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”。

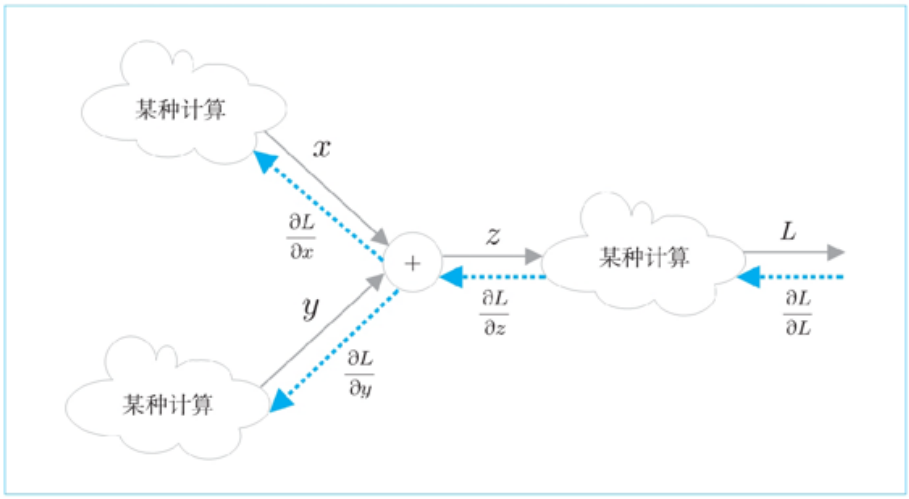

Figure 1.1: 计算图的反向传播

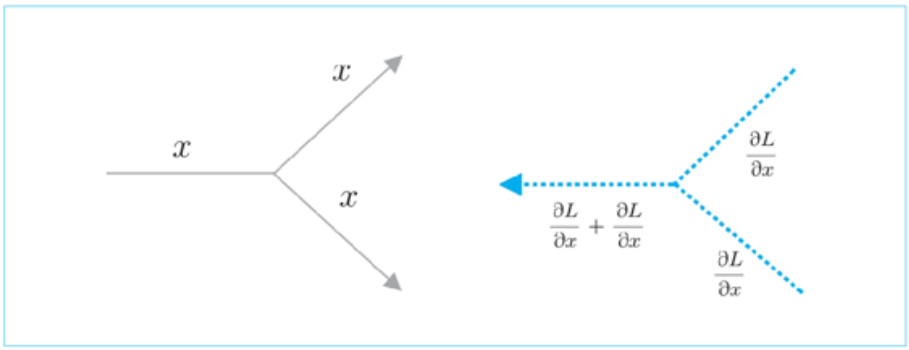

Figure 1.2: 分支节点的正向传播（左图）和反向传播（右图）

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）。

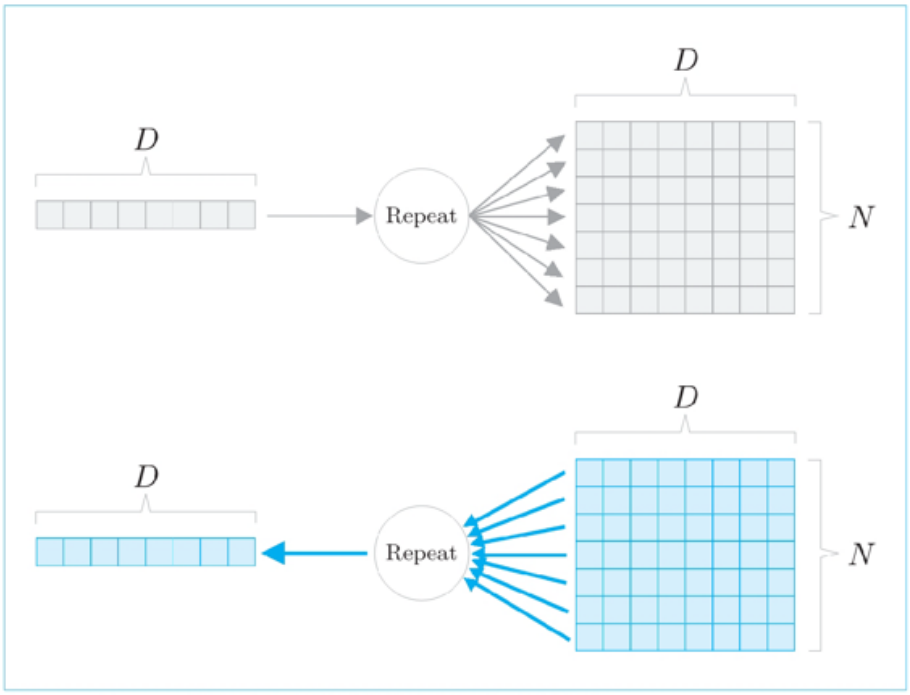

Figure 1.3: Repeat 节点的正向传播（上图）和反向传播（下图）

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：

