第5章　深度学习

5.1　深度学习概论

深度学习是机器学习的一个分支，它通过构建多层神经网络模型[1-2]来模拟人脑的学习过程。这些神经网络由大量的神经元（或称节点）相互连接而成，能够自动地从输入数据中提取特征，并进行高效的信息处理。与传统机器学习算法相比，深度学习模型具有更强的表达能力和泛化能力，能够处理更加复杂和抽象的任务。

在深度学习中，数据通过逐层神经元进行传递和变换，每一层都会对输入数据进行一定的处理，从而提取出更高层次的特征。这种层次化的特征提取方式使得深度学习模型能够自动地学习数据的内在规律和模式，而无须以人工方式进行特征工程。深度学习模型的核心是其层次结构。一个基本的深度学习模型由输入层、隐藏层和输出层组成，其中，输入层接收原始数据，隐藏层负责提取特征和学习数据的抽象表示，而输出层则根据这些特征做出预测或分类。随着技术的发展，深度学习模型的层数越来越多，形成了所谓的“深度”网络，从而使模型能够捕捉更加细微和复杂的数据模式。

深度学习同样可分为监督与非监督学习，其中监督学习包括深度前馈网络、卷积神经网络、循环神经网络等；无监督学习包括深度信念网络、深度玻尔兹曼机，深度自编码器等[3]。

深度学习的应用领域非常广泛，从图像识别、语音识别到自然语言处理、自动驾驶等，深度学习都在发挥着重要作用。例如，在图像识别领域，深度学习模型能够识别和分类成千上万种不同的物体；在自然语言处理领域，深度学习模型能够理解和生成人类语言，实现机器翻译和聊天机器人等功能。

5.2　深度学习发展历程

深度学习是机器学习领域中的一个新的研究方向。一般认为，至目前，深度学习已经经历了三次发展阶段，即起源阶段、发展阶段、爆发阶段。

5.2.1　起源阶段

如前所述，1943 年提出的 MP 模型[4]作为人工神经网络的起源，开创了人工神经网络的新时代，也奠定了神经网络模型的基础。

1949年，加拿大著名心理学家唐纳德·赫布在 *The Organization of Behavior* 中提出了一种基于无监督学习的规则——赫布学习规则。赫布规则模仿人类认知世界的过程建立一种“网络模型”，该网络模型针对训练集进行大量的训练，并提取训练集的统计特征，然后按照样本的相似程度进行分类，把相互之间联系密切的样本分为一类，这样就把样本分成了若干类。赫布规则与“条件反射”机理一致，为以后的神经网络学习算法奠定了基础，具有重大的历史意义。

20世纪50年代末，在MP模型和赫布规则的研究基础上，美国科学家弗兰克·罗森布拉特设计了一种类似于人类学习过程的算法——感知机学习，对神经网络的发展具有里程碑式的意义。本书第4章4.1.2节对感知机进行了介绍，本节不再赘述。

随着研究的不断深入，1969年，AI之父马文·明斯基和LOGO语言的创始人西蒙·派珀特共同编写了一本书《Perceptrons》，在书中他们证明了单层感知机无法解决线性不可分问题。由于这个致命的缺陷以及没有及时将感知机推广到多层神经网络中，20世纪70年代，人工神经网络进入了第一个寒冬期，人们对神经网络的研究也停滞了将近20年。

5.2.2 发展阶段

1982年，著名物理学家约翰·霍普菲尔德发明了Hopfield神经网络。Hopfield神经网络是一种结合存储系统和二元系统的循环神经网络。Hopfield网络也可以模拟人类的记忆，根据激活函数的选取不同，可分为连续型和离散型两种不同类型，分别用于优化计算和联想记忆。遗憾的是，由于容易陷入局部最小值等缺陷，该算法在当时并未引起足够的重视。

如前所述，直到1986年，深度学习之父杰弗里·辛顿提出的适用于多层感知机的反向传播算法在传统神经网络正向传播的基础上增加了误差的反向传播过程。反向传播过程不断地调整神经元之间的权值和阈值，直到输出的误差减小到允许的范围之内，或达到预先设定的训练次数为止。反向传播算法完美地解决了非线性分类问题，让人工神经网络再次引起了人们的广泛关注。

