案例导读

第5章

神经网络中的特征提取

5.1 神经网络简介

特征提取是指将原始数据转换为具有统计意义和机器可识别的特征。在自然语言处理中，由于机器学习无法直接处理文本，需要将文本转换为数值特征（如向量化）。在图像处理领域，将像素特征提取为轮廓信息也是一种特征提取的应用。因此，特征提取关注的是特征的转换方式，以符合机器学习算法的要求。此外，可以通过对现有特征进行加工的方式来创建新的特征，即特征提取可能是原特征的某种混合。接下来将介绍如何使用不同类型的神经网络进行特征提取。

人工神经网络（Artificial Neural Network，ANN）也被称为神经网络。它是一种广泛应用于解决复杂人工智能问题的大规模并行计算模型。人工神经网络由许多人工神经元组成，并通过分层组织形成大规模平行互联网络。这些神经元通过加权连接相互作用，以达到信息处理的目的。人工神经网络具有良好的学习和泛化能力。其设计灵感源自人类大脑，大脑可以看作一个高度复杂、非线性和并行的信息处理系统。它可以组织其结构成分——神经元，并执行一些计算任务，如模式识别、感知和运动控制。与当今最快的数字计算机相比，大脑的处理速度要快得多。神经网络以与生物神经系统相同的方式与现实世界的对象进行互动，它被设计用来模拟大脑执行特定任务或感兴趣的功能的方式，但人工神经网络中使用的处理元素和架构已经远远超过了生物灵感。神经网络是一个庞大的学科，本书只涉及它与机器学习的交汇点。

5.1.1 生物神经网络

生物神经网络（biological neural network）能够引发动物产生意识并驱使它们采取行动。在生物大脑中，数以千亿计的神经元相互连接，以并行处理信息。每个生物神经元可以看作生物神经网络中的一个较小的处理单元。生物神经元是一种特殊的生物细胞，它在一些电和化学变化的帮助下，将信息从一个神经元传递到另一个神经元。当每个神经元接收信号的累积效果超过神经元的“阈值”时，它就会被激活，向相连的神经元发送“兴奋”或“抑制”的化学物质，从而影响下一个神经元的状态。因此，神经元传递的信息既可以起刺激作用，又可以起抑制作用。生物神经网络的工作流程如图 5-1 所示。

图 5-1　生物神经网络的工作流程

5.1.2　人工神经元

人工神经元是神经网络运行的基础之一，它被用作信息处理单元。与生物神经元类似，人工神经元接收其他人工神经元传递过来的输入信号，将这些输入信号进行加权求和，然后通过激活函数进行转换，以产生输出信号。每个人工神经元可以接收多个输入信号，但只能产生一个信号。神经元模型的基本要素通常包括以下 4 部分。

(1) 连接：每个连接都伴随一个权重，这个权重系数反映了该连接在神经网络中的重要性程度。具体来说，每个输入到神经元 j 的信号 x_j 会乘以权重 w_j。

(2) 累加器：用于将神经元的各个连接加权的输入信号累加。

(3) 激活函数：为神经元引入非线性因素。

(4) 偏置：代表每个神经元的偏好属性。

神经网络通过学习来获取知识，这些知识蕴含在连接权重与偏置中，一个神经元的模型如图 5-2 所示。

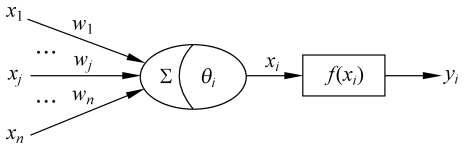


图 5-2　神经元的模型

用数学术语，可以这样来描述一个神经元：

$$y = f\left(\sum_{i=1}^{n} w_i x_i + \theta_i\right) \tag{5-1}$$

其中，x_i 表示第 i 个输入信号；y 是神经元的输出信号；w_i 表示第 i 个输入信号的权重；θ_i 是第 i 个神经元的偏置；$\sum$ 表示求和符号，对所有输入信号进行加权求和；f 为非线性函数，也可以称作激活函数。神经元接收多个信号 x_i，每个输入信号都乘以对应的权重 w_i，并加上偏置 θ_i，然后将它们累加后的结果输入激活函数 $f(\cdot)$中进行非线性变换，得到最终的输出 y。这种神经元具有阈值类型的激活函数，被称为 MP 模型，自 1943 年提出以来沿用至今。

使用激活函数主要有以下两个目的。

(1) 如果不使用激活函数，神经网络每一层输出都是上一层输入的线性组合，因此，无论神经网络有多少层，其输出都可以通过所有输入的线性组合来表示。

(2) 激活函数引入了非线性因素，使得神经网络可以逼近任何非线性函数，从而扩展神经网络的应用到更多的非线性模型中。

激活函数必须具备以下 3 个特征。

(1) 激活函数应该是连续可导的非线性函数(允许在少数点上可以不可导)。可导的激活函数可以通过参数优化的方法来学习网络参数。

(2) 激活函数及其导函数要尽可能地简单，这样可以提高网络计算效率。

(3) 激活函数的导函数的值域要在一个合适的范围内，太大或者太小会影响模型训练的

效率和稳定性。

常见的几个激活函数有以下 3 种，分别是 Sigmoid 函数、Tanh 函数和 ReLU 函数。

(1) Sigmoid 函数。Sigmoid 函数的图像呈现为 S 形状，是迄今为止在构建人工神经网络中最常用的激活函数之一。它的主要特点是将神经元的输出映射到 0～1，因此非常适用于以预测概率作为输出的模型，如二元分类问题。Sigmoid 函数可表示为式(5-2)，其曲线如图 5-3 所示。

$$y = F(x) = \frac{1}{1 + e^{x}} \tag{5-2}$$

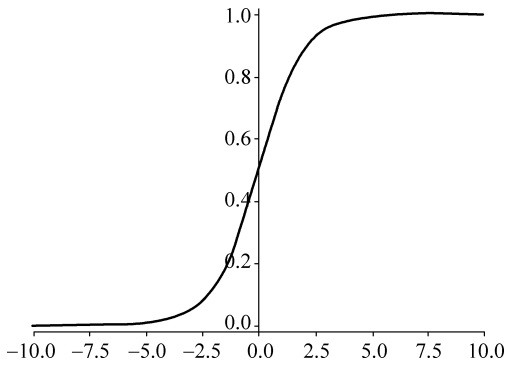


图 5-3　Sigmoid 函数

(2) Tanh 函数。Tanh 函数又称为双曲正切函数。与 Sigmoid 函数类似，Tanh 函数也将输入映射到一个特定范围内，即 −1～1。它克服了 Sigmoid 函数不以 0 为中心输出的问题。它的定义由式(5-3)给出，其曲线如图 5-4 所示。

$$\mathrm{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5-3}$$

(3) ReLU 函数。ReLU 函数即修正线性单元(Rectified Linear Unit)。当输入为正时，该函数直接输出该输入值，不存在梯度饱和问题；当输入小于或等于 0 时，该函数输出 0。因为 ReLU 函数中只存在线性关系，所以它的计算速度比 Sigmoid 函数和 Tanh 函数更快。ReLU 函数可以表示为式(5-4)，其曲线如图 5-5 所示。

$$\mathrm{ReLU}(x) = \max(0, x) \tag{5-4}$$

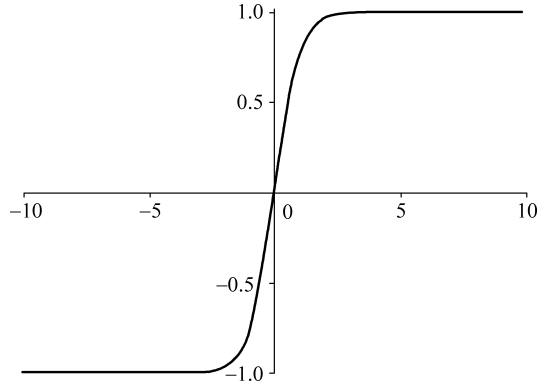


图 5-4　Tanh 函数

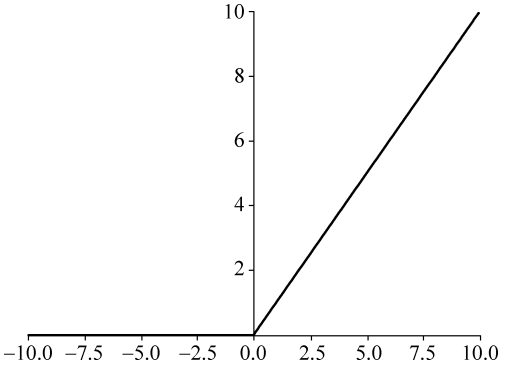


图 5-5　ReLU 函数

5.1.3　人工神经网络

人工神经元以层的形式组织，构成人工神经网络。虽然神经网络有许多不同的类型，但它们都遵循相似的基本原理。从原理上讲，神经网络可以被视为通用逼近器，即可以实现从一个向量空间到另一个向量空间的任意映射。此外，神经网络的另一个优势在于其能够捕获隐含在数据中的一些先验或未知信息，这些信息可能难以通过传统方法提取出来。这个过程被称为“神经网络学习”或“神经网络训练”。人工神经网络利用预先提供的输入输出数据，通过分析和研究两者之间存在的复杂联系和变化规律，最终通过挖掘出来的规律形成一个复杂的非线性函数。训练过程主要有两种类型：有监督和无监督。有监督的训练意味着神经网络知道

真实的输出标签，然后通过计算网络输出和真实标签的误差来调整权重系数。无监督训练则表明真实的输出是未知的，其本质上可以看作一个统计方法，是在缺乏标签的前提下找到数据中潜在特征的一种训练方式。这两种训练方式在不同场景下都具有重要的应用，可以根据任务的性质和数据的可用性来选择适当的训练方法。

