第3章

人工神经网络

3.1 概述

3.1.1 人工神经网络简介

神经网络一般分为生物神经网络(Biological Neural Network)和人工神经网络(Artificial Neural Networks,ANN,在计算机领域中通常简称为“神经网络”)。生物神经网络指由生物大脑内的神经元组成的网络结构,能够产生生物的意识。人工神经网络是从信息处理角度建立起来的一种计算模型。通过模拟生物神经元结构来建立基本的信息处理单元——人工神经元,然后将大量的人工神经元相互连接成网络进行信息传递,最终产生结果输出,这样的网络结构被称为人工神经网络。它是人工智能的三大思想流派中联结主义流派(或称仿生学派)的基础。人工神经网络通常需要通过大量的数据来学习并获得人的知识(规律),才能输出(计算出)正确结果,所以,它属于机器学习算法中的一个分支。但由于它实现“学习”的方法与其他机器学习算法有显著的不同,而且它当前在各个应用领域取得了颠覆性的突破和成就,因此常把其他机器学习算法统称为经典机器学习算法或传统机器学习算法来与之区分。

通过前面章节对机器学习的介绍可以知道,机器学习算法是通过用程序处理数据来获得人类的知识。而根据高文院士在 CNCC 2016 上所做的大会特邀报告,人的知识可以从两个维度(是否可统计、是否可推理)来分成四类,如图 3.1 所示。对于可统计、可推理的部分知识,不管是用经典机器学习算法还是人工神经网络的方法,原则上都能找到答案;对那些可推理、不可统计的部分知识,可以用举一反三的办法,经典机器学习算法在这个领域取得了大量成果;可统计、不可推理的部分知识可以采用模糊识别的方法,当前的神经网络方法(主要是深度神经网络,或称深度学习)在此领域取得了大量成就,包括下围棋的 AlphaGo 算法、人脸识别、语言翻译等;不可统计、不可推理的部分知识则依赖人的顿悟,很

难使用具体方法处理。

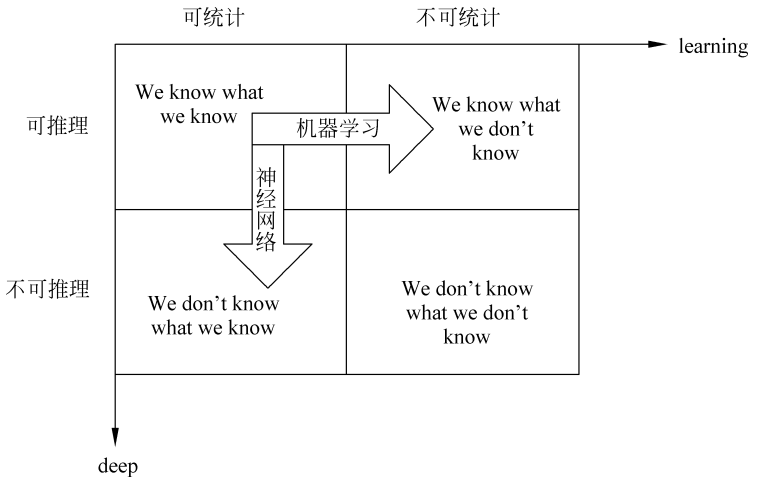


图 3.1 知识维度

从图 3.1 中也可以看出，神经网络算法属于统计学习算法，其通常不具备推理能力，而主要采用类似“记忆”的方法来从大量数据（称为训练数据）中统计出规律并通过参数存储起来，称为学习到了知识；在应用学习到了知识的神经网络的时候，就根据存储的参数去对输入数据进行数学运算，计算出答案，其实质类似于从存储的知识中找匹配的类似知识。由于这些代表神经网络特征的参数是由神经网络学习算法根据训练数据去自行修改、调整出来的，没有办法用确定的数学模型或统计模型去解释，因此业界广泛认为神经网络是一个黑箱模型，缺乏可解释性。但这并不妨碍其在数据处理、语音识别、自然语言处理、图像识别等领域取得的巨大成功，具有非常高的准确率，在一些领域的表现甚至超过了人类。

在编写计算机程序实现神经网络算法的时候，构建的神经网络结构通常如图 3.2 所示。整个神经网络按层次划分，每层包含一定数量的神经元实现信息处理，上一层神经元的处理结果作为下一层神经元的输入。输入层只是表示数据的输入，不计入神经网络层次数量中。因为使用者输入数据到输入层，并从最后一层获得输出结果，不直接使用中间其他层次，因

