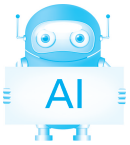
第 5 章




# 人工神经网络

## 本章学习目标

- 了解人工神经网络的发展历程。
- 理解感知机的工作原理和局限性。
- 掌握 BP 神经网络的结构和 BP 学习算法。
- 掌握卷积神经网络的结构和运算过程。

深度学习(deep learning)是近年来计算机学科领域中发展最迅猛的热门研究方向之一,在人工智能的许多应用领域都取得了令人瞩目的进展。深度学习是机器学习的一个分支,而机器学习又是实现人工智能的必经路径,故而人工智能、机器学习和深度学习之间是依次包含的关系。人工智能的研究内容包括机器学习、知识工程、搜索策略、推理、规划、模式识别、组合调度问题、机器人学等;传统机器学习的研究方向主要包括人工神经网络、决策树、随机森林、支持向量机、贝叶斯学习、关联规则、Boosting 与 Bagging 集成学习算法等,而深度学习的概念正是起源于人工神经网络。由于深度学习在计算机视觉、语音识别、机器翻译、图像处理、自然语言处理等方面获得成功应用,在学术圈引起了巨大的反响,因此人工神经网络的研究热度反而超过了其他传统方法,在机器学习研究中占据了主流的地位。

本章首先介绍人工神经网络的发展历程、感知机与神经网络的基础知识,然后介绍 BP 神经网络及其学习算法和卷积神经网络。

## 5.1 人工神经网络的发展历程

1943 年,美国神经生理学家麦卡洛克(McCulloch)和数理逻辑学家皮茨(Pitts)在分析与总结了生物神经元的激活方式后,合作设计了第一个人工神经元模型。它由固定的结构和权重组成,是一种基于阈值逻辑运算的数学模型,称为 MP 模型。虽然 MP 模型没有学习机制,但它开创了用电子装置模仿人脑结构和功能的新途径,是最早的人工神经网络雏形,开创了人工神经网络研究的时代。

1949 年,加拿大心理学家赫布(Hebbian)提出了大名鼎鼎的赫布理论,即“突触前神经元向突触后神经元的持续重复的刺激,可以导致突触传递效能的增加。”简单地说,就是:突触前给的刺激越强,突触后的反应越大。赫布理论解释了在学习过程中脑神经元所发生

的变化，为后续研究神经网络学习算法奠定了基础，具有重大历史意义。

受到赫布理论的启发，1957 年，美国心理学家罗森布拉特(Rosenblatt)在康奈尔航空实验室(Cornell Aeronautical Laboratory)提出了感知机(perceptron)，这是第一个**人工神经网络**。还为其设计了第一个神经网络学习算法，首次将神经网络研究从纯理论探讨推向工程实现。感知机本质是一种仅包含输入层和输出层的二元线性分类器，也称为**单层神经网络**。感知机的输入层可以包含多个单元，而输出层只有一个单元。与 MP 模型不同的是，感知机模型可以利用学习算法自动更新输入的权重和阈值。此后，神经网络的研究进入了第一次高潮。

但好景不长，1969 年，明斯基与派普特的《感知机：计算几何导论》指出了感知机的局限性，使神经网络的研究走向衰落，导致人工智能的连接主义研究流派陷入了长达十多年的低谷期。

1974 年，韦伯斯在其博士论文提出利用反向传播算法训练多层神经网络，但他并未公开发表相关的学术论文，几乎无人了解这项研究成果。

1980 年，日本学者福岛邦彦(Kunihiko Fukushima)提出了名为 Neocognitron 的神经网络，被认为是最早的研究卷积神经网络的工作，也是最早的深度神经网络模型之一。Neocognitron 用于处理图像，其重要特性是平移不变性。为此，福岛邦彦获得了 2021 年鲍尔科学成就奖，以表彰其对深度学习的巨大贡献，尤其是他提出的极具影响力的卷积神经网络架构。

1982 年，加州理工学院的霍普菲尔德教授提出 Hopfield 神经网络，可用于实现联想记忆和优化计算，在旅行商问题上获得了突破。尤其是 1984 年又用模拟集成电路实现了 Hopfield 神经网络，有力地推动了神经网络的研究，连接主义迎来了第二次高潮。

受此启发，辛顿于 1984 年提出了一种随机型的 Hopfield 网络，即玻尔兹曼机。它是一种反馈神经网络，借鉴了模拟退火法的思想，具有一定的“跳出局部最优”的能力。玻尔兹曼机的特点是：一是包含显层与隐层两层结构，显层代表输入和输出，隐层则被理解为数据的内部表达；二是其神经元是布尔型的，即只能取 0 或 1。