神经网络至少由两层组成：一个输入层和一个输出层。输入层中的源节点主要负责接收和传递输入数据，不进行计算。而输出层是网络的最后一层，该层中的“输出单元”需要对前一层的输入进行计算，用于生成最终的输出。只包含单层计算单元的前向神经网络被称为单层感知器，如图 5-6 所示。单层感知器是模式分类神经网络中最简单的一种。当一个有限的样本集是线性可分时，它可以被单层感知器正确分类。

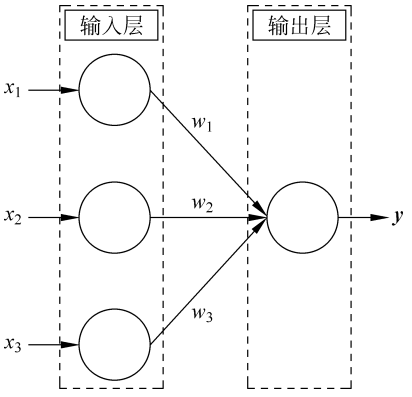


图 5-6　单层感知器

5.2　多层神经网络

多层神经网络也被称为多层感知机（Multilayer Perceptron，MLP）。由于单层感知机的表达能力有限，要实现更加复杂的函数拟合或特征提取，需要使用多层神经网络。多层神经网络由多个神经网络层组成，通常包括输入层、隐含层和输出层，每个层都包含多个神经元。位于输入层和输出层之间的层被称为隐含层，其计算节点也相应地被称为隐含神经元或隐含单元。通过添加一个或多个隐含层，神经网络可学习并提取更高阶的特征表示。神经网络是完全连接的，即网络的每一层的每个节点都连接到相邻的前向层的其他节点。在神经网络的设计中，输入层与输出层的节点数依据任务决定，因此往往是固定的。然而，隐含层的设计需要根据具体问题的复杂性进行调整，这是构建神经网络的关键之一。图 5-7 展示了一个多层感知机的模型架构，其包含以下 3 部分，分别是输入层、输出层和隐含层（2 层）。

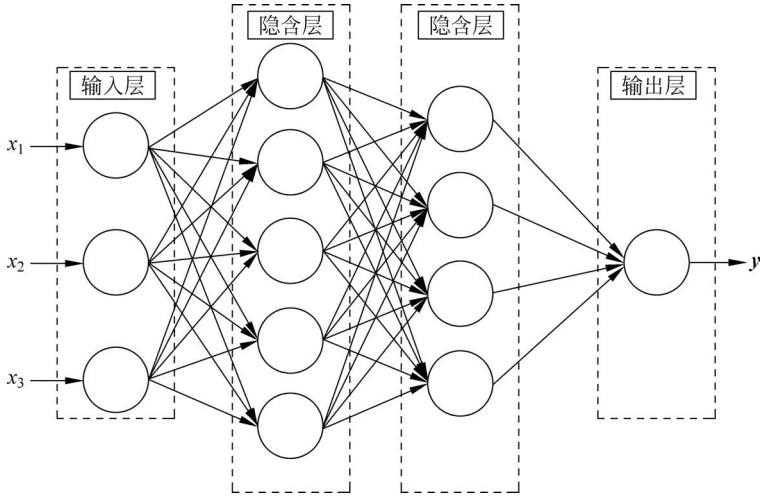


图 5-7　多层感知机的模型架构

（1）输入层：由输入数据的特征数量决定，通常情况下，每个输入特征对应一个神经元。以预测车辆行为为例，如果我们需要考虑车辆的速度、加速度、位置和方向这 4 个关键特征，那

么我们会设计一个包含 4 个神经单元的输入层。每个神经单元负责接收和处理其中一个特征的信息，这有助于神经网络有效地理解和利用输入数据的各方面。

（2）输出层：根据任务需要输出的结果种类而定。例如，如果需要判断输入图片是小猫还是小狗，这是一个二分类问题，因此输出层将包含两个神经单元，每个神经单元对应一个可能的类别（小猫或小狗）。

（3）隐含层：需要工程师精心设计和测试，以获得一个较好的模型。

5.2.1　前向传播

前向传播过程可以简单理解为信息从神经网络的输入层逐层向前传递，每一层都通过加权和激活操作对信息进行处理，然后将处理后的信息传递给下一层，直到输出层生成最终的模型预测结果。图 5-8 展示了一次前向传播过程，$\boldsymbol{x}$ 表示一个训练样本，$\boldsymbol{y}$ 表示期望的输出。$\boldsymbol{W}$ 和 $\boldsymbol{b}$ 分别代表每一层的权重矩阵和偏置向量。

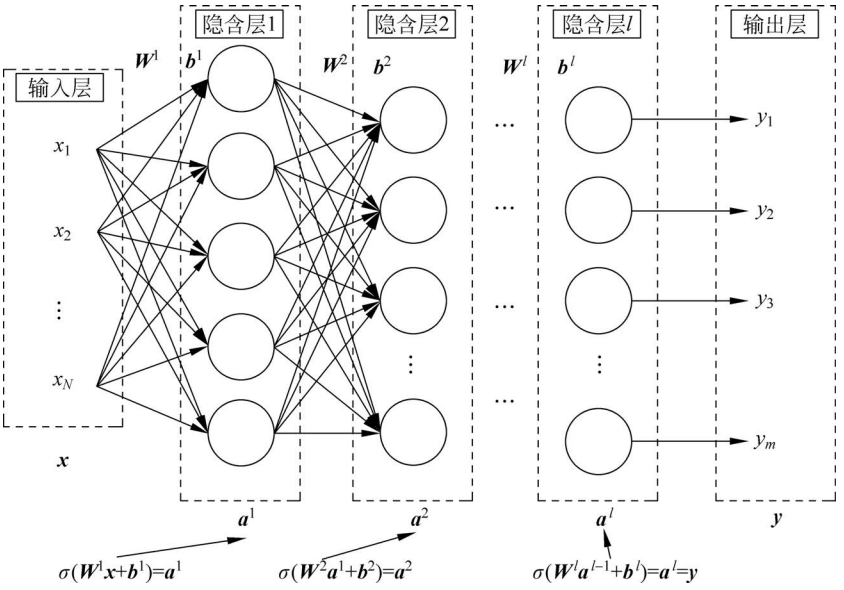


图 5-8　一次前向传播过程

5.2.2　反向传播算法

假设要构建一个图片分类系统，目标是对猫、狗和马进行分类。首先，需要收集大量关于这些动物的图片，并为每张图片标注正确的类别标签。在训练过程中，每张图片被输入模型中，模型会生成一个分数向量，其中每个类别都有一个相应的得分，我们的目标是使模型能够准确地预测最高得分对应的类别。但在没有经过训练的情况下，模型通常无法做到这一点。因此，引入一个目标函数，用来衡量模型的输出分数与期望分数之间的差距，然后通过调整内部可调参数（权重）来缩小这个差距。机器学习的核心任务就是通过反复的训练来找到合适的权值和偏置，使得系统的输出满足任务的需求。

当设计好神经网络的结构，且有训练样本时，在给定损失函数的情况下，最终的目标是通过优化神经网络中的参数权重 $\boldsymbol{W}$ 和偏置 $\boldsymbol{b}$，以使模型能够更准确地进行预测。为了实现这一目标，我们使用的主要算法是梯度下降法，也称作 BP（Back Propagation）算法，它是神经网络领域最成功及最常用的优化算法。本节主要介绍 BP 算法的推导过程。在开始推导之前，首

先回顾一下链式法则。

法则 1：

$$y=g(x),\quad z=h(y)$$

$$\Delta x\to\Delta y\to\Delta z$$

$$\frac{\mathrm{d}z}{\mathrm{d}x}=\frac{\mathrm{d}z}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}x}$$

法则 2：

$$x=g(s),\quad y=h(s),\quad z=k(x,y)$$

$$\Delta\frac{\partial z}{\partial s}=\frac{\partial z}{\partial x}\frac{\partial x}{\partial s}+\frac{\partial z}{\partial y}\frac{\partial y}{\partial s}$$

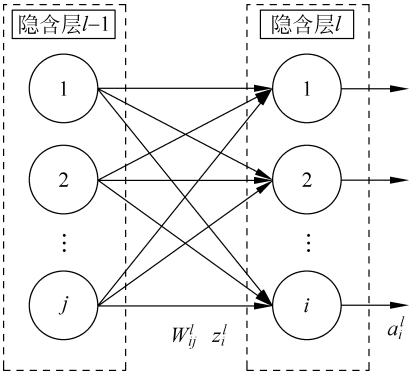


图 5-9 第 $l-1$ 层与第 l 层的神经网络

给定训练样本集合$\{(x^1,\hat{y}^1),\cdots,(x^r,\hat{y}^r),\cdots,(x^R,\hat{y}^R)\}$，假定损失函数为

$$C(\boldsymbol{\theta})=\frac{1}{R}\sum_r\|y^r-\hat{y}^r\|^2=\frac{1}{R}\sum_r C^r(\boldsymbol{\theta}) \tag{5-5}$$

$$\nabla C(\boldsymbol{\theta})=\frac{1}{R}\sum_r\nabla C^r(\boldsymbol{\theta}) \tag{5-6}$$

其中，$\boldsymbol{\theta}$ 为损失函数中的参数向量，包含了神经网络中的 $\boldsymbol{W}$ 和 $\boldsymbol{b}$，这里的目标就是求出$\partial C^r/\partial W_{ij}^l$ 和$\partial C^r/\partial b_i^l$。图 5-9 所示为第 $l-1$ 层与第 l 层的神经网络。