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

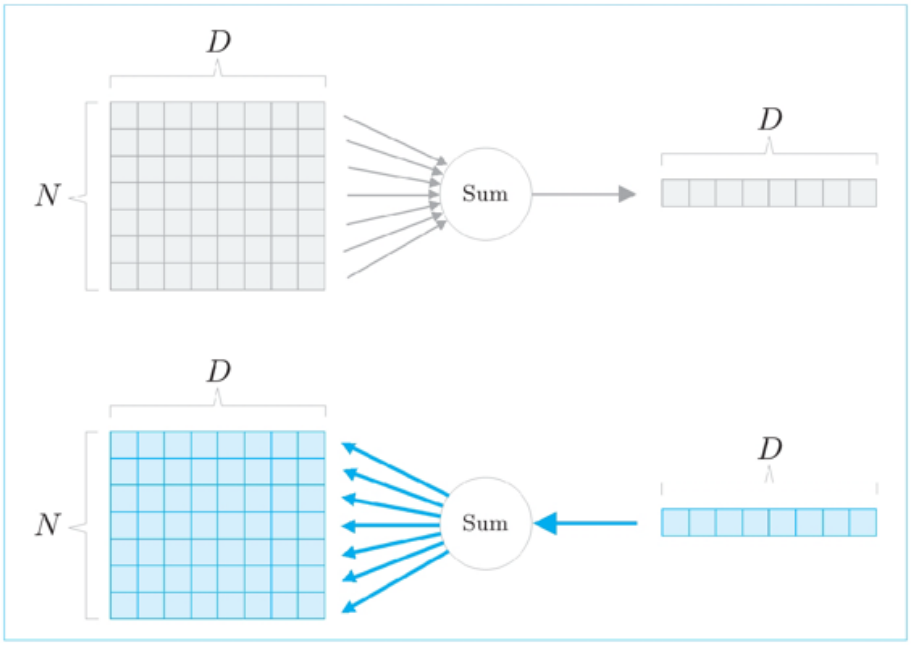

Figure 1.4: Sum 节点的正向传播（上图）和反向传播（下图）

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）。

1.2 计算高速化

1.2.1 位精度

我们已经知道，如果只是神经网络的推理，则即使使用 16 位浮点数进行计算，精度也基本上不会下降。不过，虽然 NumPy 中准备有16 位浮点数，但是普通 CPU 或 GPU 中的运算是用 32 位执行的。因此，即便变换为 16 位浮点数，因为计算本身还是用 32 位浮点数执行的，所以处理速度方面并不能获得什么好处。但是，如果是要（在外部文件中）保存学习好的权重，则 16 位浮点数是有用的。具体地说，将权重数据用 16 位精度保存时，只需要 32 位时的一半容量。