1986 年，鲁梅尔哈特、辛顿和威廉姆斯重新独立地提出了多层神经网络的学习算法——BP 算法，采用 sigmoid 函数代替阶跃函数作为激活函数，可以解决多层神经网络中参数优化的问题，且成功解决了非线性分类问题。更值得一提的是，同年，他们在《自然》期刊上联名发表了《通过反向传播误差学习表示》(*Learning Representations by Back-Propagating Errors*)的经典学术论文，因其精确、清晰的观点陈述而令 BP 算法传播甚广。

1989 年，杨乐昆采用 BP 算法训练 LeNet 系列的卷积神经网络，并将其成功应用于美国邮政业务中手写邮政编码数字的识别，随后该网络又成功应用于银行 ATM 机中支票上手写金额数字的识别。杨乐昆构建的 LeNet-5 模型成为现代卷积神经网络的基础，这种卷积层、池化层堆叠的结构可以保持输入图像的平移不变性，且能自动提取图像特征。为此，他被誉为“卷积神经网络之父”。

但在此后的十多年里，由于 BP 算法仍然存在一些问题，例如梯度消失、梯度爆炸等，再加上当时算力不足，使得该算法只适合于训练浅层神经网络，其应用受到了许多限制。相比之下，万普尼克于 1995 年提出的支持向量机可以通过核(kernel)技巧将非线性问题转换成线性问题，其理论基础清晰、证明完备、可解释性好，获得了广泛的认同。同时，统计机器学

习专家从理论角度怀疑神经网络的泛化能力，使得神经网络的研究第二次陷入低谷。

直到进入21世纪，随着大数据时代的来临以及计算机算力的大幅提升，神经网络的研究迎来了第三次高潮。2006年，辛顿教授及萨拉赫丁诺夫发表的论文首次提出了“深度信念网络”(deep belief network)，它是由多个受限玻尔兹曼机(restricted Boltzmann machine)串联堆叠而组成的一个深度网络。深度信念网络在分类任务上的性能超过了传统经典的浅层学习模型(如支持向量机)，引起了学术圈的广泛关注。与传统的训练方式不同，深度信念网络利用“预训练”技术为神经网络中的权值找到一个接近最优解的值，然后再采用“微调”技术对整个网络进行优化训练，逐层预训练的技巧极大地提高了神经网络的泛化能力。随着神经网络层数的不断加深，辛顿将这种深层神经网络上的学习方法命名为“深度学习”。从此，连接主义研究学派开始大放异彩。通常，将包含多个(大于3即可)隐藏层的人工神经网络称为**深度神经网络**，在这样的网络上学习的过程称为**深度学习**。

2012年，辛顿教授及其学生提出的AlexNet模型在ILSVRC比赛的图像分类组一举夺魁。从此深度学习不仅是学术圈的研究热点，还得到了产业界的广泛关注。

深度学习发展至今，神经网络的层数不断加深，模型性能不断提升，甚至在图像分类任务上的能力已超过了人类。2012—2017年，ImageNet图像分类的Top-5(即排名前5的分类标签)错误率从28%降到了3%，目标识别的平均准确率从23%上升到了66%。此外，深度学习方法在自然语言处理、机器翻译、无人驾驶、语音识别、生物信息学、医学影像分析与金融大数据分析等方面也都有广泛而成熟的应用。

## 5.2　感知机与神经网络

科学家受到人脑或生物神经元结构和学习机制的启发，提出了人工神经元的数学模型，又在此基础上增加了学习机制，研制出了可运行的感知机，即单层人工神经网络。最后，将若干个感知机连接在一起，形成了人工神经网络。

人工神经网络，简称神经网络(neural network，NN)或类神经网络，是模拟人脑或生物神经网络的学习机制而建立起来的一种运算模型。它由大量简单的信息处理单元按照一定拓扑结构相互连接而组成人工网络。

### 5.2.1　生物神经元结构

生物学家早在20世纪初就发现了生物神经元的结构，神经元(neuron)也称为神经细胞，其结构如图5.1所示。

一个神经元通常由一个细胞体(soma，也称为体细胞)；多个树突(dendrites)和一条长长的轴突(axon)组成。细胞体是神经元的主体部分，由细胞核、细胞质、细胞膜等组成。**树突**是由细胞体向外伸出的许多较短的分支，相当于细胞的输入端，用于接收信息，树突的各个部位都能接收其他神经元的冲动。**轴突**是细胞体向外伸出的最长的一条分支，也叫神经纤维，用于发送信息。轴突尾端有许多末梢，称为突触，是专门用于与其他神经元的树突相连接的组织，并将神经冲动信号传递给其他神经元。典型的轴突长1cm，是细胞体直径的100倍。一个神经元通过突触与$10\sim10^5$个其他神经元相连接。神经冲动只能由前一个神经元的突触传向下一个神经元的树突或细胞体，不能反方向传递。