根据链式法则，$\Delta W_{ij}^l\to\Delta z_i^l\cdots\to\Delta C^r$，$\partial C^r/\partial W_{ij}^l$ 其导致由两部分相乘所得：

$$\frac{\partial C^r}{\partial W_{ij}^l}=\frac{\partial z_i^l}{dW_{ij}^l}\frac{\partial C^r}{\partial z_i^l} \tag{5-7}$$

第一项$\partial z_i^l/\partial W_{ij}^l$ 的计算过程如下。

当 $l>1$，即这里的观察对象是神经网络中间的两个隐含层时

$$z_i^l=\sum_j W_{ij}^l a_j^{l-1}+b_i^l\qquad\frac{\partial z_i^l}{\partial W_{ij}^l}=a_j^{l-1} \tag{5-8}$$

当 $l=1$，即神经网络只包含一个隐含层时

$$z_i^l=\sum_j W_{ij}^l x_j^r+b_i^l\qquad\frac{\partial z_i^l}{\partial W_{ij}^l}=x_j^r \tag{5-9}$$

归纳可得

$$\frac{\partial z_i^l}{\partial W_{ij}^l}=\begin{cases}x_j^r, & l=1\\ a_j^{l-1}, & l>1\end{cases} \tag{5-10}$$

第二项可以定义为 $\delta_i^l=\dfrac{\partial C^r}{\partial z_i^l}$，当 $l<L$ 时，观察的对象是中间的隐含层，根据链式法则，δ_i^l 与 δ_i^{l+1} 的关系如下：

$$\delta_i^l=\frac{\partial C^r}{\partial z_i^l}=\frac{\partial a_i^l}{\partial z_i^l}\frac{\partial C^r}{\partial a_i^l}=\frac{\partial a_i^l}{\partial z_i^l}\sum_k\frac{\partial z_k^{l+1}}{\partial a_i^l}\frac{\partial C^r}{\partial z_k^{l+1}} \tag{5-11}$$

其中的每一项可以表示为

$$\sigma'(z_i^l)=\frac{\partial a_i^l}{\partial z_i^l}W_{ki}^{l+1}=\frac{\partial z_k^{l+1}}{\partial a_i^l}\delta_k^{l+1}=\frac{\partial C^r}{\partial z_k^{l+1}} \tag{5-12}$$

可得

$$\delta_i^l = \sigma'(z_i^l)\sum_k W_{ki}^{l+1}\delta_k^{l+1} \tag{5-13}$$

当 $l=L$ 时，观察的对象是输出层：

$$\delta_i^L = \frac{\partial C^r}{\partial z_i^L} = \frac{\partial y_i^r}{\partial z_i^L}\frac{\partial C^r}{\partial y_i^r} = \sigma'(z_i^L)\frac{\partial C^r}{\partial y_i^r} \tag{5-14}$$

根据 $C^r = \| y^r - \hat{y}^r \|$，有

$$\delta^l = \begin{cases} (\boldsymbol{W}^{l+1})^{\mathrm{T}}\delta^{l+1}\odot\sigma'(z^l), & l < L \\ \nabla_{y^r}C^r\odot\sigma'(z^l), & l = L \end{cases} \tag{5-15}$$

此外，还需要计算$\partial C^r/\partial b_i^l$，其推导过程如下：

$$\frac{\partial C^r}{\partial b_i^l} = \frac{\partial z_i^l}{db_i^l}\frac{\partial C^r}{\partial z_i^l} = \delta_i^l \tag{5-16}$$

因此，整个反向传播的过程本质就是先第一层正向计算$\partial z_i^l/\partial W_{ij}^l$，再从最后一层反向计算$\partial C^r/\partial z_i^l$，最后求出$\partial C^r/\partial W_{ij}^l$ 和$\partial C^r/\partial b_i^l$。

综上所述就是 BP 算法的流程，误差反向传播算法使用链式求导法则将输出层的误差反向传回网络，然后根据误差信息来调整权重参数，使得神经网络的权值具有较简单的梯度计算法。此过程从输出层开始，计算该层中的各个神经元权值的梯度，以确定它们对误差的贡献。然后基于上一层的梯度值，计算当前层参数的梯度值，并不断重复此过程，直到梯度信息传播至网络的第一层。

5.2.3　神经网络之特征提取 Word2Vec

在介绍 Word2Vec 之前，需要先解释一下词嵌入(word embedding)。词嵌入是一种文本表示方法，它将文中的词语转换为可计算、结构化的向量表示。由于文本是一种非结构化数据，因此不能直接计算和分析。词嵌入的作用就是将这些词语转换为向量，以便于在计算机上进行各种自然语言处理任务，如文本分类、情感分析等。下面介绍 3 种常见的文本表示方法。

1. one-hot 编码

假如要计算的文本中包含 4 个词：我、爱、大、家。可以将每个词表示为向量中的一个位置。因此，用 one-hot 编码来表示就会得到一个向量：

我：(1,0,0,0)　爱：(0,1,0,0)

大：(0,0,1,0)　家：(0,0,0,1)

每个词都被表示为一个唯一的向量，其中只有一个元素为 1，其余元素为 0，这种表示方法非常直观和易于理解。然而，在处理大型词汇表时，one-hot 编码会生成非常高维的稀疏向量。这不仅浪费了存储空间，还增加了计算的复杂性。此外，这种编码方式无法捕获词语之间的语义关系，因为每个词语都被视为彼此独立的。

2. 整数编码

这种编码方式也非常好理解，它用一种数字来代表一个词，继续使用上面的例子，则这 4 个词分别被编码为：

我：1　爱：2　大：3　家：4

将句子里的每个词拼起来就是可以表示一句话的向量。这种编码方式相对于 one-hot 编

码来说可以显著降低维度，但仍无法捕获词语之间的语义关系。

3. 词嵌入

词嵌入也是文本表示的一类方法，它并不特指某个具体的算法，而是指将文本中的词语映射为连续的低维向量的通用方法。相较于前面提到的两种方式，它有几个明显的优势：可以使用低维向量来表示文本，与 one-hot 编码相比，这不仅节省了存储空间，还降低了计算复杂性；在词嵌入的向量空间上，语义相似的词会比较接近；通用性强，适用于各种自然语言处理任务。

Word2Vec 是一种基于统计方法来获得词向量的词嵌入方法，由谷歌的 Mikolov 于 2013 年首次提出。虽然在 2018 年之前，Word2Vec 方式比较主流，但随着 BERT、GPT 等模型的出现，Word2Vec 不再是效果最好的方法。简而言之，Word2Vec 是一种将稀疏的 one-hot 形式的词向量通过一个一层的神经网络映射为一个 n（一般为几百）维的稠密向量的过程。Word2Vec 包括两个重要的模型：CBOW(Continuous Bag-of-Word)模型与 Skip-gram 模型，这两个模型分别如图 5-10 和图 5-11 所示。

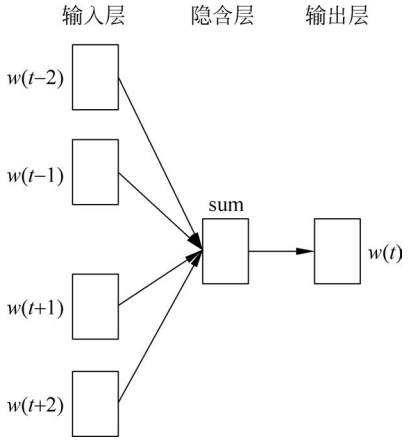


图 5-10　CBOW 模型

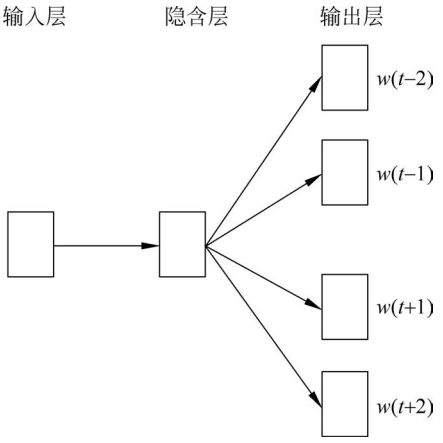


图 5-11　Skip-gram 模型

CBOW 模型通过上下文来预测当前词汇，试图从一句话中删除一个单词，然后预测被删除的单词是什么。与之不同，Skip-gram 模型则以当前单词来预测周围可能出现的上下文单词，即猜测前面和后面可能会出现哪些单词。CBOW 模型的训练过程如图 5-12 所示。

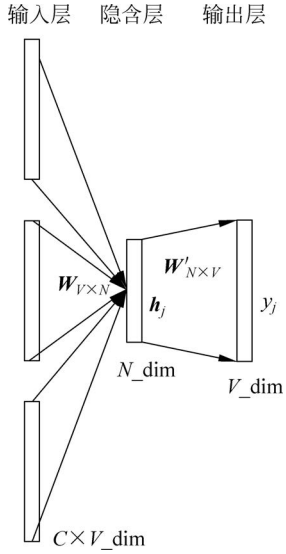


图 5-12　CBOW 模型的训练过程

(1) 输入层是上下文单词的 one-hot。假设单词向量空间的维度为 V，即整个词库的词典大小为 V，上下文单词窗口的大小为 C。

(2) 假设最终词向量的维度大小为 N，则图中的权值共享矩阵为 $\boldsymbol{W}$。$\boldsymbol{W}$ 的大小为 $V\times N$，并且初始化。

(3) 假设语料中有一句话“我爱大家”。如果现在关注“爱”这个词，令 $C=3$，则其上下文为“我”“大”“家”。模型把“我”“大”“家”的 one-hot 形式作为输入。易知其大小为 $1\times V$。C 个大小为 $1\times V$ 的向量分别跟同一个大小为 $V\times N$ 的权值共享矩阵 $\boldsymbol{W}$ 相乘，得到的是 C 个大小为 $1\times N$ 的隐含层。