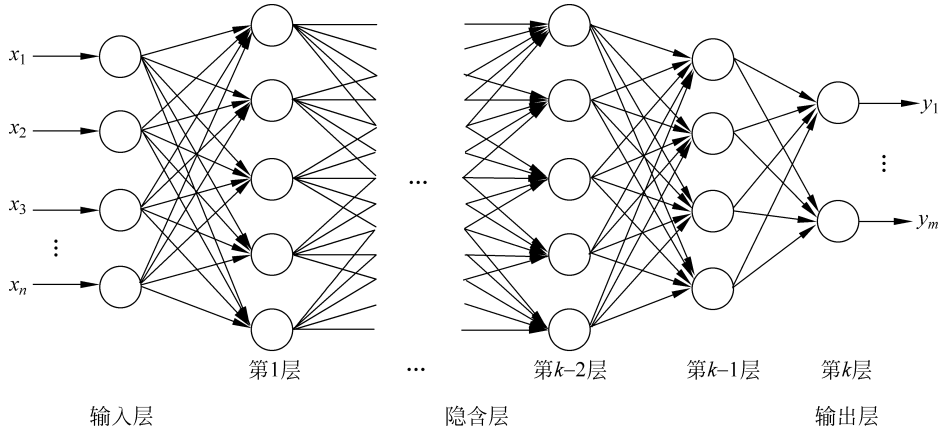


图 3.2 k 层的神经网络结构

此也把中间层次称为隐藏层或隐含层。如果构建的网络层数比较少(譬如只有几层),可以称为浅层学习;如果层数比较多,则称为深度学习(Deep Learning),但目前学术界对此并没有明确定义。

综上,深度学习与人工智能、机器学习、人工神经网络的关系如图 3.3 所示。机器学习是人工智能研究领域的一部分,是实现人工智能的算法。而人工神经网络算法是众多机器学习算法中的一个分支,也是当前取得极大成功的一个分支,它又可以分为浅层神经网络(实现浅层学习)和深层神经网络(实现深度学习)。由于深度学习算法处理出来的结果具有非常高的准确度,在众多领域超越现有的其他各种方法,达到或接近人的智能程度,从而掀起了当前人工智能的新高潮,是人工智能在各个领域得到广泛应用的核心技术和驱动力。

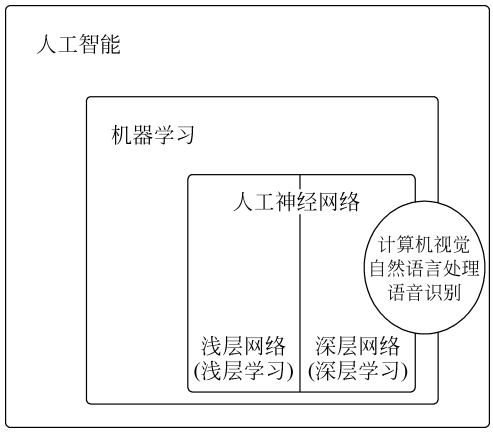


图 3.3　人工智能、机器学习与深度学习的关系

3.1.2　人工神经网络发展史

通常把人工神经网络的发展分成 4 个时期:启蒙时期(1890—1969 年)、低潮时期(1969—1982 年)、复兴时期(1982—1986 年)、新时期(1986 年至今)。

1. 启蒙时期(1890—1969 年)

1890 年,心理学家威廉·詹姆斯(William James)出版了世界上第一部详细论述人脑结构及功能的专著《心理学原理》,描述了人的神经系统的工作过程:一个神经元细胞接受到多个刺激(输入),这些刺激在细胞体内叠加(处理),在达到一定程度后产生冲动(输出),并将冲动传播到另一些神经细胞。

1943 年,神经生理学家沃伦·麦卡洛克(Warren S. McCulloch)和数理逻辑学家沃尔特·皮茨(Walter Pitts)在合作撰写的 *A logical calculus of the ideas immanent in nervous activity* 论文中提出人工神经网络的概念并给出了人工神经元的数学模型——M-P 模型,模仿人类神经元的工作原理来模拟函数计算。这被认为是世界上第一个人工神经网络模型,从而开创了人工神经网络研究的时代。