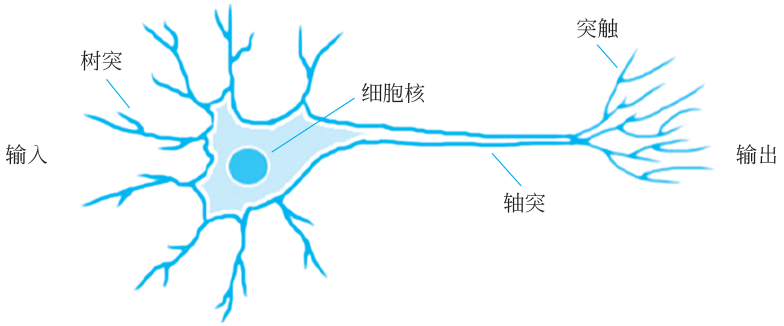



图 5.1 生物神经元结构

神经元有两种常规的工作状态：兴奋状态和抑制状态。当传入的神经冲动使细胞膜电位升高超过一个“阈值”时，该细胞就会被激活，进入兴奋状态，产生神经冲动，并由轴突经过突触输出；当传入的神经冲动使细胞膜电位下降到低于一个“阈值”时，该细胞就进入抑制状态，不输出神经冲动。

## 5.2.2 神经元数学模型——MP 模型

人工神经元是构成人工神经网络的基本单位，它模拟生物神经元的结构和特性，可以接收一组输入信号，并产生输出。1943 年提出的 MP 模型是模仿生物神经元结构而建立的第一个人工神经元数学模型，其结构如图 5.2 所示。

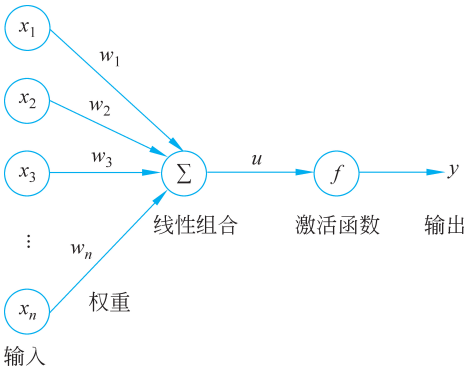



图 5.2 神经元数学模型(MP 模型)

MP 模型可以接收多路输入信号。假设一个神经元同时接收的 $n$ 个输入信号用向量 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$ 表示，则所有输入信号的线性组合称为该神经元的净输入(net input)，记为 $u\in\mathbf{R}$，计算公式如公式(5.1)所示。

$$u=\sum_{j=1}^{n}w_jx_j+\theta \tag{5.1}$$

其中，$w_1,w_2,\cdots,w_n$ 为该神经元各个输入信号的权值，$\theta\in\mathbf{R}$ 为偏置。然后，净输入 $u$ 会被函数 $f(x)$ 转换为输出值 $y$，该函数称为激活函数(activation function)。在 MP 模型中，激活函数 $f(x)$ 采用非线性的阶跃函数，其表达式如公式(5.2)所示。

$$\begin{aligned}&y=f(u)\\&f(x)=\begin{cases}1, & x>0\\0, & x\leqslant 0\end{cases}\end{aligned} \tag{5.2}$$

阶跃函数具有二值化输出，当净输入值大于 0 时，输出 1；当净输入值小于或等于 0 时，输出 0。因此，如果采用阶跃函数作为激活函数，就得到了一个非常简单的二类分类器，它可以根据一个输入向量做出明确的分类决策。但是，阶跃函数有一个致命的缺陷：不连续、不平滑，因为它在 0 点处的导数是无穷大，除了 0 点处之外，导数都是 0，这意味着：若学习算法采用基于梯度的优化方法，是不可行的。

在 MP 模型中引入激活函数的目的是：用于模拟生物神经元的工作机制，当电位高于一个设定的阈值时，则进入兴奋状态，输出信号；否则进入抑制状态，不输出信号。

可见，MP 模型中的输入和输出数据只能是二值化数据 0 或 1，而且网络中的权重、阈值等参数都需要人为设置，无法从数据中学习得到。MP 模型的激活函数是一个简单的阶跃函数。MP 模型只能处理一些简单的分类任务，例如线性二分类问题，但无法解决线性不可分问题。