(4) 将 C 个大小为 $1\times N$ 的隐含层取平均，得到一个大小为 $1\times N$ 的向量，即图 5-12 中的 $\boldsymbol{h}_i$。

(5) 输出权重矩阵 $\boldsymbol{W}'$的大小为 $N\times V$，并进行相应的初始化。

(6) 将得到的隐含层向量与输出权重矩阵相乘，并使用激活函数计算得到大小为 $1\times V$ 的向量。此向量的每一维代表语料库中的一个单词。概率中最大的索引所代表的单词为预测出的中间词。

(7) 将预测结果与真实值中的 one-hot 编码进行比较，求损失函数的极小值。

Word2Vec 的实现方法可以总结为：首先基于训练数据构建一个神经网络。当这个网络训练完成后，不直接将其用于处理新任务，而是需要使用该模型通过训练数据所学得的参数，如隐含层中的权重矩阵 $\boldsymbol{W}$。这些权重实际上代表了我们试图学习的词的特征表示。

5.3　卷积神经网络

近年来，随着人工神经网络的兴起，机器学习领域发生了很大的变化。这些受生物启发的计算模型在常规机器学习任务中的性能明显超越了以往各种形式的人工智能。其中最引人瞩目的人工神经网络架构形式之一是卷积神经网络(Convolutional Neural Networks，CNN)。最初，卷积神经网络主要用于计算机图像处理，但随着人们的不断探索和创新，它也被广泛应用于视频数据分析、自然语言处理、药物发现等领域。卷积神经网络的运作方式与标准的神经网络非常相似，由通过学习自我优化的神经元组成。每个神经元仍然会接收一个输入并执行一个操作(如一个标量积和一个非线性函数)——这是无数人工神经网络的基础。从输入的原始图像向量到最终的类别分数输出，整个网络仍然将表示一个单一的感知分数函数(权重)。最后一层包含与类别相关联的损失函数，传统人工神经网络开发的所有通用技巧和窍门仍然适用。卷积神经网络和传统的人工神经网络之间唯一的显著区别是，卷积神经网络主要用于图像内的模式识别领域。这允许将特定于图像的特性编码到架构中，使网络更适合于以图像为核心的任务，并进一步减少模型构建所需的参数。

卷积神经网络概念的产生可以追溯到 20 世纪 60 年代初期，当时 Hubel 和 Wiesel 通过对猫的大脑视觉皮层进行研究，首次提出了“感受野”这个新概念。感受野指的是卷积神经网络在每一层输出的特征图(feature map)上的像素点在输入图片上的映射区域。更通俗的解释是，特征图上的一个点对应输入图上的一个区域。1989 年，LeCun 将反向传播算法与权值共享结合，提出了卷积神经网络，并首次成功地将其应用到美国邮局的手写字符识别系统中。1998 年，LeCun 又提出了卷积神经网络的经典网络模型 LeNet-5，从而进一步提高手写字符识别的准确度。

通常，人们对外界的感知是从部分到整体的过程。在图像中，像素点之间的位置联系是局部的，即空间位置较远的像素点之间的相关性较弱。而卷积神经网络的每个神经元只需对局部图像进行感知，然后在更高层将这些局部的信息综合起来，从而获取全局信息。

卷积神经网络架构可用于手写数据集的分类，如图 5-13 所示。从这个示例可以发现卷积神经网络的基本功能可以分解为 4 个关键领域：输入层、卷积层(convolution layer)、池化层(pooling layer)及全连接层(fully-connected layer)。与其他形式的人工神经网络类似，输入层提供图像的原始像素信息。卷积层和池化层通常会设置多个，并采用交替方式排列，即一个卷积层连接一个池化层，再接一个卷积层，以此类推。通过这种简单的层次结构，卷积神经网络可以使用卷积和降采样技术对原始输入进行逐层转换，从而进行特征提取，并最终通过连接全

连接层进行分类处理。卷积神经网络通过卷积操作和池化操作学习输入特征的局部模式。随着网络层数的叠加，卷积神经网络将不断地对这些局部信息进行组合和抽象，最终可以学习到更高级的特征。

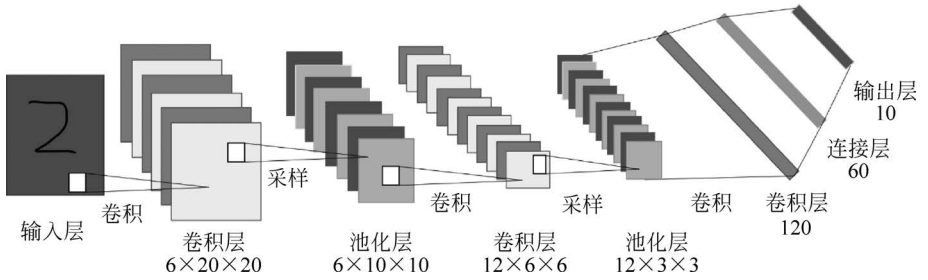


图 5-13　LeNet-5 卷积神经网络

接下来将详细描述每个层，并结合实际的例子来理解卷积神经网络是如何利用各个层在图片中进行特征提取的。

5.3.1　卷积层

卷积层在卷积神经网络的运作方式中起着至关重要的作用。该层的参数主要是可学习的卷积核。当数据到达卷积层时，卷积层将计算神经元的权重与连接到输入区域的标量的乘积，以确定连接到输入的局部区域的神经元的输出。当输入数据为图片时，实际上输入神经网络的并不是彩色图片，而是一系列数字。如图 5-14 所示，图像上有一个灰色方块组成的 X 形，其中灰色方块表示值为 1 的像素点，白色方块表示值为 0 的像素点。当神经网络要处理这么多数据信息时，卷积神经网络就能够充分发挥其优势。如果要识别出这个 X，只需要通过卷积神经网络识别出左下和右下的斜线即可。

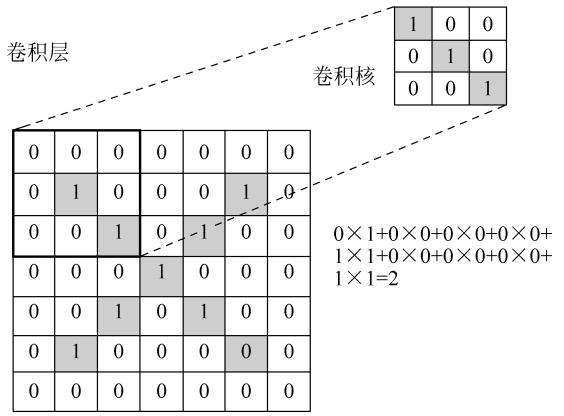


图 5-14　卷积核

卷积计算的结果如图 5-15 所示，可以看到图中数值越大，表示越符合卷积和右下斜线的特征，因此数值较大的区域基本上对应着原图上有斜线的部分。同理，如果想要计算左下斜线的部分，可以用左下斜线的卷积核进行相应的计算。

传统的神经网络和机器学习方法通常需要对图像进行复杂的预处理，以便提取特征，然后将得到的特征输入神经网络中。通过引入卷积操作，我们能够利用图片空间上的局部相关性，从而自动提取特征。一般情况下，卷积层包含多个卷积核，对应多个通道。这是由于权值共享，每个卷积核只负责捕获一种特征。如果想要提高卷积神经网络的表达能力，就需要设置更

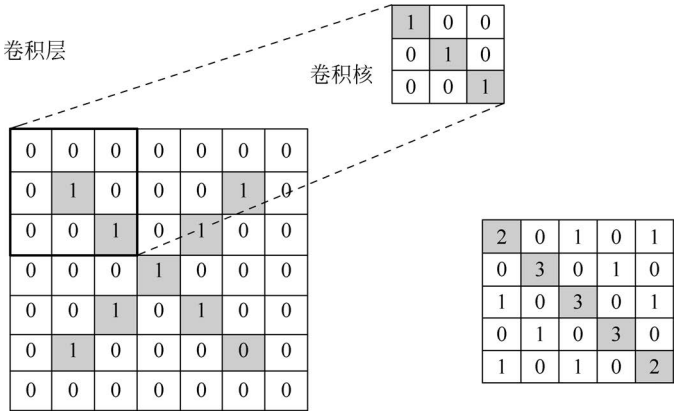


图 5-15　卷积计算的结果

多的卷积核。如图 5-16 所示，卷积操作有以下 3 个重要的参数。

（1）卷积核尺寸：感受野的大小，一般指卷积核的长和宽，如 3×3 的卷积核。

（2）卷积核步长：卷积核在核宽度方向上每次移动的距离。例如，步长为 1 表示每次移动 1 格，步长为 3 则表示每次移动 3 格。

（3）卷积核的数量：对应卷积核输出特征的深度。每个卷积核的输出为一个通道，多个卷积核堆叠就会形成一个特征立方体。例如，如果有 4 个不同的卷积核，那么就会形成由 4 个特征平面组成的立方体。

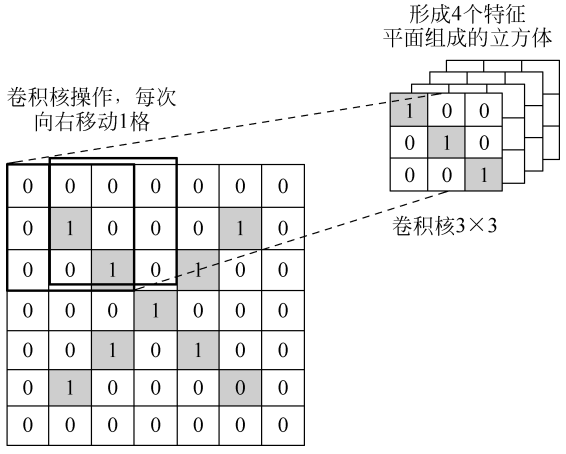


图 5-16　卷积核的参数

5.3.2　池化层

接下来进入池化层。实际上，一张图的像素非常多，因此计算它们的信息需要大量时间。为了减少计算负担，需要对图片进行压缩，而这正是池化层的作用。图像具有一种“静态性”的属性，即某个图像区域有用的特征极有可能在其他区域也适用。池化层的任务是根据不同位置的特征进行聚合和统计。其目的是逐步降低表示的维度，从而进一步降低参数数量和模型的计算复杂度。常见的池化方法包括以下 3 种。