然而，由于20世纪80年代计算机的硬件水平有限，相关理论的研究也不够深入，人工神经网络的应用和发展受到了很大的限制。另一方面，90年代中期，以支持向量机（SVM）为代表的其他浅层机器学习算法陆续出现，并在分类、回归等问题上均取得了很好的效果。相比之下，人工神经网络的发展再次进入了瓶颈期。

5.2.3 爆发阶段

如前所述，2006年，杰弗里·辛顿以及他的学生鲁斯兰·萨拉赫丁诺夫正式提出了深度学习的概念[5]。该深度学习方法的提出立即引起了巨大反响，以斯坦福大学、多伦多大学为代表的众多世界知名高校纷纷投入巨大的人力、财力进行深度学习领域的相关研究，之后又迅速蔓延到工业界中。

2012年，在著名的ImageNet图像识别大赛中，杰弗里·辛顿领导的小组采用深度学习模型AlexNet一举夺冠。同年，由斯坦福大学著名的吴恩达教授和世界顶尖计算机专家杰夫·迪恩共同主导的深度神经网络在图像识别领域取得了惊人的成绩，在ImageNet评测中

成功地把错误率从 26%降低到了 15.3%。深度学习算法在世界大赛的脱颖而出，也再一次吸引了学术界和工业界对于深度学习领域的关注。

随着深度学习技术的不断进步以及数据处理能力的不断提升，2014 年，Facebook 基于深度学习技术的 DeepFace 项目，在人脸识别方面的准确率已经能达到 97%以上。这样的结果也再一次证明了深度学习算法在图像识别方面的绝对优势。

2016 年，随着谷歌公司基于深度学习开发的 AlphaGo 以 4∶1 的比分战胜了国际顶尖围棋高手李世石，深度学习技术得到了学术界之外的更广泛关注。后来，AlphaGo 又接连和众多世界级围棋高手过招，均取得了完胜。这也证明了在围棋界，基于深度学习技术的机器人已经超越了人类。

2017 年，基于强化学习算法的 AlphaGo 升级版 AlphaGo Zero 横空出世。其采用“从零开始”“无师自通”的学习模式，以 100∶0 的比分轻而易举打败了之前的 AlphaGo。除了围棋，它还精通国际象棋等其他棋类游戏，可以说是真正的棋类“天才”。此外，在这一年，深度学习的相关算法在医疗、金融、艺术、无人驾驶等多个领域均取得了显著成果。所以，也有专家把 2017 年看作是深度学习甚至是人工智能发展最为突飞猛进的一年。

5.3　深度神经网络基本原理

5.3.1　深度神经网络核心知识

深度神经网络是一种具有多层次复杂结构的神经网络，可以将其认为是一种堆叠神经网络，由输入层、输出层和多个隐藏层组成。神经网络的每一层都由很多人工神经元构成，这些神经元通过可学习的参数连接起来，并传递到下一层。在训练过程中，深度神经网络不断更新网络中的参数，最终获得将输入数据处理为目标输出结果的能力。本节将从神经元、激活函数、多层网络结构三个部分介绍深度神经网络的核心知识。

1. 神经元

神经元是深度神经网络中最基础的组成单元，模拟了生物领域神经细胞的结构和特性，现有研究[6]表明，成年男性的大脑平均包含约 861 亿神经元。在人脑神经系统中，每个神经元通过突触与其他神经元相连，通过发送电信号改变神经元的状态，并向其他神经元传递信息。具体来说，神经元的状态可分为“兴奋”和“抑制”两种，当接收到的电信号经由神经细胞处理达到一个阈值时，神经元的状态将被激活为“兴奋”，并向其他神经元发送电信号。图 5-1 展示了神经元结构，其中树突可接收来自其他神经元的信息，轴突则可向其他神经元传递信息。