### 5.2.3　感知机

由一个神经元构成的神经网络称为**感知机**，也称为**单层神经网络**。1957 年提出的感知机是第一个工程实现的**人工神经网络**，可以运行感知机学习算法来训练模型。

感知机是一种简单的非线性神经网络，是人工神经网络的基础。感知机仅包含输入层和输出层，其输入层可以包含多个单元，而输出层只有一个单元。感知机通过采用有监督学习来逐步增强模式分类的能力，达到学习的目的。

从本质上说，感知机与 MP 模型没有太大的区别，两者的结构相同（图 5.2），计算过程也相同，都能完成线性可分的二分类任务，也都无法解决线性不可分问题。但 MP 模型与感知机的不同之处在于：①MP 模型的权值 $w$ 和偏置 $b$ 都是人为设定的，没有“学习”的机制；而感知机引入了“学习”的概念，权值 $w$ 和偏置 $b$ 是通过学习得到的，并非人为设置的，在一定程度上模拟了人脑的“学习”功能，这也是两者最大的区别。②两者采用的激活函数不同，MP 模型采用阶跃函数作为激活函数，而感知机通常采用 sigmoid 函数作为激活函数。

sigmoid 的中文意思是“S 形状的”，所以 sigmoid 函数原本是指函数图像如 S 型的一类函数，包括逻辑（logistic）函数、双曲正切（tanh）函数等。后来，遵从大多数人的习惯，当提及 sigmoid 函数时，就专指 logistic 函数了，而其他形如 S 的函数则直接称呼其名称，如 tanh 函数。下面分别介绍 sigmoid（logistic）函数和 tanh 函数。

#### 1. sigmoid 函数

sigmoid 函数也称为 logistic 函数，具有平滑性、连续性、单调性和渐近性，而且是连续可导的。2012 年 ReLU 函数被重视之前，sigmoid 函数是最常用的非线性激活函数，其输出值为（0，1），可用于表示概率或输入的归一化。sigmoid 函数的数学表达式如公式（5.3）所示。

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

sigmoid 的函数图像如图 5.3 所示。

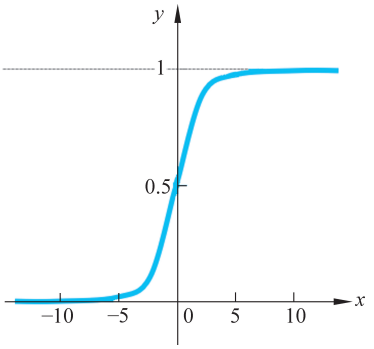



图 5.3　sigmoid 的函数图像

sigmoid 函数的求导公式如下：

$$\begin{aligned}\frac{\partial\sigma(x)}{\partial x}&=-\frac{1}{(1+\mathrm{e}^{-x})^2}\times\mathrm{e}^{-x}\times(-1)=\frac{\mathrm{e}^{-x}}{(1+\mathrm{e}^{-x})^2}\\&=\frac{1}{1+\mathrm{e}^{-x}}\times\left(1-\frac{1}{1+\mathrm{e}^{-x}}\right)=\sigma(x)(1-\sigma(x))\end{aligned} \tag{5.4}$$

sigmoid 函数的优点是平滑、易于求导，其导数可直接用函数的输出计算，简单高效。sigmoid 函数很好地解释了神经元在受到刺激的情况下是否被激活和向后传递的情景。当取值接近 0 时，几乎没有被激活；当取值接近 1 时，几乎完全被激活。

sigmoid 函数的缺点如下。

(1) 当输入的绝对值大于某个阈值时，会快速进入饱和状态(即函数值趋于 1 或 −1，不再有显著的变化，梯度趋于 0)，会出现梯度消失的情况，权重无法再更新，会导致算法收敛缓慢，甚至无法完成深层网络的训练。因此在一些现代的神经网络中，sigmoid 函数逐渐被 ReLU 激活函数取代。