（1）最大池化（max pooling）：通过选取图像区域上某个特征的最大值来代表这个图像区域的特征。

（2）最小池化（min pooling）：通过选取图像区域上某个特征的最小值来代表这个图像区

域的特征。

(3) 平均池化(average pooling)：通过计算图像区域上某个特征的平均值来代表这个图像区域的特征。

如图 5-17 所示，这里采用了最大池化方法。该过程非常简单，只需从 4 个像素点中保留最大值。通过多次迭代计算，最终可以得到一个更小的图像。可以看到，压缩后的图像仍然保留了原有的特征。

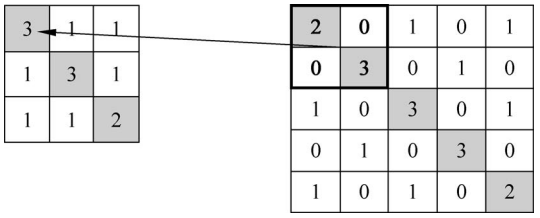


图 5-17　池化操作

5.3.3　全连接层

池化层得到的结果虽然已经提取了特征，计算机仍然无法直接识别这些特征。这时需要使用全连接层。如图 5-18 所示，将两个 3×3 的图像展开并拼接成一维数组。这个数组是从图片中提取的特征表示。在识别图片的类别之前，计算机会通过训练样本进行训练。完成训练后，计算机会为每个特定的图像保存一个特征向量。然后，计算机将用于识别图像的特征向量与训练后得到的特征向量进行比较，根据相似度进行识别。实际上，全连接层可以看作一个大小为 1×1 的卷积核的卷积层。全连接层中的每个单元都与前一层的所有单元紧密相连。在典型的卷积神经网络体系结构中，全连接层通常位于末端。如图 5-18 所示，经过训练的卷积神经网络可以成功识别字母 X。

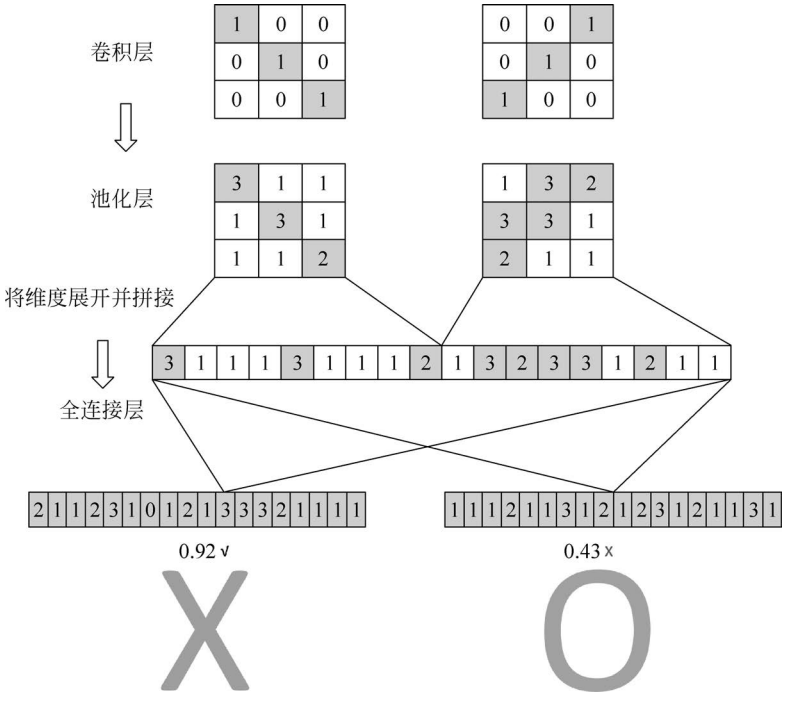
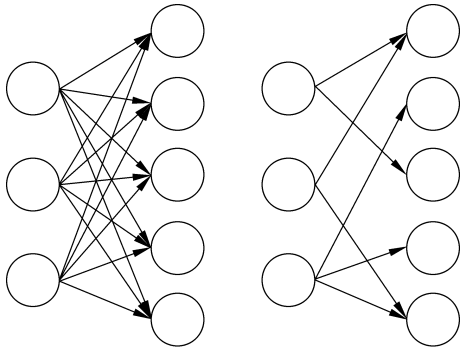


图 5-18　将图片展开并拼接

5.3.4　卷积神经网络的特点

卷积神经网络演变自多层感知机，具有局部连接、权值共享和降采样等特性，在图像处理方面表现出色。相比其他神经网络，卷积神经网络的特殊之处在于权值共享和局部连接。局部连接使得网络能够获取图像的局部特征，而权值共享则降低了网络的训练复杂度。此外，池化操作实现了数据的降维，使得低层次的局部特征能够组合成为更高层次的特征，从而获得整个图像的特征表示。

在传统的神经网络结构中，每个神经元都与上一层的所有神经元全连接。然而，在卷积神经网络中，上一层的神经元仅连接到下一层的部分神经元，如图 5-19 所示。



(a) 全连接　　(b) 局部连接

图 5-19　全连接与局部连接

在卷积神经网络中，图像像素之间存在局部相关性，每个像素点都有着具体的实际意义。为了减少模型参数并提取局部特征，采用稀疏的局部连接方式。

卷积核像一个滑动窗口，在整个输入图像中以特定步长滑动，通过卷积运算生成输入图像的特征图。这个特征图包含了卷积层所提取的局部特征，而卷积核则共享参数。在整个网络的训练过程中，卷积核中的权值会随着训练的进行而更新。因此，整张图像都使用同一个卷积核内的参数，实现了权值共享。

5.4　循环神经网络

5.4.1　序列数据

人工神经网络和卷积神经网络都属于前馈神经网络。前馈神经网络是一种静态网络，数据传递是单向的。这意味着网络的输出只取决于当前的输入，不具备记忆能力。在实际任务中，许多数据都具有上下文关联性，这些数据被称为序列数据，如文本、语音和视频等。这些数据长度不固定，而前馈神经网络的输入和输出是固定长度的，因此难以处理序列数据。在序列数据中，各元素之间存在一定的关联，因此，在处理序列数据时，需要考虑之前时间步的数据，而不仅仅是当前时间步的数据。

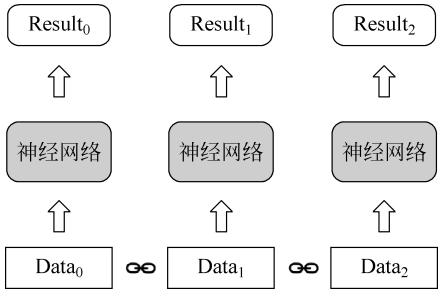


图 5-20　序列数据

现给定一组序列数据 $Data_{0,1,2}$，如图 5-20 所示。根据 $Data_0$ 可以预测 $Result_0$，当预测其他数据时，仍

然只使用单个数据，并且每次使用的神经网络都是相同的。虽然这些数据具有顺序关系，但传统的神经网络结构无法学习到这些数据之间的相关性。

那么，如何使神经网络能够分析数据之间的相关性呢？可以想象一下人类是如何分析不同事物之间的关联的。最基本的方式就是记住之前发生的事件。如果让神经网络也具备这种记忆功能，它在分析数据 Data_0 后，就可以将结果存储在记忆中。在分析 Data_1 时，神经网络会产生新的记忆，这些新的记忆与旧的记忆有关？因此，可以调用旧的记忆来一起分析数据。然而，要分析更多有序数据，就需要用到循环神经网络（Recurrent Neural Networks，RNN），它能够高效地处理序列数据，并记录各个时序的输出。由于先前时间步的输出会影响后续的输出，因此循环神经网络可以挖掘序列数据中的时序信息和语义信息。

5.4.2 循环神经网络

循环神经网络常用于语音识别、机器翻译、视频解析等领域，它模拟了人脑记忆功能。该网络利用周期性的隐含层节点连接来捕捉序列数据中的动态信息。在循环神经网络中，一个序列的当前输出与之前的输出相关联。具体而言，网络会对之前的数据加以记忆并将其应用于当前输出的计算中。这是通过隐含层之间的相互连接实现的，因此隐含层的输入不仅包括输入层的数据，还包括上一时刻隐含层的输出。将循环神经网络展开成一个全连接的神经网络，如图 5-21 所示。可以看到隐含层中的神经元不仅与上下层的神经元互相连接，还与同层的其他节点互相连接。

理论上，循环神经网络可以处理任意长度的序列数据。然而，在实际应用中，为了降低复杂度，通常会假设当前的状态只与前面的几个状态有关。循环神经网络由多个循环体堆叠而成，为了方便堆叠，循环体有两类输出：隐含层输出和最终输出。循环体及其按时间展开后的效果如图 5-22 所示。