1949 年,心理学家赫布(Hebb)出版了 *The Organization of Behavior*(行为组织学),他在书中提出了突触连接强度可变的设想。这个假设认为学习过程最终发生在神经元之间的突触部位,突触的连接强度随着突触前后神经元的活动而变化,这个突触的连接强度的可变性是学习和记忆的基础。Hebb 法则为构造有学习功能的神经网络模型奠定了基础。

1958 年,就职于 Cornell 航空实验室的罗森布拉特(Frank Rosenblatt)发明了一种称为感知机(Perceptron)的人工神经网络模型,采用单层神经元的网络结构,能根据样本数据自主学习分类规则,从而实现对数据的线性二分类。感知机被称为人工神经网络的第一个实际应用,标志着神经网络进入了新的发展阶段。此后大量数学家、物理学家投入到神经网络研究中。

2. 低潮时期(1969—1982 年)

1969 年,符号主义学派的代表人物明斯基(M. Minsky)与麻省理工学院的佩珀特(Seymour Papert)合作撰写了《感知机:计算几何学》一书,指出了神经网络的两个关键问题:简单神经网络只能运用于线性问题的求解,无法解决异或问题等非线性可分问题;当时的计算机也没有足够的能力来处理大型神经网络所需要的高计算量。由于明斯基的学术地位和影响力,人们对感知机的学习能力产生了怀疑,进而使神经网络发展进入了"寒冬"。

1974 年,保罗·沃博斯(Paul Werbos)在哈佛大学攻读博士学位期间,就在其博士论文中首次提出了反向传播算法并构建了反馈神经网络,据此可以构建多层神经网络并能解决异或等问题,但当时没有引起重视。

3. 复兴时期(1982—1986 年)

1982 年,霍普菲尔德(John Hopfield)将物理学的动力学思想引入到神经网络中,提出了 Hopfield 神经网络。该算法采用全互联结构和反馈结构,所有神经元全部连接在一起,网络的输出端又连入输入端,使得网络不断循环、反复运行,直到达到稳定的平衡状态,从而产生稳定的输出值。该算法在著名的旅行商(TSP)问题这个 NP(Non-Deterministic Polynomial)完全问题的求解上获得了当时最好结果,引起了巨大的反响,使人们重新认识到人工神经网络的威力,从而推动神经网络的研究再次进入了蓬勃发展的时期。

1986 年,大卫·鲁梅哈特(David Rumelhart)、杰弗里·辛顿(Geoffrey E. Hinton)和罗纳德·威廉姆斯(Ronald Williams)在《自然》杂志上合作发表了一篇突破性的论文《通过反向传播算法实现表征学习》(*Learning representations by Back-propagating Errors*),清晰论证了"误差反向传播"(Back-Propagating Errors)算法是切实、可操作的训练多层神经网络的方法,彻底扭转了明斯基《感知机:计算几何学》一书带来的负面影响,多层神经网络的有效性终于再次得到学术界的普遍认可,从而将神经网络的研究推向了新的高潮。

4. 新时期(1986 年至今)

1987 年 6 月,首届国际神经网络学术会议在美国圣地亚哥召开,会上成立了国际神经网络学会(INNS)。

20 世纪 80 年代末,迈克尔·乔丹(Michael I. Jordan)和杰弗里·埃尔曼(Jeffrey Elman)提出了简单循环神经网络,拉开了循环神经网络(Recurrent Neural Network,RNN)研究的序幕。

1989 年,杨立昆(Yann LeCun) 等人使用 BP 算法训练深度神经网络来识别手写邮编,该神经网络被命名为 LeNet。LeNet 被认为是最早的卷积神经网络(Convolutional Neural Network,CNN)之一,在很大程度上推动了深度学习领域的发展。

1997 年,塞普·霍克赖特(Sepp Hochreiter)和于尔根·施米德胡贝(Jürgen Schmidhuber)提出了长短期记忆人工神经网络(Long-Short Term Memory,LSTM),用于解决一般 RNN 存在的长期依赖问题。

2006 年,杰弗里·辛顿提出了深度置信网络(DBN),用于建立一个观察数据和标签之间的联合分布。

2012 年,杰弗里·辛顿和他的学生 Alex 合作开发了 Alexnet 深度卷积网络,结构类似于 LeNet5,但是卷积层深度更大,参数总数达数千万。该算法在 ImageNet 大赛上把图像分

类错误率从25%以上降到了15%，以远超第二名的成绩夺冠，从此卷积神经网络乃至深度学习重新引起了学术界的广泛关注，呈现出爆发式的发展。当前，卷积神经网络在人脸识别、自动驾驶、安防等领域都得到了广泛的应用。