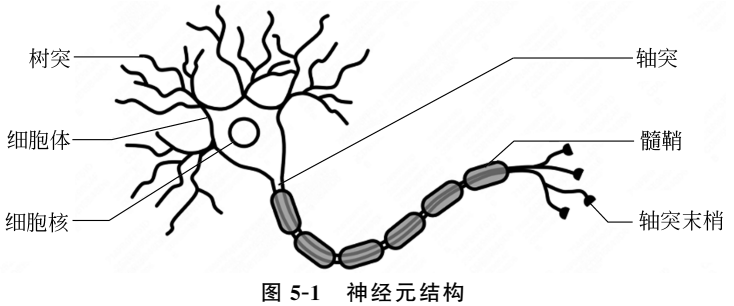


图 5-1　神经元结构

如前所述，基于人脑网络中的神经细胞，1943 年沃伦·麦卡洛克和沃尔特·皮茨提出了“MP 神经元模型”，用于模拟大脑神经元的工作机制[4]。如图 5-2 所示，在 MP 神经元模型中，神经元接收来自其他 m 个神经元的输入信号，并按照不同权重 $\boldsymbol{w}$ 对接收到的输入信号 x 加权求和，使用偏置项 b 对输入进行调节，计算得到刺激强度 S，具体表示如下：

$$S=\sum_{i=1}^{m}\boldsymbol{w}_i x_i+b \tag{5.1}$$

之后，通过激活函数 σ 得到输出：

$$y=\sigma(S) \tag{5.2}$$

输出 y 会继续传递给其他相邻的神经元，并作为它们的输入。对于每个神经元，权重 $\boldsymbol{w}$ 和偏置项 b 都是可训练的参数，通过对网络的训练和优化，参数不断更新。为了模拟人脑神经系统，通常会将多个神经元排列起来作为一层，然后将多层叠加构成一个复杂的神经网络，用于解决多种机器学习问题。

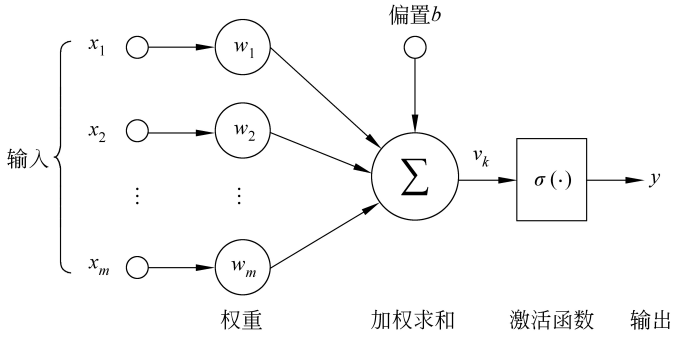


图 5-2　MP 神经元模型

2. 激活函数

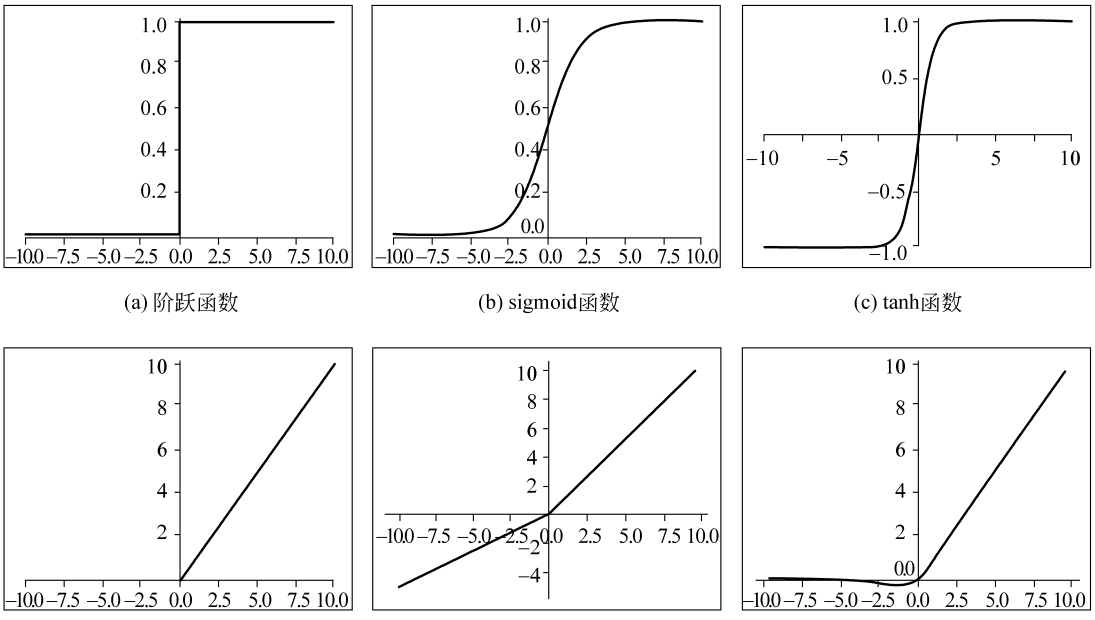
激活函数在神经元中非常重要，它可以控制神经元的输出幅度，决定一个神经元是否被激活[7]。在 MP 神经元中，激活函数为 0 或 1 的阶跃函数，它将输入映射为 0 或 1，分别表示神经元的“抑制”或“兴奋”状态。然而，阶跃函数是非连续、非光滑的。由于近年来流行的深度神经网络普遍需要反向传播进行优化，为增强网络的表示能力和学习能力，激活函数往往选择为连续可导，或者仅仅少数点上不可导的非线性函数。图 5-3 展示了几种常见的激活函数，下面将分别介绍。