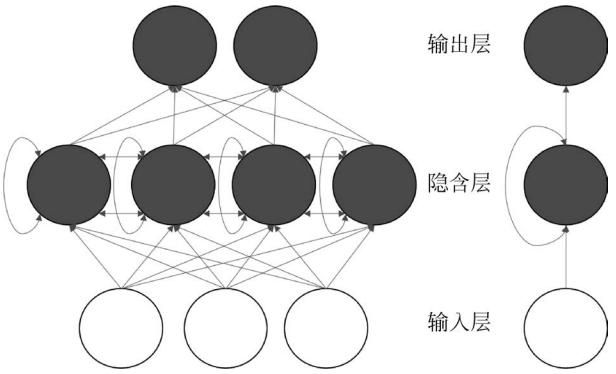


图 5-21 将循环神经网络展开成全神经网络

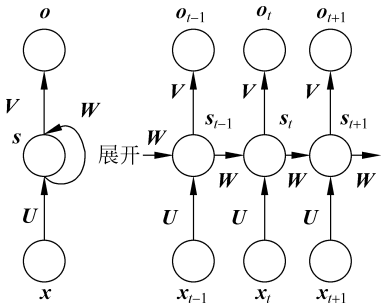


图 5-22 循环体及其按时间展开后的效果

在同一个隐含单元中，在 $t-1$ 时刻，接收了 $\boldsymbol{x}_{t-1}$ 的输入；在 t 时刻，接收了 $\boldsymbol{x}_t$ 的输入。在 $t+1$ 时刻，接收了 $\boldsymbol{x}_{t+1}$ 的输入。也就是说，同一个隐含单元中在不同时刻会接收到不同的输入。

$\boldsymbol{s}_t$ 的输入包括 $\boldsymbol{x}_t$ 和 $\boldsymbol{s}_{t-1}$，$\boldsymbol{x}_t$ 是当前时刻的输入，$\boldsymbol{s}_{t-1}$ 是上一时刻的信息。其中，$\boldsymbol{s}_t$ 是序列在时间 t 处的记忆单元，缓存了之前的信息。f 通常是非线性的激活函数，如 Tanh 函数或 ReLU 函数，$\boldsymbol{s}_t$ 的更新公式如下所示：

$$\boldsymbol{s}_t = f(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{s}_{t-1}) \tag{5-17}$$

$\boldsymbol{o}_t$ 是序列在时间 t 处的输出，softmax 是激活函数，当然也可以使用其他的激活函数，其

更新公式如下：

$$\boldsymbol{o}_t = \text{softmax}(\boldsymbol{V}\boldsymbol{s}_t) \tag{5-18}$$

在传统神经网络中，每个网络层的参数都是独立的。然而，在循环神经网络中，每一步的输入都共享相同的 $\boldsymbol{U}$、$\boldsymbol{V}$ 和 $\boldsymbol{W}$ 参数集。换句话说，循环神经网络中的每一步所执行的任务都是相同的，只是输入数据不同，从而极大地减少了需要学习的参数数量。当展开循环神经网络时，它变成一个多层的网络。在一个多层传统神经网络中，连接 $\boldsymbol{x}_t$ 到 $\boldsymbol{s}_t$ 之间的 $\boldsymbol{U}$ 矩阵与连接 $\boldsymbol{x}_{t+1}$ 到 $\boldsymbol{s}_{t+1}$ 之间的 $\boldsymbol{U}$ 矩阵是不同的；然而，对于循环神经网络，这些 $\boldsymbol{U}$ 矩阵是相同的。同样地，连接 $\boldsymbol{s}_{t-1}$ 与 $\boldsymbol{s}_t$ 之间的 $\boldsymbol{W}$、连接 $\boldsymbol{s}_t$ 与 $\boldsymbol{o}_t$ 之间的 $\boldsymbol{V}$ 也是相同的。尽管图 5-22 中每个时间步都有输出，但并非每个时间步的输出都是必需的。例如，在需要预测一条语句表达的情感时，仅需要最后一个单词的输出，而不需要每个单词的输出。循环神经网络的关键在于隐含层，该层可以捕获序列信息。

5.4.3　循环神经网络的变体

在训练中，原始的循环神经网络随着训练时间的增加和网络层数的增多，容易出现梯度爆炸或梯度消失的问题，导致无法处理长序列数据和捕获长距离依赖关系的问题。为解决这一问题，提出了改进方案，即长短期记忆神经网络（Long Short-term Memory，LSTM）。本节将详细介绍 LSTM 的网络架构。

LSTM 是目前最知名、最成功的循环神经网络改进之一。它具备对重要的信息进行长期记忆的能力，一定程度上缓解了梯度消失的问题。与传统的循环神经网络相比，LSTM 在 $\boldsymbol{s}_{t-1}$、$\boldsymbol{x}_t$ 的基础上加了一个长时记忆状态 $\boldsymbol{c}_{t-1}$（cell state）来计算 $\boldsymbol{s}_t$，同时对网络模型内部进行了精心设计，增加了遗忘门 $\boldsymbol{f}_t$、输入门 $\boldsymbol{i}_t$、输出门 $\boldsymbol{o}_t$ 三个门控单元以及一个内部记忆神经元 $\tilde{\boldsymbol{c}}_t$。遗忘门 $\boldsymbol{f}_t$ 的作用是控制前一步记忆单元中信息被遗忘的程度，输入门 $\boldsymbol{i}_t$ 则控制当前记忆中的信息更新到记忆单元的程度，而输出门 $\boldsymbol{o}_t$ 则决定了当前的隐含状态的输出。在训练好的网络中，当输入序列不包含重要信息时，LSTM 遗忘门的值接近于 1，输入门的值接近于 0，这有助于保留过去的信息，实现了长时记忆的功能。然而，当输入序列中出现了重要信息，且该信息意味着之前的记忆不再重要时，输入门的值会接近于 1，而遗忘门的值会接近于 0，从而实现了旧记忆的遗忘，同时新的重要信息被纳入记忆。通过这样的设置，整个网络更容易学习到序列之间的长期依赖关系。

图 5-23 展示了 LSTM 的网络架构。

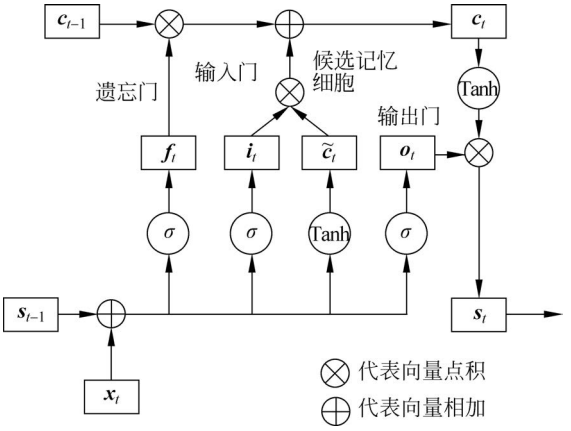


图 5-23　LSTM 的网络架构

经典的 LSTM 中，第 t 步的更新计算公式如下所示：

$$\boldsymbol{x}_{\text{input}} = \text{concat}(\boldsymbol{s}_{t-1}, \boldsymbol{x}_t) \tag{5-19}$$

遗忘门神经元：

$$\boldsymbol{f}_t = \sigma(\boldsymbol{x}_{\text{input}} \cdot \boldsymbol{W}_f + \boldsymbol{b}_f) \tag{5-20}$$

输入门神经元：

$$\boldsymbol{i}_t = \sigma(\boldsymbol{x}_{\text{input}} \cdot \boldsymbol{W}_i + \boldsymbol{b}_i) \tag{5-21}$$

记忆门神经元：

$$\tilde{\boldsymbol{c}}_t = \text{Tanh}(\boldsymbol{x}_{\text{input}} \cdot \boldsymbol{W}_c + \boldsymbol{b}_c) \tag{5-22}$$

遗忘后的长时记忆：

$$\tilde{\boldsymbol{c}}'_{t-1} = \boldsymbol{f}_t \cdot \boldsymbol{c}_{t-1} \tag{5-23}$$

输入后的记忆：

$$\tilde{\boldsymbol{c}}'_t = \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{5-24}$$

输出门神经元：

$$\boldsymbol{o}_t = \sigma(\boldsymbol{x}_{\text{input}} \boldsymbol{W}_o + \boldsymbol{b}_o) \tag{5-25}$$

t 时刻的长时记忆：

$$\boldsymbol{c}_t = \tilde{\boldsymbol{c}}'_{t-1} + \tilde{\boldsymbol{c}}'_t \tag{5-26}$$

t 时刻的短时记忆：

$$\boldsymbol{s}_t = \boldsymbol{o}_t \odot \text{Tanh}(\boldsymbol{c}_t) \tag{5-27}$$

其中，$\boldsymbol{x}_{\text{input}}$ 指的是对上一时刻 $t-1$ 的记忆状态 $\boldsymbol{s}_{t-1}$ 以及当前时刻 t 的向量输入 $\boldsymbol{x}_t$ 进行特征维度的拼接所得到的结果。σ 指的是 Sigmoid 函数。$\boldsymbol{W}_f$、$\boldsymbol{b}_f$、$\boldsymbol{W}_i$、$\boldsymbol{b}_i$、$\boldsymbol{W}_o$、$\boldsymbol{b}_o$ 是各个门神经元的可学习参数。遗忘门、输入门以及输出门均采用 Sigmoid 作为激活函数，因此输出向量 $\boldsymbol{f}_t$、$\boldsymbol{i}_t$、$\boldsymbol{o}_t$ 的每个元素均为 0～1，用于调节各维度信息流通过门的数量；而记忆门与遗忘门、输入门和输出门神经元的输出向量具有相同的维度。不同的是，记忆门使用的激活函数是 Tanh，因此其输出向量 $\tilde{\boldsymbol{c}}_t$ 的每个元素均为－1～1。