2014年，伊恩·古德费洛(Ian J. Goodfellow)等人提出了生成对抗网络(Generative Adversarial Network，GAN)，它是无监督学习的一种方法，通过让两个神经网络相互博弈的方式进行学习。

2015年，何恺明等人提出了残差网络(ResNet)，该网络易于优化，且能够通过很大的深度来提高准确率，在当年的ImageNet大规模视觉识别竞赛中获得了图像分类和物体识别组的优胜。其内部的残差块使用了跳跃连接，缓解了在深度神经网络中增加深度带来的梯度消失问题。

2017年11月，萨拉·萨伯(Sara Sabour)、尼古拉斯·弗罗斯特(Nicholas Frosst)和杰弗里·辛顿合作发表了一篇名为《胶囊间的动态路由》的论文，该论文介绍了一个在MNIST(著名的手写数字图像数据集)上达到最先进性能的胶囊网络架构，并且在MultiMNIST(一种不同数字重叠对的变体)上得到了比卷积神经网络更好的结果。

3.2　M-P模型

1943年，沃伦·麦卡洛克和沃尔特·皮茨提出了模仿生物神经元的结构和工作原理的数学模型，称为M-P模型。

3.2.1　生物神经元

生物神经元是一种神经细胞，主要包含多个树突、一个轴突(末端有多个神经末梢)、一个细胞体这三个部分，如图3.4所示。

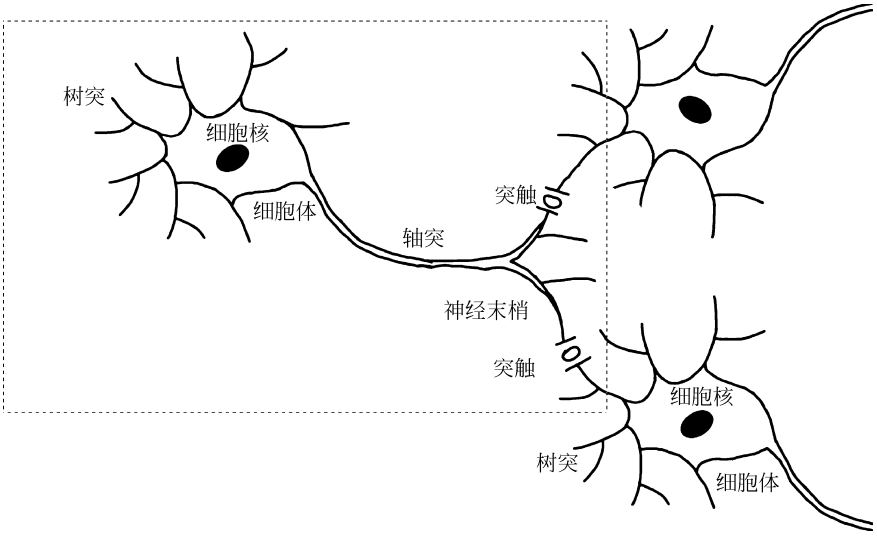


图3.4　生物神经元的结构

树突短而分支多，与其他神经元的神经末梢相连接，接收来自其他神经元传来的神经冲动（生物电），可以理解为信息输入端。

细胞体汇集传入的神经冲动，并产生新的神经冲动，可以理解为加工信息的部分。

轴突是细胞体突起的最长的外伸管状纤维，每个神经元只有一个轴突。它能够把神经元产生的神经冲动通过众多的神经末梢传递给其他神经元的树突，可以理解为信息输出端。

神经末梢与其他神经元的树突连接的位置存在突触。突触传递神经冲动的能力可以发生改变，从而使得一个神经元传出神经冲动时，不同神经元接收到的神经冲动强度不一样；即便是同一个神经元，在不同状态下树突接收到的神经冲动强度也可能是不同的。突触的这种传递能力的变化决定着两个神经元间的连接强度。