(1) sigmoid 函数。

sigmoid 函数又名挤压函数，它将较大范围的输入值挤压到区间(0,1)。输入越大，结果越接近 1；输入越小，结果越接近 0，这与生物神经元的“兴奋”与“抑制”状态相似。相比于阶跃函数，sigmoid 函数连续可导，其具体定义如下。

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{5.3}$$

然而，sigmoid 函数的输出并不是以 0 为中心，而是恒大于 0 的，这会导致后面神经元的输入产生偏置偏移，从而使梯度下降的收敛速度变慢。此外，当输入较大或较小时，输出的梯度较小，并逐渐趋近于 0，不利于权重更新。



(a) 阶跃函数　(b) sigmoid函数　(c) tanh函数

(d) ReLU函数　(e) Leaky ReLU函数　(f) swish函数

图 5-3　常见激活函数

（2）tanh 函数。

tanh 函数对 sigmoid 函数的非零中心问题进行改进，它同样是挤压函数，输出区间是（−1，1）。函数的具体定义为

$$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{5.4}$$

tanh 函数可以看作是 sigmoid 函数的放大并平移：

$$\tanh(x)=2\sigma(2x)-1 \tag{5.5}$$

遗憾的是，tanh 函数同样面临输入较大或较小时梯度消失的问题。

（3）ReLU 函数。

ReLU 函数（rectified linear unit，修正线性单元）[8]，是当前深度神经网络中较为常用的激活函数。函数具体定义如下：

$$\mathrm{ReLU}(x)=\begin{cases}x, & x\geqslant 0\\ 0, & x<0\end{cases} \tag{5.6}$$

相比于 sigmoid 函数和 tanh 函数，ReLU 不含指数运算，计算过程较为简单，提高了计算效率。此外，当 $x>0$ 时，导数为 1，在一定程度上缓解了梯度爆炸和梯度消失问题。然而，ReLU 函数面临非零中心问题，后面神经元的输入会发生偏置偏移。此外，还有死亡 ReLU 问题，即当 $x<0$ 时，ReLU 在反向传播过程中梯度为 0，后续训练中将永远不能被激活。

（4）Leaky ReLU 函数。

Leaky ReLU 函数是基于 ReLU 函数的改进版本，它主要解决了死亡 ReLU 问题。对于 $x<0$，Leaky ReLU 具有一个很小的梯度 γ。一般情况下，采用一个很小的常数，如 0.01。Leaky ReLU 函数定义如下。

$$\text{LeakyReLU}(x)=\begin{cases}x, & x\geqslant 0\\ \gamma x, & x<0\end{cases} \tag{5.7}$$