5.4.4 双向 LSTM 之特征提取 ELMo

ELMo 于 2018 年 3 月提出，源自论文 *Deep contextualized word representations*。作者认为好的词表征模型应该能够同时兼顾两个问题：一是处理词语用法在语义和语法上的复杂用法；二是根据不同语境灵活调整词语的表征。传统的 Word2Vec 或者 Glove 只能解决第一个问题，它们生成的词向量是静态的，也就是说每个词的向量化表示是固定的。然而，很多单词在不同的语境下有不同的含义。例如，“我去洗手间方便一下”和“你今晚几点方便”这两句话中的“方便”表达的意思显然不同。因此，在这种情况下，需要一种动态的词向量能够根据语境来表示单词，ELMo 所做的就是这件事。该模型会根据上下文来推断每个词对应的词向量，并能够根据不同语境来理解多义词的含义。值得一提的是，ELMo 也是预训练语言模型的先河。

ELMo 的网络结构采用了双层双向 LSTM，该语言模型的训练任务目标是根据单词 w 的上下文来正确预测单词 w，其中 w 之前的单词序列称为上文，之后的单词序列称为下文。对于前向的 LSTM 中的第 t 个时刻而言，由于其模拟的是语言模型，所以第 t 个时刻的输出为

$$P(w_n \mid w_1, w_2, \cdots, w_{n-1}) \tag{5-28}$$

对于整个序列的输出有

$$P(h_1, h_2, \cdots, h_n) = \prod_{k=1}^{n} P(h_k \mid h_1, h_2, \cdots, h_{k-1}) \tag{5-29}$$

同理，对于后向的 LSTM 而言，其计算结果为

$$P(h_1,h_2,\cdots,h_n)=\prod_{k=1}^{n}P(h_k \mid h_{k+1},h_{k+2},\cdots,h_n) \tag{5-30}$$

综上所述，对于第一层的 BI-LSTM，可以综合前向和后向的两个 LSTM 得到输出的似然函数为

$$\sum_{i=1}^{n}\log(P(h_i))=\log(P(h_i \mid h_1,h_2,\cdots,h_{i-1});\theta_x,\theta_{\text{left}},\theta_s)+$$
$$\log(P(h_i \mid h_{i+1},h_{i+2},\cdots,h_n);\theta_x,\theta_{\text{right}},\theta_s) \tag{5-31}$$

其中，三个 θ 分别表示输入的参数、不同方向的 LSTM 的参数和 Softmax 的参数。图 5-24 展示的是 ELMo 的模型架构。

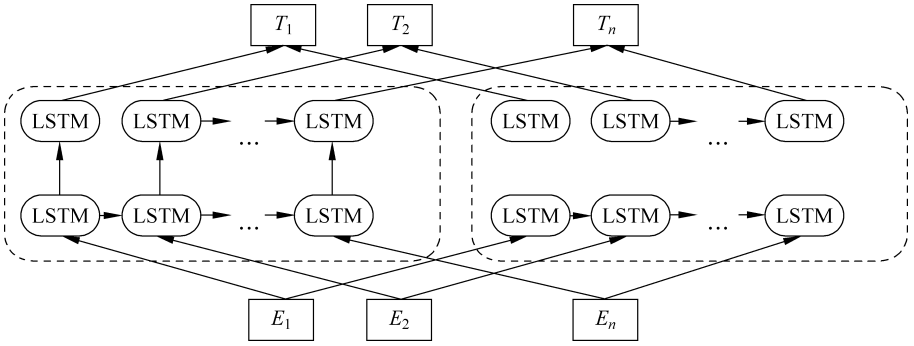


图 5-24　ELMo 预训练语言模型

图 5-24 左侧的前向双层 LSTM 代表正向编码器，它接收从左到右顺序排列的上下文文本(除了要预测的单词 w)context-before；右侧的逆向双层 LSTM 代表反方向编码器，接收从右到左逆序排列的下文文本 context-after。每个编码器都由两层 LSTM 堆叠而成。该网络结构在自然语言处理领域被广泛应用。通过使用大量的语料库训练该网络结构进行语言模型任务，可以事先预训练出模型。如果成功训练了该模型，在输入新的句子 X 时，每个单词都可以得到三个对应的嵌入向量；最底层是单词初始化的嵌入向量；第一层是双向 LSTM 中对应单词位置的嵌入向量，这一层更多地编码了单词的句法信息；第二层是 LSTM 中对应单词位置的嵌入向量，这一层更多地编码了单词的语义信息。因此，正向编码器和反向编码器都会获得相应单词的嵌入向量。接下来，为这三个嵌入向量中的每一个分配一个权重 a(可以通过学习获得)，并根据各自的权重进行加权求和以将它们整合成一个向量。然后，将整合后的向量作为输入 X 句子中对应单词的特征，用于下游任务。换句话说，ELMo 的预训练过程不仅学习了单词的嵌入向量，还学习了一个双层双向 LSTM 网络结构，这两者都非常有用。图的形式存在如图 5-25 所示。

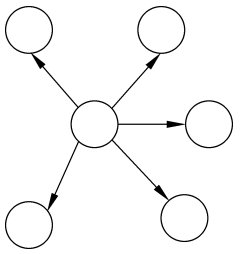


图 5-25　有向图

5.5　图神经网络

随着机器学习、深度学习的快速发展，语音、图像、自然语言处理等领域取得了巨大的突破。然而，这些领域中的数据通常具有非常简单的结构，例如序列或网格数据，而深度学习在处理这种类型的数据方面表现出色。然而，在实际工作中，并不是所有事物都能以序列或者网格形式来表示，例如，社交网络、知识图谱和生物网络等往往以图的形式存在，其中元素之间的

关系非常复杂。因此，许多学习任务需要有效地处理这种丰富的图数据。

图结构数据的复杂性对现有机器学习算法提出了重大挑战。图结构数据具有不规则性，每张图大小不同、节点无序，并且图中的每个节点都可以有不同数量的邻节点，这使得一些在图像中容易计算的重要运算（如卷积操作）不能直接应用于图结构数据。此外，现有机器学习算法的核心假设是实例之间彼此独立而图结构数据中的每个实例都与周围的其他实例相关，包含复杂的连接信息，用于捕获数据之间的依赖关系，包括引用、朋友关系和相互作用等。因此，如何利用深度学习方法对图结构的数据进行有效的分析和推理已经引起了广泛的研究与关注。图神经网络（GNN）是处理图结构数据中相邻节点间信息传播和聚合的重要技术，它有效地将深度学习的理念应用于非欧几里得空间的数据上。本节将分类介绍不同类型的图结构，并分析对比不同的图神经网络技术。

5.5.1 图结构定义

图神经网络所处理的数据为在欧氏空间内特征表示为不规则网络的图结构数据。基本的图结构定义为(G,V,E)，其中V代表节点集合，E表示边集合。在空间上，图结构的变化可以从节点和边来进行区分，如边异构的有向图、权重图和边信息图，以及节点异构图。

（1）有向图是指在图结构中，连接节点之间的边包含指向性关系，如图 5-25 所示。有向图节点之间的关联包含方向的传递性关系。对于图神经网络而言，这种传递关系类似于深度学习神经网络中神经元之间的信号传递结构。有向图的输入是各个节点所对应的参数。

（2）权重图是指在图结构中的边包含权重信息，这些权重可以有效地描述节点之间相互作用的可靠程度，定量地表现关系的连接强度，如图 5-26 所示。

（3）边信息图是指在图结构中存在不同结构的边，节点之间的关联关系可以包含权重、方向以及异构的关系，如图 5-27 所示。例如，在一个复杂的社交网络图中，节点之间的关系既可以是单向的“关注”关系，又可以是双向的“朋友”关系。对于包含复杂边信息的图结构，简单的权重限制无法直接表示复杂的关系。

（4）节点异构图是指在图中的节点属于多个不同的类型的图结构，如图 5-28 所示。这种图结构往往可以根据异构节点的类型对其进行向量表示。为了实现节点的向量表示，可以使用 one-hot 等编码方式。

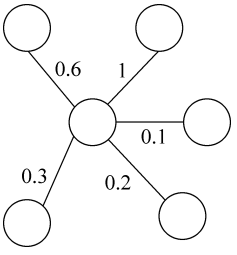


图 5-26 权重图

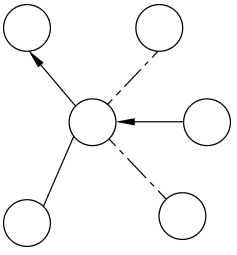


图 5-27 边信息图

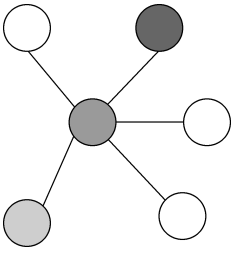


图 5-28 节点异构图

5.5.2 通用的图神经网络框架

图神经网络在深度学习中的应用对于处理非欧几里得数据具有非常重要的意义。特别是，它可以用于传统贝叶斯因果网络的解释的、以定义深度神经网络关系可推理和因果可解释的问题。下面总结和归纳了现有的图神经网络算法，并提出了一个通用的图神经网络结构。这个图神经网络的推理过程包括以下几个步骤。

(1) 图节点预表示：通过图嵌入(graph embedding)的方法对图中每个节点进行嵌入表示。

(2) 图节点采样：对图中每个节点或节点对的正负样本进行采样。

(3) 子图提取：提取图中每个节点的邻节点构建 n 阶子图，其中 n 表示子图包含目标节点的 n 阶(跳)邻节点，从而形成通用的子图结构。

(4) 子图特征融合：对每个输入神经网络的子图进行局部或全局的特征提取。

(5) 生成图神经网络和训练：定义网络的层数和输入输出的参数，并对图数据进行网络训练。

接下来将以几个经典的图神经网络模型为线索，介绍图神经网络的发展历程。

5.5.3　图卷积网络

图卷积网络(GCN)是图神经网络的“开山之作”，它首次成功地将卷积操作引入图结构数据处理中，并给出了具体的推导。其中涉及复杂的谱图理论，推导过程较为复杂，这里不再介绍。尽管推导过程比较复杂，但最终的结果非常简单明了。GCN 将原始图结构的数据 $G=(V,E)$ 映射到一个新的特征空间：

$$f^G \rightarrow f^* \tag{5-32}$$

GCN 的分层传播规则如下：

$$\boldsymbol{H}^{l+1}=\sigma(\check{\boldsymbol{D}}^{-\frac{1}{2}}\check{\boldsymbol{A}}\check{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{l}\boldsymbol{w}^{l}) \tag{5-33}$$