(2) sigmoid 函数公式中有幂函数，计算耗时长，在反向传播误差梯度时，求导运算涉及除法。

(3) sigmoid 函数的输出恒大于 0，非零中心化，在多层神经网络中，可能会造成后面层神经元的输入发生偏置偏移，导致梯度下降变慢。

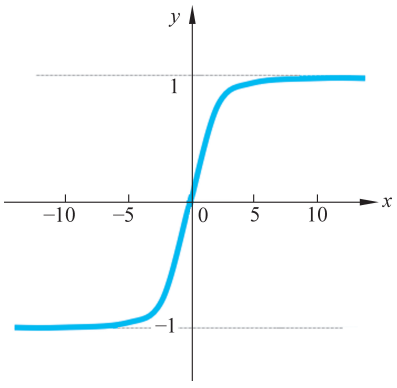



图 5.4 tanh 的函数图像

### 2. tanh 函数

tanh 函数是 sigmoid 函数的一个变形，称为双曲正切函数。tanh 函数值为(−1,1)，改进了 sigmoid 变化过于平缓的问题，而且其输出是零中心化的，解决了 sigmoid 函数的偏置偏移问题。

tanh 函数的数学表达式如公式(5.5)所示。

$$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = 2\sigma(2x) - 1 \tag{5.5}$$

tanh 的函数图像如图 5.4 所示。对比 tanh 和 sigmoid 的函数图像，tanh 函数可以看作是在纵轴方向上放大到 2 倍并向下平移的 sigmoid 函数。

tanh 函数的导数公式如下：

$$\frac{\partial y}{\partial x} = -\frac{(e^{x} + e^{-x})(e^{x} + e^{-x}) - (e^{x} - e^{-x})(e^{x} - e^{-x})}{e^{x} + e^{-x}} = 1 - y^{2} \tag{5.6}$$

tanh 函数的优点是：tanh 在线性区的梯度更大，能加快神经网络的收敛。但其缺点是：tanh 函数两端的梯度也趋于零，依旧存在梯度消失的问题，同时，幂运算也会导致计算耗时长。

## 5.2.4 多层神经网络结构

单个人工神经元的结构简单，功能有限。若想完成复杂的功能，就需要将许多人工神经元按照一定的拓扑结构相互连接在一起，相互传递信息，协调合作。组成神经网络的所有神经元是分层排列的，一个神经网络包括输入层(input layer)；隐藏层(hidden layer，也称为隐层、隐含层)和输出层(output layer)，每个网络只能有一个输入层和一个输出层，却可以有 0 个或多个隐藏层，每个隐藏层上可以有若干个神经元。只有输入层与输出层的神经元可以与外界相连，外界无法直接触及隐藏层，故而得名“隐藏层”。

包含至少一个隐藏层的神经网络称为**多层神经网络**(multi-layer neural network)。在包含神经元个数最多的一层，神经元的个数称为该神经网络的**宽度**。除输入层外，其他层的层数称为神经网络的**深度**，即等于隐藏层个数加 1(输出层)。感知机没有隐藏层，只有输入层和输出层，因此感知机的层数为 1，称为单层神经网络。

人工神经网络的行为并非各个神经元行为的简单相加，而是具有学习能力的、行为复杂

的非线性系统，既可以提取和表达样本的高维特征，又可以完成复杂的预测任务。

值得注意的是：多层神经网络的每个隐藏层后面都有一个非线性的激活函数。这里激活函数的作用比感知机中作为激活函数的阶跃函数的作用要大得多，因为激活函数是对所有输入信号的线性组合结果进行非线性变换，而且多层神经网络就有多个激活函数。所以，这些激活函数最主要的作用是向模型中加入非线性元素，用以解决非线性问题。一般在同一个网络中使用同一种激活函数。

根据神经元之间的连接范围，可以将多层人工神经网络分为**全连接神经网络**和**部分连接神经网络**。若每个神经元与其相邻层的所有神经元都相连，这种结构的网络称为**全连接神经网络**(fully connected neural network)，如图 5.5 所示。若每个神经元只与相邻层上的部分神经元相连，则是**部分连接神经网络**。

根据网络层之间的连接方式，又可以将多层人工神经网络分为**前馈神经网络**和**反馈神经网络**。

### 1. 前馈神经网络

前馈神经网络(feedforward neural network)是一种多层神经网络，其中每个神经元只与其相邻层上的神经元相连，接收前一层的输出，并输出给下一层，即第 $i$ 层神经元以第 $i-1$ 层神经元的输出作为输入，第 $i$ 层神经元的输出作为第 $i+1$ 层神经元的输入。同层的神经元之间没有连接。整个网络中的信息是单向传递的，即只能按一个方向从输入层到输出层的方向传播，没有反向的信息传播，可以用一个有向无环图表示。

若前馈神经网络采用全连接方式，则称为**前馈全连接神经网络**，如图 5.5 所示。网络最左边的是输入层，最右边的是输出层，输入层和输出层之间有 2 个隐藏层，该神经网络是一个 3 层前馈全连接神经网络。