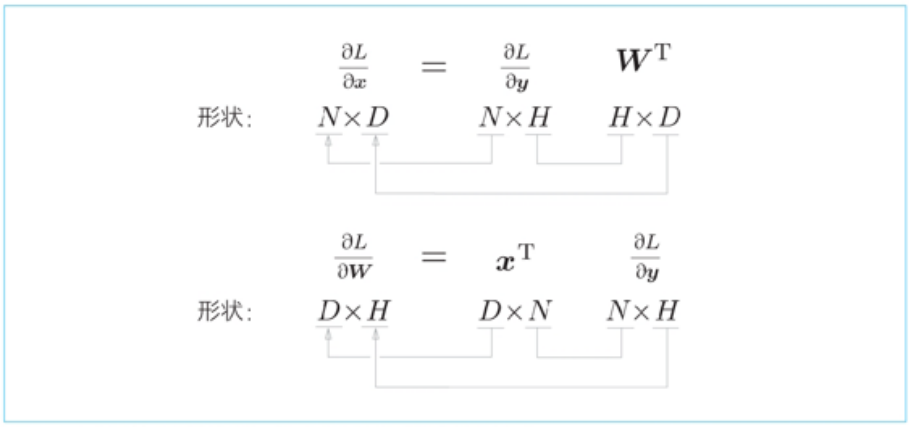

Figure 1.5: 通过确认矩阵形状，推导反向传播的数学式

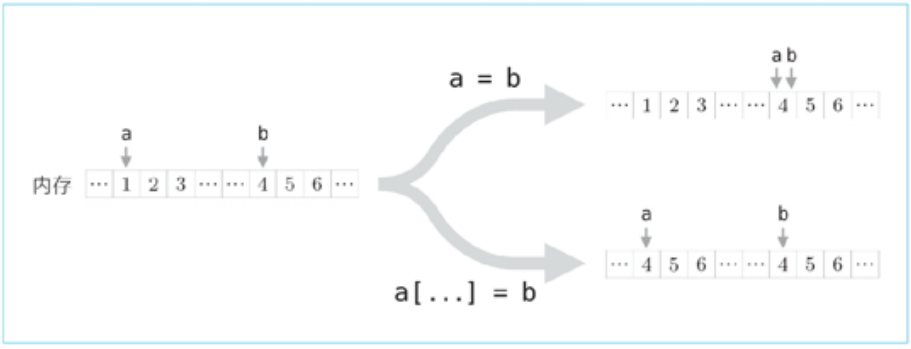

Figure 1.6: `a = b` 和 `a[...] = b` 的区别：使用省略号时数据被覆盖，变量指向的内存地址不变

Chapter 2

自然语言和单词的分布式表示

2.1 同义词词典

回顾自然语言处理的历史，人们已经尝试过很多次类似这样的人工定义单词含义的活动。但是，目前被广泛使用的并不是《新华字典》那样的常规词典，而是一种被称为同义词词典（thesaurus）的词典。在同义词词典中，具有相同含义的单词（同义词）或含义类似的单词（近义词）被归类到同一个组中。比如，使用，我们可以知道 car 的同义词有 automobile、motorcar 等

2.1.1 WordNet

在自然语言处理领域，最著名的同义词词典是 WordNet。使用 WordNet，可以获得单词的近义词，或者利用单词网络。使用单词网络，可以计算单词之间的相似度。

2.1.2 同义词词典的问题

- 难以顺应时代变化
- 人力成本高
- 无法表示单词的微妙差异

2.2 基于计数的方法

从介绍基于计数的方法开始，我们将使用语料库（corpus）。简而言之，语料库就是大量的文本数据。不过，语料库并不是胡乱收集数据，一般收集的都是用于自然语言处理研究和应用的文本数据。

自然语言处理领域中使用的语料库有时会给文本数据添加额外的信息。比如，可以给文本数据的各个单词标记词性。在这种情况下，为了方便计算机处理，语料库通常会被结构化（比

	you	say	goodbye	and	i	hello	.
you	0	1	0	0	0	0	0
say	1	0	1	0	1	1	0
goodbye	0	1	0	1	0	0	0
and	0	0	1	0	1	0	0
i	0	1	0	1	0	0	0
hello	0	1	0	0	0	0	1
.	0	0	0	0	0	1	0

Figure 2.1: 用表格汇总各个单词的上下文中包含的单词的频数

如，采用树结构等数据形式）。这里，假定我们使用的语料库没有添加标签，而是作为一个大的文本文件，只包含简单的文本数据。

能不能将类似于颜色的向量表示方法运用到单词上呢？更准确地说，可否在单词领域构建紧凑合理的向量表示呢？接下来，我们将关注能准确把握单词含义的向量表示。在自然语言处理领域，这称为分布式表示。

单词的分布式表示将单词表示为固定长度的向量。这种向量的特征在于它是用密集向量表示的。密集向量的意思是，向量的各个元素（大多数）是由非0实数表示的。例如，三维分布式表示是 $[0.21, -0.45, 0.83]$。

2.2.1 分布式假设

“某个单词的含义由它周围的单词形成”，称为分布式假设（distributional hypothesis）。分布式假设所表达的理念非常简单。单词本身没有含义，单词含义由它所在的上下文（语境）形成。上下文是指某个居中单词的周围词汇。这里，我们将上下文的大小（即周围的单词有多少个）称为**窗口大小**（window size）。窗口大小为 1，上下文包含左右各 1 个单词；窗口大小为 2，上下文包含左右各 2 个单词，以此类推。

2.2.2 共现矩阵

2.2.3 向量间的相似度

测量向量间的相似度有很多方法，其中具有代表性的方法有向量内积或欧式距离等。虽然除此之外还有很多方法，但是在测量单词的向量表示的相似度方面，**余弦相似度**（cosine similarity）是