3.2.2 M-P 模型的结构

M-P 神经元模型就是对神经元的工作流程进行简单抽象和模拟，其结构如图 3.5 所示。

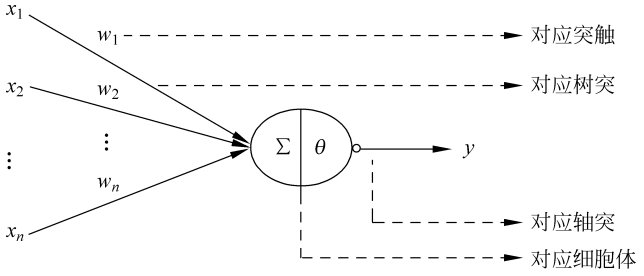


图 3.5 M-P 模型

神经元可以接收多个输入信号，x_i 表示从第 i 个神经元传来的信号强度值，信号经突触传递给树突，突触的连接强度（传递能力）用 w_i 表示，则树突传入的信号强度为 $w_i x_i$。在细胞体中，对各个树突传来的信号进行汇集和处理：汇集是对输入信号直接相加（即 $\sum_{i=1}^{n} w_i x_i$）；处理则是根据预先设定的阈值 θ，如果汇集后的信号强度大于该阈值就通过轴突产生冲动（输出 1），否则不产生冲动（输出 0）。

M-P 模型最终输出的描述如式(3.1)所示：

$$y = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right) \tag{3.1}$$

函数 f 是阶跃函数，如图 3.6 所示，当刺激强度 $\sum_{i=1}^{n} w_i x_i$ 大于该神经元的阈值 θ，则该神经元表现为兴奋状态，输出 $y=1$；反之则表现出抑制状态，输出 $y=0$；可用式(3.2)表示。

$$f(x) = \begin{cases} 1, & \text{当 } x \geqslant 0 \\ 0, & \text{当 } x < 0 \end{cases} \tag{3.2}$$

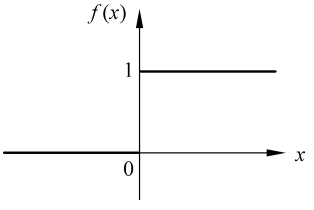


图 3.6 阶跃函数

在 M-P 模型中，函数 f 也被称作激活函数；各个 w_i 被称为权值参数，θ 被称为偏置（即上面的阈值）。由于权值参

数 w_i 和偏置参数 θ 都需要人为设定，因此 M-P 模型并没有学习能力。

3.2.3　M-P 模型实现与门电路逻辑运算的应用案例

采用 M-P 模型可以实现一些门电路的逻辑运算功能。与门电路是有两个输入和一个输出的门电路，其输入信号 x_1、x_2 和输出信号 y 的对应关系如表 3.1 所示，这种表称为真值表。与门电路的逻辑运算规则是：仅在两个输入均为 1 时输出 1，其他情况则输出 0。

表 3.1　与门真值表

x_1	x_2	y	x_1	x_2	y
0	0	0	1	0	0
0	1	0	1	1	1

建立一个有两个数据输入端的 M-P 模型，其权值用 w_1、w_2 表示，偏置(阈值)用 θ 表示。当设置 $w_1=0.5$，$w_2=0.5$，$\theta=0.8$ 时，就可以用来实现与门电路的逻辑运算，模型如图 3.7 所示。

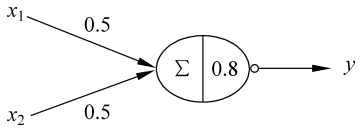


图 3.7　实现与门电路逻辑运算的 M-P 模型

根据 M-P 模型的计算公式，有以下几种情况。

(1) x_1、x_2 分别为 0、0 时，M-P 模型输出值 0，计算如下：

$$y=f(w_1x_1+w_2x_2-\theta)=f(0.5\times0+0.5\times0-0.8)=f(-0.8)=0$$

(2) x_1、x_2 分别为 0、1 时，M-P 模型输出值 0，计算如下：

$$y=f(w_1x_1+w_2x_2-\theta)=f(0.5\times0+0.5\times1-0.8)=f(-0.3)=0;$$

(3) x_1、x_2 分别为 1、0 时，M-P 模型输出值 0，计算如下：

$$y=f(w_1x_1+w_2x_2-\theta)=f(0.5\times1+0.5\times0-0.8)=f(-0.3)=0;$$

(4) x_1、x_2 分别为 1、1 时，M-P 模型输出值 1，计算如下：

$$y=f(w_1x_1+w_2x_2-\theta)=f(0.5\times1+0.5\times1-0.8)=f(0.2)=1。$$

综上，对于与门电路的各种输入，M-P 模型都能给出正确的输出结果，因此能实现与门电路的逻辑运算。

3.3　感知机

在 M-P 模型中，参数需要人为设置。而在实际应用中，需要有办法能自动选择合适的参数，才具有实用价值。感知机中引入了学习的概念，先预设参数，然后通过已知结果的数据来获得模型输出误差，引导权值参数进行调整来不断减小误差，就可以找到最合适的参数，从而达到学习的目的。简而言之，“学习”就是通过已知数据(称为训练数据)来获得网络中的参数。