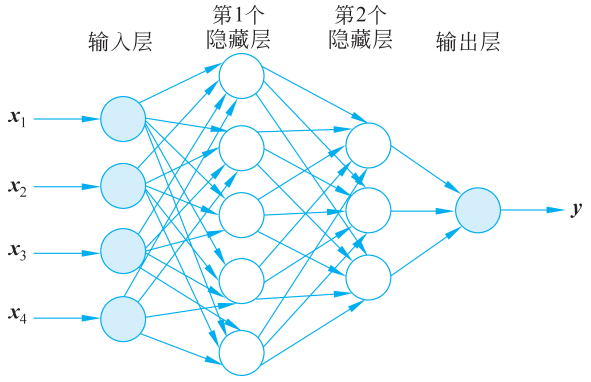



图 5.5　多层前馈全连接神经网络

在图 5.5 中，输入层是由 4 个元素组成的一维向量，第 1 个隐藏层有 5 个神经元，则输入层与第 1 个隐藏层之间有 $4\times5=20$ 个连接；第 2 个隐藏层有 3 个神经元，则第 1 个隐藏层与第 2 个隐藏层之间有 $5\times3=15$ 个连接；输出层是由 1 个元素组成的一维向量，则第 2 个隐藏层与输出层之间有 $3\times1=3$ 个连接。神经网络中的每条连接都有各自的权重参数，用于控制神经元输入信息的权重，这些参数将通过网络训练得到最终值。

图 5.5 中神经网络的深度为 3，宽度为 5，其中有 41 个待学习的参数($20+15+3=38$ 个连接，3 个偏置值)，还有 3 个超参数：隐藏层的个数以及两个隐藏层中包含的神经元个数，

这是人为设定的。

前馈神经网络可以看作一个函数，通过多次简单非线性函数变换，实现从输入空间到输出空间的复杂映射。这种网络结构简单，易于实现。

前馈全连接神经网络的缺点是：当网络很深时，参数量巨大，计算量大，训练耗时。

前馈神经网络包括 BP 神经网络和卷积神经网络，将分别在 5.3 节、5.4 节中介绍。

**2. 反馈神经网络**

反馈神经网络(feedback neural network)是一种反馈动力学系统。在这种网络中，第 $i$ 层的神经元将自身的输出信号作为输入信号反馈给第 $j(j < i)$ 层的神经元，即有些神经元不但可以接收其前一层上神经元的信息，还可以接收来自于其后面层上神经元的信息。神经元的连接可以形成有向循环。反馈神经网络中的信息既可以单向传递，也可以双向传递，且神经元具有记忆功能，在不同时刻具有不同的状态，能建立网络的内部状态，可展现动态的时间特性。本书只讲解前馈神经网络。

## 5.3 BP 神经网络及其学习算法

多层前馈神经网络的表达能力比单层感知机要强得多。显然，要训练多层前馈神经网络，单层感知机的学习算法是远远不够的，需要更强大的学习算法。迄今为止，最成功的多层神经网络学习算法就是反向传播(back-propagation，BP)算法。BP 算法能解决对参数逐一求偏导、计算效率低下的问题。迄今为止，学术界和产业界依然在用 BP 算法训练神经网络。

### 5.3.1 BP 神经网络的结构

由于前馈神经网络大多采用反向传播学习算法来进行训练模型，故被称为 BP 神经网络。BP 神经网络是一种多层前馈神经网络，其结构示意图如图 5.6 所示。第一层称为输入层，最后一层称为输出层，中间各层称为隐藏层。该 BP 神经网络的深度为 $m-1$，每个节点就是一个神经元，神经元之间带有箭头的连线表示信息传递的方向。