很常用的。设有 $\boldsymbol{x} = (x_1, x_2, x_3, \cdots, x_n)$ 和 $\boldsymbol{y} = (y_1, y_2, y_3, \cdots, y_n)$ 两个向量，它们之间的余弦相似度的定义如下式所示：

$$similarity(\boldsymbol{x}, \boldsymbol{y}) = \frac{x_1 y_1 + \cdots + x_n y_n}{\sqrt{x_1^2 + \cdots + x_n^2}\sqrt{y_1^2 + \cdots + y_n^2}} \tag{2.1}$$

余弦相似度直观地表示了“两个向量在多大程度上指向同一方向”。两个向量完全指向相同的方向时，余弦相似度为 1；完全指向相反的方向时，余弦相似度为 -1。

2.3 基于计数的方法的改进

2.3.1 点互信息

共现矩阵的元素表示两个单词同时出现的次数。但是，这种“原始”的次数并不具备好的性质。如果我们看一下高频词汇（出现次数很多的单词），就能明白其原因了。比如，我们来考虑某个语料库中 the 和 car 共现的情况。这意味着，仅仅因为 the 是个常用词，它就被认为与 car 有很强的相关性。为了解决这一问题，可以使用点互信息（Pointwise Mutual Information，PMI）这一指标。对于随机变量 x 和 y，它们的 PMI 定义如下：

$$PMI(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)} \tag{2.2}$$

其中，$P(x)$ 表示 x 发生的概率，$P(y)$ 表示 y 发生的概率，$P(x, y)$ 表示 x 和 y 同时发生的概率。PMI 的值越高，表明相关性越强。在自然语言的例子中，$P(x)$ 就是指单词 x 在语料库中出现的概率。

虽然我们已经获得了PMI这样一个好的指标，但是 PMI 也有一个问题。那就是当两个单词的共现次数为 0 时，$\log_2 0 = -\infty$。为了解决这个问题，实践上我们会使用下述正的点互信息（Positive PMI，PPMI）：

$$PPMI(x, y) = \max(0, PMI(x, y)) \tag{2.3}$$

根据式 Equation 2.3，当 PMI 是负数时，将其视为 0，这样就可以将单词间的相关性表示为大于等于 0 的实数。

但是，这个 PPMI 矩阵还是存在一个很大的问题，那就是随着语料库的词汇量增加，各个单词向量的维数也会增加。而且，其中很多元素都是 0。这表明向量中的绝大多数元素并不重要，也就是说，每个元素拥有的“重要性”很低。另外，这样的向量也容易受到噪声影响，稳健性差。

2.3.2 降维

向量中的大多数元素为 0 的矩阵（或向量）称为稀疏矩阵（或稀疏向量）。这里的重点是，从稀疏向量中找出重要的轴，用更少的维度对其进行重新表示。结果，稀疏矩阵就会被转化为大多数元素均不为0 的密集矩阵。这个密集矩阵就是我们想要的单词的分布式表示。

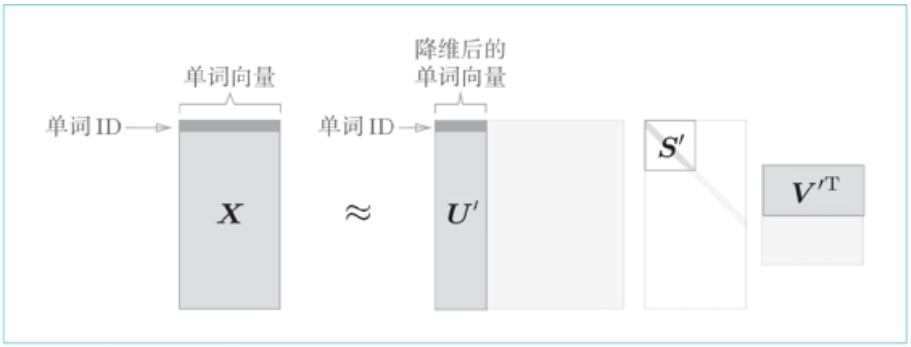

Figure 2.2: 基于 SVD 的降维示意图（白色部分表示元素为 0）

降维的方法有很多，这里我们使用奇异值分解（Singular Value Decomposition，SVD）。SVD 将任意矩阵分解为 3 个矩阵的乘积，如下式所示：

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \tag{2.4}$$

其中 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 是列向量彼此正交的正交矩阵，$\boldsymbol{S}$ 是除了对角线元素以外其余元素均为 0 的对角矩阵。