3.3.1 感知机模型

感知机是一种人工神经网络，其结构与 M-P 模型基本一致，如图 3.8 所示，包含输入数据、神经元处理层、输出数据三个部分。由于处理数据的神经元只有一层，因此也称为单层感知机。为了便于计算，把偏置进行了等价替换：增加一个值为 1 的固定输入，对其设置权值 b。因为 b 的值会被自动调整，不失一般性，在神经元进行求和运算时可以等价地用加 $1b$ 替换原 M-P 模型公式中的减去阈值 θ，即输出计算公式变为式(3.3)，这里的激活函数 f 仍然为阶跃函数。

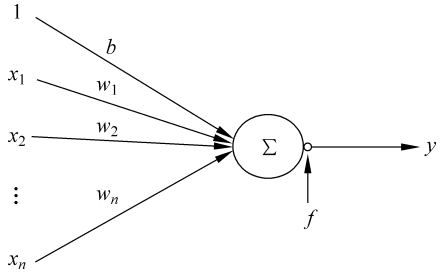


图 3.8 感知机模型

$$y=f\left(\sum_{i=1}^{n}w_i x_i+b\right) \tag{3.3}$$

3.3.2 感知机工作过程

感知机的计算过程和 M-P 模型一样。为了便于描述，这里以用只有两个输入的感知机实现与门电路的逻辑运算为例来介绍其工作过程，其模型结构如图 3.9 所示，计算过程如式(3.4)所示。

$$y=f(w_1x_1+w_2x_2+b) \tag{3.4}$$

当 $w_1=0.5$，$w_2=0.5$，$b=-0.8$ 时，该模型同样能实现与门电路的逻辑计算(见图 3.10)，计算过程可以参照 M-P 模型的计算案例。感知机把这种能力推广，用来进行分类，并具有几何上的意义。与门电路的逻辑计算可以看作是把平面中 4 个点(1,1)、(1,0)、(0,1)、(0,0)分为两类，(1,1)属于一类，其他三个点属于另一类，分别通过输出值 $y=1$ 和 $y=0$ 来表示。感知机对输入数据的处理，可以理解为平面中建立一条划分这两个类别的直线 $w_1x_1+w_2x_2+b=0$(这里为直线 $0.5x_1+0.5x_2-0.8=0$)，将两类不同输出分开。阶跃函数的输出就表示输入的点位于直线的上部分还是下部分，因此，如果输入的数据是(1.5,1.1)，也被分为 $y=1$ 类。

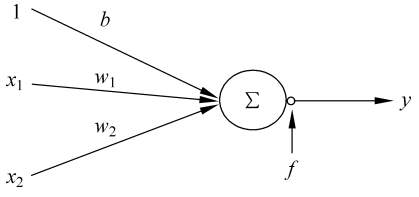


图 3.9 实现与门电路逻辑运算的感知机模型

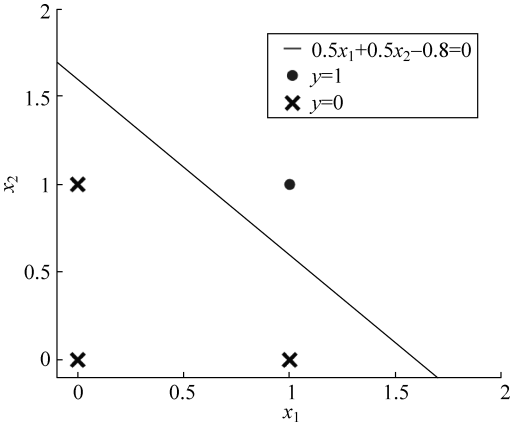


图 3.10 感知机解决与门问题

现在，需要做的事情就是如何通过数据来学习到参数 w_1、w_2、b 的值。

3.3.3　感知机的学习过程

感知机最初并不知道正确的权值参数，就设置为随机值；然后把已经知道结果的数据（称为训练数据或样本数据）逐条送进去进行计算；由于最初的权值参数是随机设置的，会产生很多错误的输出，通过误差（实际值与输出值之间的差）来调整权值参数，使修改后重新计算的结果误差减小；最终，当输入的每个数据都能计算出正确的结果，这时的感知机就已经正确学习到了所有参数。