（5）swish 函数。

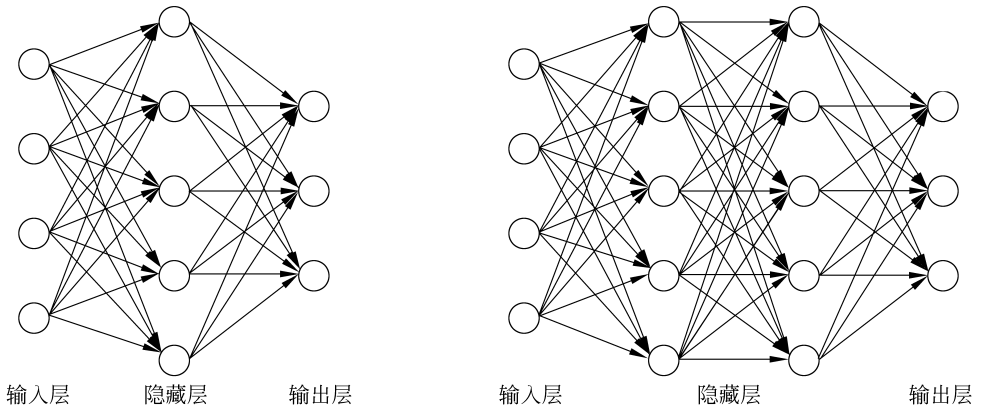
swish 函数[8]是一种自门控激活函数，函数定义如下。

$$\text{swish}(x)=x\sigma(x) \tag{5.8}$$

它采用 sigmoid 函数作为一种门控机制（gating mechanism），当 x 较大时，$\sigma(x)$趋向于 1，swish 函数的结果近似 x 本身，此时 swish 函数近似为 ReLU 函数；当 x 较小时，$\sigma(x)$趋向于 0，swish 函数结果近似 0。由于 swish 激活函数的形式较为复杂，在训练过程中较为耗时。

3. 多层网络结构

为解决复杂的问题，单个神经元的能力往往是不足的，需要众多神经元一起协作实现复杂功能。深度神经网络通常具有多层网络结构，包括一个输入层、一个或多个隐藏层、一个输出层。输入层神经元仅接收输入数据，并不进行数据处理，隐藏层和输出层的神经元对输入数据进行加权和激活运算。如图 5-4 所示，根据隐藏层数量，可分为单隐层网络结构和多隐层网络结构。每层的神经元通常会与下一层的全部神经元相互连接，同一层的神经元之间彼此互不相连，跨层神经元之间也不相连。



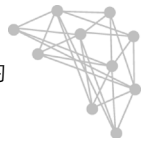
(a) 单隐层网络结构　　(b) 多隐层网络结构

图 5-4　多层网络结构示意图

5.3.2　前向神经网络与反馈神经网络

1. 前向神经网络

前向神经网络又称为前馈神经网络，是最早发明的一种简单的人工神经网络，具有单向多层结构的神经网络。在前向神经网络中，每个神经元按接收信息的先后顺序分为不同的组，每一组可以看作是一个网络结构层。数据从输入层开始，每一层中的神经元接收前一层神经元的输出，经过运算处理输出到下一层神经元，不断向后传播直至输出层。整个网络中的信息是从前向后朝一个方向传播的，没有反向的信息传播过程，在网络拓扑结构上是一个有向无环图。图 5-4 即为一种全连接前向神经网络。



前向网络可以看作一个函数，它通过对于简单的非线性函数进行多次复合，可实现输入空间到输出空间的复杂映射。接收输入数据 x 后，输入层不对数据进行运算，因此输入层的输出表示为 $a^0=x$。随后的隐藏层和输出层对输入信息进行加权运算，并通过激活函数得到输出，第 l 层的输出可以表示为

$$a^l=\sigma^l(h^l) \tag{5.9}$$

$$h^l=\boldsymbol{W}^l a^{l-1}+b^l \tag{5.10}$$

其中，σ^l 表示第 l 层的激活函数，$\boldsymbol{W}^l$ 和 b^l 分别为第 l 层的权重参数和偏置参数。

在前向神经网络中，通过每一层的运算和传递，最后的结果 y 可以视为多层运算的叠加，最终的输出结果由输入 x，网络参数 $\boldsymbol{W}$ 和 b 共同决定：