> 单词的共现矩阵是正方形矩阵，但在 Figure 2.2 中，为了和之前的图一致，画的是长方形。另外，这里对 SVD 的介绍仅限于最直观的概要性的说明。

2.4 小结

使用基于同义词词典的方法，需要人工逐个定义单词之间的相关性。这样的工作非常费力，在表现力上也存在限制（比如，不能表示细微的差别）。而基于计数的方法从语料库中自动提取单词含义，并将其表示为向量。具体来说，首先创建单词的共现矩阵，将其转化为 PPMI 矩阵，再基于 SVD 降维以提高稳健性，最后获得每个单词的分布式表示。另外，我们已经确认过，这样的分布式表示具有在含义或语法上相似的单词在向量空间上位置相近的性质。

Chapter 3

word2vec

在 自然语言和单词的分布式表示 中，我们使用基于计数的方法得到了单词的分布式表示。本章我们将讨论该方法的替代方法，即基于推理的方法。

3.1 基于推理的方法和神经网络

基于计数的方法使用整个语料库的统计数据（共现矩阵和 PPMI 等），通过一次处理（SVD 等）获得单词的分布式表示。而基于推理的方法使用神经网络，通常在 mini-batch 数据上进行学习。这意味着神经网络一次只需要看一部分学习数据（mini-batch），并反复更新权重。

3.1.1 基于推理的方法的概要

解开 Figure 3.1 中的推理问题并学习规律，就是基于推理的方法的主要任务。通过反复求解这些推理问题，可以学习到单词的出现模式。从“模型视角”出发，这个推理问题如 Figure 3.2 所示。

基于推理的方法和基于计数的方法一样，也基于分布式假设。分布式假设假设“单词含义由其周围的单词构成”。基于推理的方法将这一假设归结为了上面的预测问题。由此可见，不管是哪种方法，如何对基于分布式假设的“单词共现”建模都是最重要的研究主题。

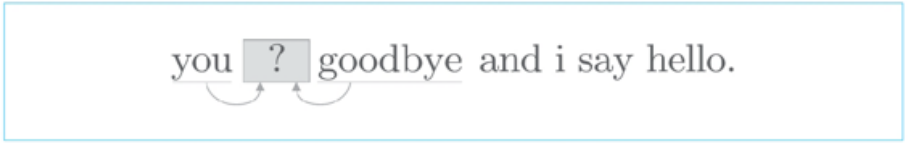

Figure 3.1: 基于两边的单词（上下文），预测“？”处出现什么单词

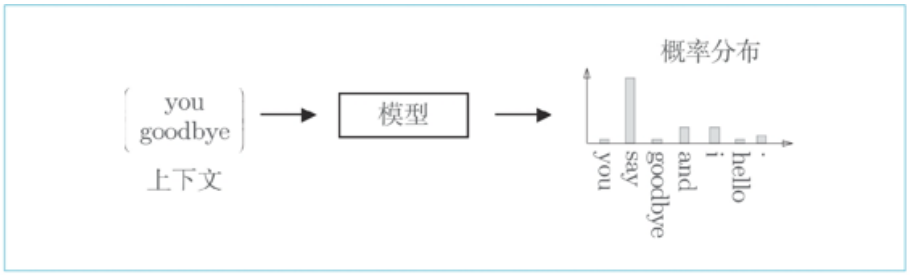

Figure 3.2: 基于推理的方法：输入上下文，模型输出各个单词的出现概率

3.1.2 神经网络中单词的处理方法

神经网络无法直接处理 you 或 say 这样的单词，要用神经网络处理单词，需要先将单词转化为固定长度的向量。对此，一种方式是将单词转换为 one-hot 表示（one-hot 向量）。在 one-hot 表示中，只有一个元素是 1，其他元素都是 0。像这样，只要将单词转化为固定长度的向量，神经网络的输入层的神经元个数就可以固定下来。

3.2 简单的 word2vec

word2vec 一词最初用来指程序或者工具，但是随着该词的流行，在某些语境下，也指神经网络的模型。正确地说，CBOW 模型和 skip-gram 模型是 word2vec 中使用的两个神经网络。

3.2.1 CBOW模型的推理

CBOW 模型的输入是上下文。这个上下文用 `['you', 'goodbye']` 这样的单词列表表示。我们将其转换为 one-hot 表示，以便 CBOW 模型可以进行处理。在此基础上，CBOW 模型的网络可以画成 Figure 3.3 这样。