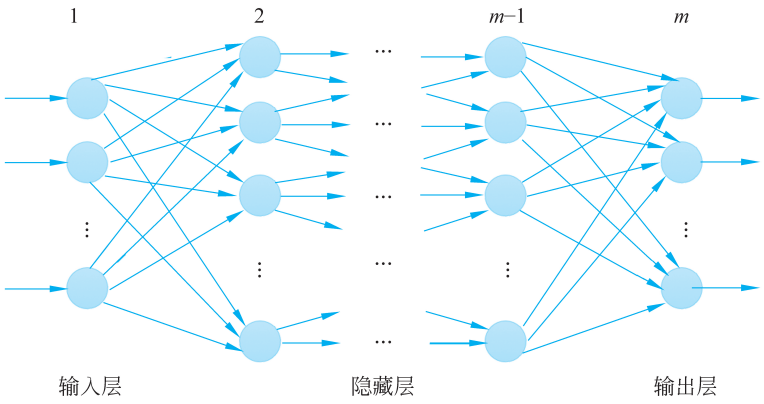



图 5.6 BP 神经网络结构示意图

图 5.7 是图 5.6 中某相邻两层网络各变量之间关系的示意图，假设神经网络中第 $k-1$

层中有 $p_{k-1}$ 个神经元，令 $y_j^{k-1}$ 表示第 $k-1$ 层中第 $j$ 个神经元的输出，$w_{ji}^k$ 表示第 $k-1$ 层的第 $j$ 个神经元与第 $k$ 层的第 $i$ 个神经元之间的连接权值，$\theta^k$ 表示第 $k$ 层的偏置项，$u_i^k$ 表示第 $k$ 层的第 $i$ 个神经元的净输入（即其所有输入信号的线性组合），则 $u_i^k$ 的计算表达式如公式（5.7）所示，为使公式表达整齐，可写成公式(5.8)的形式。

$$u_i^k=\sum_{j=1}^{p_{k-1}} w_{ji}^k\, y_j^{k-1}+\theta^k,\quad k=2,3,\cdots,m \tag{5.7}$$

令 $w_{0i}^k=\theta^k$，$y_0^{k-1}=1$，则有

$$u_i^k=\sum_{j=0}^{p_{k-1}} w_{ji}^k\, y_j^{k-1},\quad k=2,3,\cdots,m \tag{5.8}$$

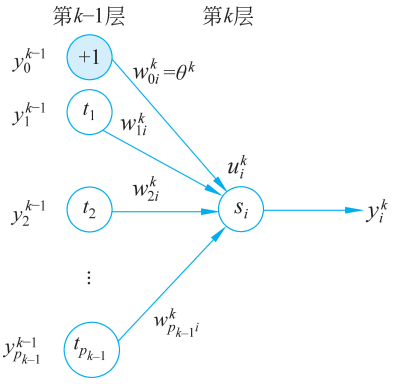



图 5.7　BP 网络中相邻两层的各变量之间关系

令 $f(\cdot)$表示激活函数，$y_i^k$ 表示第 $k$ 层的第 $i$ 个神经元的输出，它是该神经元的净输入经过激活函数映射得到的，如公式(5.9)所示。

$$y_i^k=f(u_i^k) \tag{5.9}$$

### 5.3.2　BP 学习算法

神经网络学习，也称为神经网络训练，是指利用训练数据集不断地修改神经网络的所有连接权值和偏置值，使神经网络的实际输出尽可能地逼近真实值(ground truth)。可见，BP学习算法是一种有监督学习。

在大多数情况下，并不知道如何调整神经网络的权重 $w$ 和偏置 $\theta$，才能使神经网络输出期望的结果。因此，需要采用损失函数来指导学习算法如何更新权重和偏置，以提高网络的性能。

一般地，假设有 $m-1$ 层(不算输入层)BP 神经网络 $F$，向 $F$ 输入数据 $\boldsymbol{X}=(x_1,x_2,\cdots,x_{p1})^{\mathrm{T}}$（$p_1$ 为输入层神经元的个数），则从输入层依次经过各个隐藏层节点可得到输出数据 $\boldsymbol{Y}=(y_1^m,y_2^m,\cdots,y_{pm}^m)^{\mathrm{T}}$（$p_m$ 为输出层神经元的个数）。由于神经元的激活函数是非线性函数（早期一般采用 sigmoid 函数，现在多采用 ReLU 函数），因此可以将 BP 神经网络看成是一个从输入到输出的非线性映射 $\boldsymbol{Y}=F(\boldsymbol{X})$。BP 神经网络具有很强的学习能力，由第 $k-1$ 层的第 $j$ 个神经元到第 $k$ 层的第 $i$ 个神经元的连接权值 $\{w_{ji}^k,k=2,3,\cdots,m;i=1,2,\cdots,p_k;j=1,2,\cdots,p_{k-1}\}$（$p_k$ 表示第 $k$ 层中神经元的个数）及第 $k$ 层的偏置$\{\theta^k,k=2,3,\cdots,m\}$ 为要学习的网络参数。

