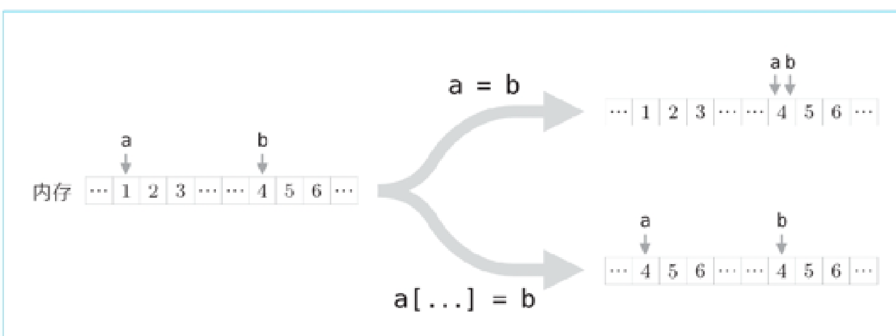

图 1.6: `a = b` 和 `a[...]  = b` 的区别：使用省略号时数据被覆盖，变量指向的内存地址不变

Chapter 2