而这个过程中关键的问题就是：如何调整权值参数（即权值更新）才能确保误差减小？下面给出感知机学习过程和权值更新规则。

第一步，随机初始化权值 $W(w_0,w_1,w_2,\cdots,w_n)$，为了描述的统一，用 w_0 代替 b。

第二步，输入一个样本 $(1,x_1,x_2,\ldots,x_n)$ 和对应的期望结果 y。其中 1 与 w_0 相乘表示偏置值，传入神经元。二分类中，一般用 $y=1$ 表示一类，用 $y=0$ 表示另一类（也有的用 -1 表示另一类别，都不影响计算）。

第三步，根据式(3.3)得感知机输出结果为：$y_{\text{out}}=f\left(\sum_{i=0}^{n}w_ix_i\right)$。

第四步，若该点被分类错误，则存在误差（$\varepsilon=y-y_{\text{out}}\neq 0$），以误差为基础对每个权值 w_i（$0\leqslant i\leqslant n$）按以下规则进行调整（称为学习规则）：

$$\Delta w_i=\eta(y-y_{\text{out}})x_i$$

$$w_i\leftarrow w_i+\Delta w_i$$

这里的 η 称为学习率，是一个人为设置的常量，一般为 0～1，用来控制每次权重的调整力度。

第五步，如果所有的样本分类的输出均正确即成功分类，则训练过程结束。只要有任何一个样本输出错误，那么都将导致权值调整，并且再次逐个输入所有样本进行训练。

训练过程的关键就是在于第四步的权重调整规则。这样的调整为什么确保能让错误的分类变正确，使分类错误的点越来越少？下面以与门电路的逻辑计算来详细分析。

该案例的训练数据集包含 4 个数据：(0,0)→0，(0,1)→0，(1,0)→0，(1,1)→1。

(1) 首先随机化权值，这里假设 (w_0,w_1,w_2) 为 $(-3,-2,4)$，学习率 η 设置为 0.6，如表 3.2 中初始行所示。

(2) 依次把样本按顺序送入感知机进行计算。

① 首先输入样本(0,0)，计算输出：

$$y_{\text{out}}=f\left(\sum_{i=0}^{n}w_ix_i\right)=f(w_0+w_1x_1+w_2x_2)=f(-3+(-2)\times 0+4\times 0)$$

$$=f(-3)=0$$

输出正确，所以权值不变化，如表 3.2 第 1 行数据所示。

② 继续输入样本(0,1),计算输出:

$$y_{\text{out}}=f\left(\sum_{i=0}^{n}w_i x_i\right)=f(w_0+w_1x_1+w_2x_2)$$
$$=f(-3+(-2)\times 0+4\times 1)=f(1)=1$$

输出结果与期望输出不同,误差为 $\varepsilon=y-y_{\text{out}}=0-1=-1$,按学习规则更新权值如下:

$$w_0=w_0+\Delta w_0=w_0+\eta\varepsilon x_0=-3+0.6\times-1\times 1=-3.6$$
$$w_1=w_1+\eta\varepsilon x_1=-2+0.6\times-1\times 0=-2$$
$$w_2=w_2+\eta\varepsilon x_2=4+0.6\times-1\times 1=3.4$$

如表 3.2 中第 2 行所示。对此调整进行如下分析:

图 3.11 显示感知机初始状态的分割直线,(0,1)被错误地分在了直线上方。从本次计算上看,误差=期望结果-输出结果=0-1=-1,为负值,说明当前该点的输出值过大,需要调整公式 $w_0+w_1x_1+w_2x_2$ 中的权值来减小输出结果。而学习规则正好是让各个权值均减小:权值 w_0 从-3 到 -3.6、w_1 从-2 到-2、w_2 从 4 到 3.4,能够让更新后的输出变小。从图 3.12 中可以看到,在调整过后,直线整体向上移动,(0,1)点被分割在直线下方,这样经过激活函数后就可以得到正确的输出,说明参数调整的方向正确,学习规则有效。

(3) 继续输入样本(1,0),计算输出:

$$y_{\text{out}}=f\left(\sum_{i=0}^{n}w_i x_i\right)=f(w_0+w_1x_1+w_2x_2)$$
$$=f(-3.6+(-2)\times 1+3.4\times 0)=f(-5.6)=0$$