假设选择神经网络的实际输出与期望输出之间的误差平方和作为损失函数，其数学表达式为

$$E=\frac{1}{2}\sum_{j=1}^{p_m}(y_j-y_j^m)^2 \tag{5.10}$$

其中，$m$ 是输出层的层号，即输出层是第 $m$ 层；$p_m$ 是输出层中神经元的个数；$y_j^m$ 是输出层第 $j$ 个神经元的实际输出；$y_j$ 是输出层第 $j$ 个神经元的期望输出。BP 学习算法的目的是：使得目标函数（即损失函数）$E$ 达到极小值。损失函数的值越小，表示神经网的预测结果越

接近真实值。

BP 算法的学习过程由**正向传播**和**反向传播**两个阶段组成，算法的实现过程如下。

**第 1 步**：用随机值初始化神经网络 $F$ 的权重 $\boldsymbol{W}$ 和偏置 $\boldsymbol{\theta}$；令 $l=1$。

**第 2 步**：若 $l \leqslant N$（$N$ 为训练样本总数），向 $F$ 输入第 $l$ 个样本的 $p_1$ 维向量 $\boldsymbol{X}_l=(x_{l1},..,x_{lp_1})$，经过一次正向传播，得到预测结果 $\boldsymbol{Y}_l^m=F(X_l)$，$\boldsymbol{Y}_l^m$ 为一个 $p_m$ 维的向量 $(y_{l1}^m,..,y_{lp_m}^m)$；否则，算法停止。

**第 3 步**：已知第 $l$ 个样本的期望输出为 $\boldsymbol{Y}_l=(y_{l1},..,y_{lp_m})$，采用公式(5.10)计算输出层的损失函数 $E$ 的值。

**第 4 步**：若 $E$ 值小于设定的阈值，则 $l=l+1$，转向第 2 步；否则，进行反向传播，采用梯度下降法，依次计算每个权重和偏置的修正值，公式如下：

$$\Delta w_{ji}^k=-\eta\frac{\partial E}{\partial w_{ji}^k},\quad \Delta\theta^k=-\eta\frac{\partial E}{\partial\theta^k}\quad(\eta>0) \tag{5.11}$$

其中，$\eta$ 是学习率，一般小于 0.5。

**第 5 步**：采用公式(5.11)计算出的修正值，依次更新所有权重值和偏置值；

**第 6 步**：转向第 2 步。

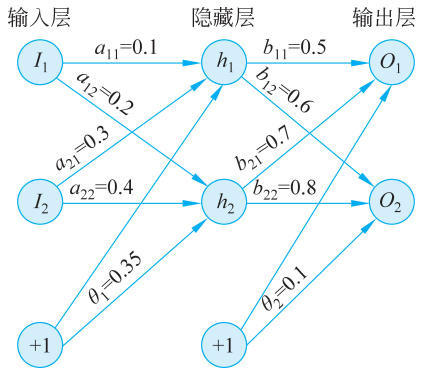



图 5.8 BP 神经网络示例

下面以图 5.8 中的神经网络为例，详细介绍 BP 算法的具体执行过程。

该网络包含输入层、隐藏层和输出层各一个。输入层有两个神经元 $I_1$ 和 $I_2$；隐藏层有两个神经元 $h_1$ 和 $h_2$；输出层有两个神经元 $O_1$ 和 $O_2$；每条有向边上的数值表示神经元之间的连接权重。为表达整齐，为每层的偏置项设置了一个虚拟的神经元“+1”。激活函数选用 sigmoid 函数，sigmoid 函数及其对 $x$ 的导数如公式(5.12)所示。

$$\begin{aligned}\sigma(x)&=\frac{1}{1+e^{-x}}\\ \sigma'(x)&=\sigma(x)(1-\sigma(x))\end{aligned} \tag{5.12}$$

假设输入样本为($x_1=0.05$，$x_2=0.1$)，期望输出值为($y_1=0.03$，$y_2=0.05$)。以此为例，BP 算法的具体实现过程如下。

### 1. 正向传播

在正向传播阶段中，输入样本($x_1=0.05$，$x_2=0.1$)从输入层经隐藏层逐层处理，传向输出层，每一层神经元的状态只影响下一层神经元的状态。

(1) **由输入层向隐藏层传播信息。**

令 $net_j$ 表示编号为 $j$ 的神经元的净输入，$out_j$ 表示编号为 $j$ 的神经元的输出，则 $out_{I_1}=x_1=0.05$，$out_{I_2}=x_2=0.1$。

① 针对隐藏层中第一个神经元 $h_1$，计算其净输入值，再计算其输出值，计算公式如下。

$$\begin{aligned}net_{h_1}&=out_{I_1}\times a_{11}+out_{I_2}\times a_{21}+\theta_1\\&=0.05\times0.1+0.1\times0.3+0.35=0.385\end{aligned}$$