$$y=\sigma^l(\sigma^{l-1}(\cdots\sigma^1(\boldsymbol{W}^1a^0+b^1))) \tag{5.11}$$

$$y=\varphi(x,\boldsymbol{W},b) \tag{5.12}$$

前向神经网络结构简单，应用广泛，且具有很强的拟合能力，能够拟合常见的连续非线性函数，以及平方可积函数。然而，前向网络存在如下几点不足：其一，网络的每一层都捕捉前一层的全局特征，导致网络参数量较大；其二，当输入数据维度较高时，前馈网络的神经元个数将大幅增长，同样会增大网络参数量，导致训练收敛速度慢，甚至无法收敛。此外，尽管前向神经网络具有拟合复杂非线性函数的能力，但是由于不同的非线性问题的复杂性，很多时候前向神经网络在实际应用中难以达到期望的精度。

2. 反馈神经网络

反馈神经网络是一种从输出到输入具有反馈连接的神经网络，其中的神经元不仅可以接收其他神经元的信息，还可以接收自己的历史信息。具体来说，在反馈神经网络中神经元可以互连，有些神经元的输出会被反馈至同层甚至前层的神经元。在拓扑结构上，可以用一个有向循环图或无向图来表示反馈神经网络。常见的反馈神经网络有循环神经网络、Hopfield 网络、玻尔兹曼机、受限玻尔兹曼机等。图 5-5 展示了一种常见的反馈神经网络结构。

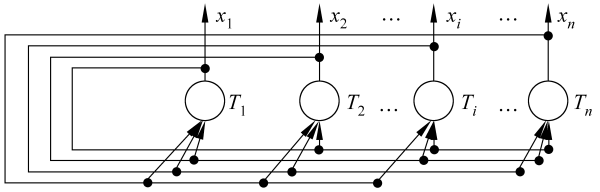


图 5-5　一种反馈神经网络结构示意图

相比前向神经网络，反馈神经网络主要有以下区别：①前向神经网络中神经元只接受来自上一层的数据，处理后传入下一层，数据仅单向流动；反馈神经网络中的神经元连接方式更多样，数据可以在同层间传递或反馈至上一层，信息传播可以是单向或双向传递。②前向神经网络不考虑输出与输入在时间上的滞后效应，而只表达输出与输入的映射关系；反馈神经网络的神经元具有记忆功能，考虑输出与输入之间在时间上的延迟，在不同时刻神经元具有不同的状态，需要利用动态方程来描述系统的模型。

5.3.3 反向传播算法

反向传播算法基于梯度下降策略最小化损失函数，并优化网络参数，用于训练包括前向神经网络在内的多层神经网络模型。

给定一组训练样本 $D=\{(x_i,y_i)\}_{i=1}^{N}$ 和一个神经网络模型 $f(x,\theta)$，其中 x_i 和 y_i 分别代表第 i 个样本的输入和目标输出，θ 代表神经网络模型中的参数。对于每个样本来说，输入 x_i 到模型中，经过神经网络的计算将会输出一个预测结果 $\hat{y}_i=f(x_i,\theta)$。基于此，可以利用损失函数来计算预测结果 $\hat{y}_i$ 和目标输出 y_i 之间的差异 $C(\hat{y}_i,y_i,\theta)$。通过对所有样本的损失进行累加，可以得到模型的整体损失 $\mathcal{L}(\theta)=\sum_{i=1}^{N}C(\hat{y}_i,y_i,\theta)$，$\mathcal{L}(\theta)$ 越小，则认为模型越好。

模型训练的目标是为了减小预测结果和实际输出之间的差异，也就是将损失函数 $\mathcal{L}(\theta)$ 作为优化目标，利用优化算法迭代寻找使损失函数最小的模型参数。本节将首先介绍梯度下降法优化算法。随后，进一步介绍用于高效计算多层神经网络模型梯度的方法，即反向传播算法。