Figure 3.3 是 CBOW 模型的网络。它有两个输入层（这里，因为我们对上下文仅考虑两个单词，所以输入层有两个。如果对上下文考虑 N 个单词，则输入层会有 N 个。），经过中间层到达输出层。这里，从输入层到中间层的变换由相同的全连接层（权重为 $\boldsymbol{W}_{in}$）完成，从中间层到输出层神经元的变换由另一个全连接层（权重为 $\boldsymbol{W}_{out}$）完成。

中间层的神经元数量比输入层少这一点很重要。中间层需要将预测单词所需的信息压缩保存，从而产生密集的向量表示。这时，中间层被写入了我们人类无法解读的代码，这相当于“编码”工作。而从中间层的信息获得期望结果的过程则称为“解码”。这一过程将被编码的信息复原为我们可以理解的形式。

3.2.2 CBOW 模型的学习

CBOW 模型只是学习语料库中单词的出现模式。如果语料库不一样，学习到的单词的分布式表示也不一样。比如，只使用“体育”相关的文章得到的单词的分布式表示，和只使用“音乐”相关

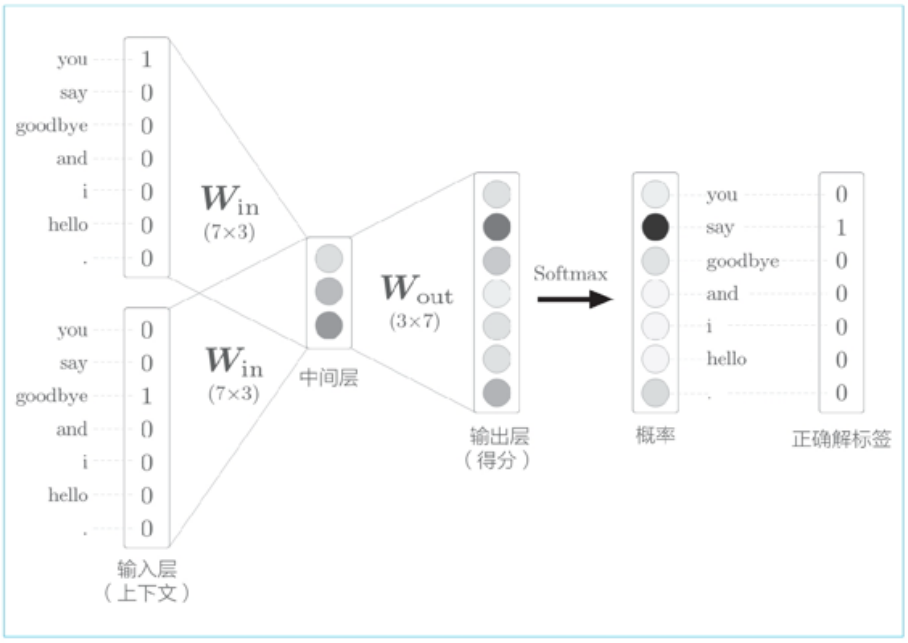

Figure 3.3: CBOW 模型的网络结构

的文章得到的单词的分布式表示将有很大不同。

3.2.3 word2vec 的权重和分布式表示

如前所述，word2vec 中使用的网络有两个权重，分别是输入侧的全连接层的权重（$\boldsymbol{W}_{in}$）和输出侧的全连接层的权重（$\boldsymbol{W}_{out}$）。一般而言，输入侧的权重 $\boldsymbol{W}_{in}$ 的每一行对应于各个单词的分布式表示。另外，输出侧的权重 $\boldsymbol{W}_{out}$ 也同样保存了对单词含义进行了解码的向量。只是，输出侧的权重在列方向上保存了各个单词的分布式表示。

那么，我们最终应该使用哪个权重作为单词的分布式表示呢？这里有三个选项：

A. 只使用输入侧的权重
B. 只使用输出侧的权重
C. 同时使用两个权重

方案 A 和方案 B 只使用其中一个权重。而在采用方案 C 的情况下，根据如何组合这两个权重，存在多种方式，其中一个方式就是简单地将这两个权重相加。

就 word2vec（特别是 skip-gram 模型）而言，最受欢迎的是方案 A。许多研究中也都仅使用输入侧的权重 $\boldsymbol{W}_{in}$ 作为最终的单词的分布式表示。遵循这一思路，我们也使用 $\boldsymbol{W}_{in}$ 作为单词的分布式表示。