其中，$\check{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}$，$\boldsymbol{A}$ 是原始的邻接矩阵，$\boldsymbol{I}$ 是单位矩阵，即对角线为 1，其余全为 0，$\check{\boldsymbol{D}}$ 是 $\check{\boldsymbol{A}}$ 的度矩阵，$\boldsymbol{H}$ 是每一层所有节点的特征向量矩阵，对于输入层而言，$\boldsymbol{H}^0$ 就等于特征矩阵 $\boldsymbol{X}$，σ 是非线性激活函数，$\boldsymbol{w}^l$ 表示的是当前卷积层变换的可训练参数矩阵。GCN 善于学习编码图的结构信息，因此能够学习到更有效的节点表示，从而在下游任务中相较于传统方法表现出显著的提升。然而，GCN 也存在一些明显的缺点。首先，GCN 需要将整个图放到内存和显存，这会导致大量的内存和显存消耗，因此难以处理大型图；其次，GCN 在训练时，需要知道整个图的结构信息(包括待预测的节点)，这在某些现实任务中并不可行，例如使用今天训练的图模型来预测明天的数据，因为明天的节点信息是无法获取的。

5.5.4　GraphSAGE

为了解决 GCN 存在的两个缺点，研究人员提出了 GraphSAGE(Graph Sample and Aggregate)。在介绍 GraphSAGE 之前，需要先了解归纳式学习(inductive learning)和直推式学习(transductive learning)。由于图数据和其他类型数据有所不同，图数据中的每个节点都可以通过边的关系利用其他节点的信息。这带来了一个问题：当使用 GCN 进行训练时，它输入整个图，并在收集邻节点信息时使用测试和验证集的样本，这被称为直推式学习。然而，大多数机器学习问题都是归纳式学习，因为通常会将样本集分为训练/验证/测试，并且仅使用训练样本进行训练。这种方法的优势在于能够处理新加入的节点，并利用已知节点的信息生成嵌入向量来表示未知节点，而 GraphSAGE 正是采用这种方法实现的。GraphSAGE 是一种归纳式学习框架，其具体实现包括采样(sample)和聚合(aggregate)两个步骤。其中，采样指的是从邻节点中取得样本，而聚合是指获取邻节点的嵌入向量之后如何将这些嵌入向量汇聚以更新节点自身的嵌入向量信息。图 5-29 展示了 GraphSAGE 学习的过程。

第一步，对邻节点进行采样；第二步，使用聚合函数聚合这些邻节点信息以更新节点的表示向量(嵌入)；第三步，根据更新后的节点表示预测节点的标签。

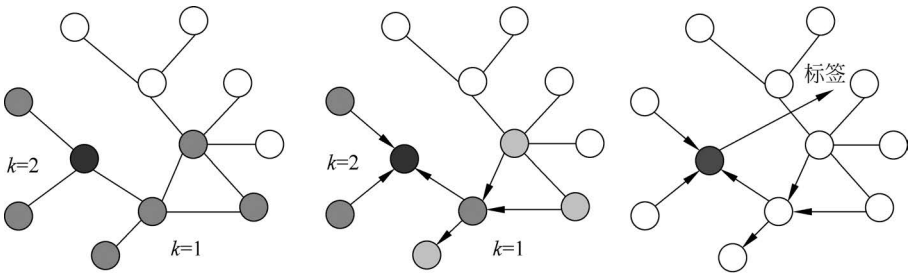


图 5-29 GraphSAGE 学习的过程

GraphSAGE 通过采样机制成功解决了 GCN 需要整个图信息的问题，并克服了 GCN 训练时内存和显存方面的限制。即使对于未知的新节点，GraphSAGE 也能够提供有效的表示。此外，该模型的参数数量与图中节点个数无关，从而使得 GraphSAGE 能够处理更大的图。然而，GraphSAGE 也存在一些缺点。由于每个节点可能具有大量邻节点，GraphSAGE 的采样方法未考虑不同邻节点的重要性差异，因此在聚合计算过程中，邻节点的重要性在不同节点间可能存在差异。

5.5.5 图注意力网络

为了解决 GraphSAGE 在聚合邻节点时未考虑不同邻节点重要性的问题，图注意力网络(Graph Attention Networks，GAT)借鉴了 Transformer 中的注意力机制。如今，注意力机制已经被广泛应用于基于序列的任务中，并具有放大数据中最重要部分影响的优点。在计算图中的每个节点的表示时，GAT 会根据邻节点特征的不同来为其分配不同的权值。GAT 的图卷积运算定义如下：

$$h_i^t = \sigma\left(\sum_{j \in N_i} \alpha(h_i^{t-1}, h_j^{t-1}) \boldsymbol{W}^{t-1} h_j^{t-1} \right) \tag{5-34}$$

其中，α 是一个注意力函数，它自适应地控制相邻节点 j 对于节点 i 的贡献。为了使模型更好地适应不同的子空间，并提高其拟合能力，该方法引入了多头注意力机制。这意味着同时使用多个自注意力计算，然后将计算的结果合并(连接或者求和)：

$$h_i^t = \Vert_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_k(h_i^{t-1}, h_j^{t-1}) \boldsymbol{W}_k^{t-1} h_j^{t-1} \right) \tag{5-35}$$

此外，由于 GAT 结构的特性，它无须使用预先构建好的图，因此 GAT 既适用于直推式学习，又适用于归纳式学习。训练 GCN 时无须了解整个图结构，只需知道每个节点的邻节点即可。

参考文献

[1] BENÍTEZ J M, CASTRO J L, REQUENA I. Are artificial neural networks black boxes? [J]. IEEE Transactions on Neural Networks, 1997, 8(5): 1156-1164.

[2] SIMON H. Neural networks: a comprehensive foundation[M]. Upper Saddle River: Prentice Hall, 1998.

[3] SHARMA V, RAI S, D A. A comprehensive study of artificial neural networks[J]. International Journal of Advanced Research in Computer Science and Software Engineering, 2012, 2(10): 278-284.

[4] MCCULLOCH W S, PITTS W. A logical calculus of the ideas immanent in nervous activity[J]. The Bulletin of Mathematical Biophysics, 1943, 5: 115-133.

[5] EMILE A, KORST J. Simulated annealing and Boltzmann machines: a stochastic approach to

combinatorial optimization and neural computing[M]. New York: John Wiley and Sons, Inc., 1989.
[6] GERALD M, ELROD D W, TRENARY R G. Computational neural networks as model-free mapping devices[J]. Journal of Chemical Information and Computer Sciences, 1992, 32(6): 732-741.
[7] SVOZIL D, KVASNICKA V, POSPICHAL J. Introduction to multi-layer feed-forward neural networks[J]. Chemometrics and Intelligent Laboratory Systems, 1997, 39(1): 43-62.
[8] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015, 521(7553): 436-444.
[9] WERBOS P. New tools for prediction and analysis in the behavioral science[D]. Cambridge: Harvard University, 1974.
[10] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Efficient estimation of word representations in vector space[C]. Proceedings of International Conference on Learning Representations, 2013.
[11] DEVLIN K J, CHANG M W, TOUTANOVA L K. Bert: pre-training of deep bidirectional transformers for language understanding[C]. Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, 1: 4171-4186.
[12] O'SHEA K, NASH R. An introduction to convolutional neural networks[J]. arXiv, 2015.
[13] HUBEL D H, WIESEL T N. Receptive fields and functional architecture of monkey striate cortex[J]. The Journal of physiology, 1968, 195(1): 215-243.
[14] LECUN Y, BOSER B, DENKER J S, et al. Backpropagation applied to handwritten zip code recognition[J]. Neural computation, 1989, 1(4): 541-551.
[15] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[C]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
[16] ZEILER M D, ROB F. Visualizing and understanding convolutional networks[C]. Proceedings of European Conference on Computer Vision, 2014, 13: 818-833.
[17] HOCHREITER S. SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.
[18] CHUNG J, GULCEHRE C, CHO K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[C]. In Conference and Workshop on Neural Information Processing Systems Workshop on Deep Learning, 2014.
[19] MATTHEW E P, MARK N, MOHIT I, et al. Deep contextualized word representations[C]. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018, 1: 2227-2237.
[20] PENNINGTON J, SOCHER R, MANNING C D. Glove: global vectors for word representation[C]. Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2014: 1532-1543.
[21] ZHOU J, CUI G, HU S, et al. Graph neural networks: A review of methods and applications[J]. AI open, 2020, 1: 57-81.
[22] WU Z, PAN S, CHEN F, et al. A comprehensive survey on graph neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 32(1): 4-24.
[23] ZHANG Z, CUI P, ZHU W. Deep learning on graphs: a survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 34(1): 249-270.
[24] KIPF T N, WELLING M. Semi-supervised classification with graph convolutional networks[C]. Proceedings of the International Conference on Learning Representations, 2016.
[25] HAMILTON W, YING Z, LESKOVEC J. Inductive representation learning on large graphs[C]. Advances in Neural Information Processing Systems, 2017, 1-11.
[26] VELICKOVIC P, CUCURULL G, CASANOVA A, et al. Graph attention networks[C]. Proceedings of the International Conference on Learning Representations, 2018, 1-12.
[27] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]. Advances in Neural Information Processing Systems, 2017, 1-11.