1. 梯度下降法

3.3.2 节已经介绍了梯度下降法，本节不再赘述。在反向传播算法中，梯度下降法的整体流程如下：已知神经网络模型 $f(x,\theta)$，梯度下降法首先随机初始化一组参数 θ^0，其中上标 0 表示初始参数。随后，计算初始参数下模型在训练样本集合 $D=\{(x_i,y_i)\}_{i=1}^{N}$ 上的损失 $\mathcal{L}(\theta^0)$，并计算参数对该损失的偏导数，作为其梯度 $\nabla\mathcal{L}(\theta^*)$。在此基础上，使用梯度来更新模型参数：

$$\theta^1=\theta^0-\eta\nabla\mathcal{L}(\theta^0) \tag{5.13}$$

其中，上标 1 代表经过第一次更新后的参数，η 为学习率。基于更新后的参数 θ^1，重新计算模型在训练样本集合上的损失 $\mathcal{L}(\theta^1)$，并计算参数对该损失的梯度 $\nabla\mathcal{L}(\theta^1)$，进而实现第二次参数更新。不断迭代执行上述过程，直到找到一组最优的模型参数 θ^F。

2. 反向传播算法

在梯度下降的优化算法中，非常关键的步骤是计算多层神经网络中各个层上的参数梯度。假设神经网络模型包含 N 层，则模型的参数包括每一层的参数，即 $\theta=\{\theta^{(1)},\theta^{(2)},\cdots,\theta^{(N)}\}$。对于输入样本 x，模型第一层会输出中间结果 $h^{(1)}$，并作为模型第二层的输入，以此类推，模型最后一层的输入 $h^{(N)}$ 即为模型的预测结果，通过计算预测结果和目标输出 y 之间的差异即可得到模型的损失 $\mathcal{L}(\theta)$，然后即可利用反向传播算法，以最小化损失函数为目标来优化模型参数。

反向传播算法的数学基础为函数求导的链式法则。假设 $y=f(\boldsymbol{x})$，$z=g(y)$，根据链式法则，z 对 $\boldsymbol{x}$ 求导可以计算如下：

$$\frac{\partial z}{\partial \boldsymbol{x}}=\frac{\partial z}{\partial y}\cdot\frac{\partial y}{\partial \boldsymbol{x}} \tag{5.14}$$

反向传播算法包括两个主要的步骤：前向传播（forward propagation）和反向传播（backward propagation）。具体算法如算法 5.1。

算法 5.1　反向传播算法

```
//初始化
1.  设置最大迭代次数
2.  初始化神经网络参数 θ,准备输入数据
3.  重复
        //前向传播
4.      输入数据进入神经网络
5.      For 神经网络的第一层(输入层)到最后一层(输出层)
6.          使用本层神经元加权输入并汇总
7.          使用激活函数对汇总后的结果进行非线性变换,得到该层的输出
8.          将该层的输出传递到下一层作为输入
9.      End for
        //计算损失
10.     使用损失函数ℒ(如均方误差、交叉熵等)计算网络输出结果与实际结果之间的差异
        //损失函数的值给出了网络性能的度量
        //反向传播
11.     For 神经网络的最后一层(输出层)到第一层(输入层)
12.         计算损失函数关于本层网络参数(权重和偏置)的梯度
13.         //对于每个神经元,梯度表示了该神经元的参数如何影响整体损失
14.         将本层梯度逆向传递给网络的前一层,使用链式法则计算梯度
15.     End for
        //参数更新
16.     使用梯度下降法更新网络的参数,参数的更新规则可以表示为 θ=θ−α·∇ℒ,其中,θ 是要更新
        的参数,α 是学习率,∇ℒ是损失函数ℒ关于 θ 的梯度
    //迭代
17. 直到网络的性能达到满意的水平,或者达到预设的迭代次数
```

5.4　典型的神经网络

5.4.1　卷积神经网络

1. 整体结构

卷积神经网络 CNN 是一种深层学习架构,专门用于处理具有网格状拓扑结构的输入数据,例如在第 7 章将要介绍的 RGB(red-green-blue)图像。该模型通过一系列层次化的处理步骤,逐步从原始输入数据中提取高级语义特征。

在 CNN 中,数据的前向传播涉及多个阶段的计算,每个阶段都由特定的层执行。首先,卷积层通过滤波器(或称为卷积核)在输入数据上滑动,执行卷积操作,以捕捉局部特征,并生成特征映射(feature map)。接着,汇聚层对特征映射进行降采样,以增强模型对空间变换的不变性,并减少后续层的参数数量。此外,将 ReLU 等非线性激活函数应用于层的输出,以引入非线性特性,从而使网络能够学习复杂的函数映射。这些层的组合构成了 CNN 的核心,每一层都对输入数据执行特定的变换,将原始数据中的低级特征逐渐转化为更抽象的高级表示,这一连续的变换过程称为前馈传播,它对应于网络的正向传播阶段。在网络的最后一层,即输出层,模型的目标任务,如分类或回归,被转化为一个目标函数。该层的激活函数

根据任务的具体需求而定，例如采用 softmax 函数用于多分类问题等。

在训练过程中，网络通过计算预测输出与真实标签之间的误差或损失来优化层间的权重和偏置参数。这一过程称为监督学习。损失函数，如交叉熵损失或均方误差，量化了模型预测的准确性。利用反向传播算法，损失信息从输出层反向传递至网络的每个层，根据梯度下降原则更新网络参数。通过迭代的前向传播和反向传播过程，网络不断学习，并调整参数，直至在给定任务上达到满意的性能，实现模型的收敛。

一个典型的卷积网络由卷积层、汇聚层、全连接层交叉堆叠而成。图 5-6 给出了一个简单的卷积神经网络示例，其中卷积块为连续的 1 个卷积层和 1 个汇聚层，卷积网络包含 2 个连续的卷积块，后面接着 3 个全连接层。

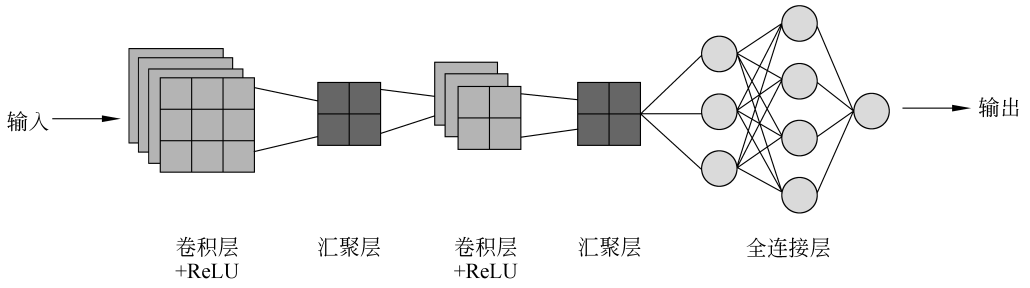


图 5-6　卷积神经网络示例

2. 卷积神经网络的基本组件

（1）卷积层。

在 CNN 的架构中，卷积层承担着从输入数据中提取局部特征的关键作用。每个卷积核（convolutional kernel）或滤波器（filter）都可以视为一个特定的特征检测器，它们在图像的局部区域内滑动，以识别和响应特定的模式或特征。

在组织结构上，尽管卷积层中的神经元与全连接网络中的神经元一样，本质上是一维实体，但考虑到 CNN 主要处理的是图像数据，而图像数据具有二维的空间结构，因此，为了更有效地捕捉图像的局部空间信息，卷积层中的神经元通常被组织成三维的体积结构，其维度为高度（H）×宽度（W）×深度（D）。这种结构由 D 个二维的特征映射组成，每个特征映射对应于输入图像中的一个特定特征的响应图。

特征映射是指经过卷积操作后，从图像（或其他特征映射）中提取出的一组特征。每个特征映射代表了一类特定的图像特征，例如边缘、纹理或形状等。为了增强网络对复杂图像特征的表示能力，通常在每一层中使用多个不同的特征映射，从而能够从多个角度和尺度捕捉图像的特征。

在 CNN 的输入层中，特征映射直接对应于图像本身。对于灰度图像，输入层由一个特征映射组成，因此其深度 D 等于 1。而对于彩色图像，每个颜色通道都会产生一个特征映射，因此输入层的深度 D 等于 3。这种基于多特征映射的输入表示为网络提供了丰富的视觉信息，也为后续的层次化特征提取和学习奠定了基础。通过这种方式，卷积层不仅能够提取图像的局部特征，还能够通过堆叠多个卷积层来构建更加复杂和抽象的特征表示，从而实现对图像数据的深入理解。