输出正确,所以权值不变化,如表 3.2 第 3 行数据所示。

(4) 继续输入样本(1,1),计算输出:

$$y_{\text{out}}=f\left(\sum_{i=0}^{n}w_i x_i\right)=f(w_0+w_1x_1+w_2x_2)$$
$$=f(-3.6+(-2)\times 1+3.4\times 1)=f(-2.2)=0$$

结果与预期不符,存在误差 $\varepsilon=y-y_{\text{out}}=1-0=1$,按规则更新权值:$w_0=w_0+\Delta w_0=w_0+\eta\varepsilon x_0=-3.6+0.6\times 1\times 1=-3$,$w_1=w_1+\eta\varepsilon x_1=-2+0.6\times 1\times 1=-1.4$,$w_2=w_2+\eta\varepsilon x_2=3.4+0.6\times 1\times 1=4$,如表 3.2 第 4 行所示。

如图 3.12 所示,在用(0,1)训练网络之后,图像进行了调整,调整后只有(1,1)被错误地分类在直线下方。输入(1,1)进行网络训练时,输出的误差为 1-0=1,表示这个点计算的和太小,需要调整公式 $w_0+w_1x_1+w_2x_2$ 中的权值来增大输出结果。学习规则正好这样调整,让权值 w_0 从-3.6 到-3、w_1 从-2 到-1.4、w_2 从 3.4 到 4,从而让输出增加。

如图 3.13 所示,尽管(1,1)仍然处于直线下方,而且本次调整让点(0,1)的分类从正确变为错误,但直线在向正确的方向移动,并有一定旋转,可以看出继续按这个趋势调整,将很快能正确分类所有数据,因此是有效的(三个权值参数在不同时候调整的大小不同,而且各自变化的比例不同,表现为分类直线的旋转和移动)。

同理,循环带入所有数据,直到每个样本的计算输出都与预设的一样,就结束学习。每次权值调整的情况见表 3.2,图 3.14 至图 3.21 中画出了权值调整之后的分类直线变化。

表 3.2　参数调整的详细过程

调整参数	样本输入	样本预设类别值 y	网络输出 y_{out}	误差 ε	调整后的 w_0	调整后的 w_1	调整后的 w_2
初始					−3	−2	4

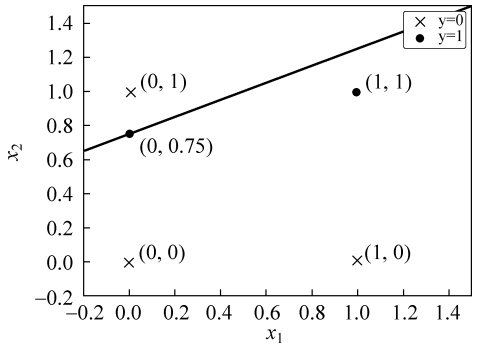


图 3.11　初始状态的直线方程为 $-2x_1+4x_2-3=0$
（请注意坐标轴以(−0.2,−0.2)为原点，直线公式为 $w_0+w_1x_1+w_2x_2=0$）

1	(0,0)	0	0	0	−3	−2	4
2	(0,1)	1	0	−1	**−3.6**	**−2**	**3.4**
3	(1,0)	0	0	0	−3.6	−2	3.4
4	(1,1)	0	1	1	**−3**	**−1.4**	**4**

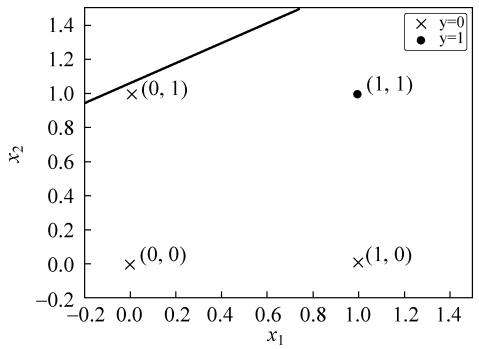


图 3.12　第 2 步代入调整后的直线方程为 $-2x_1+3.4x_2-3.6=0$

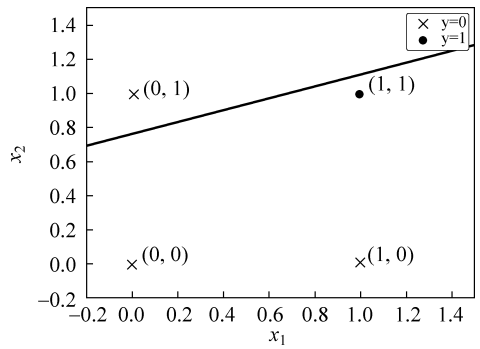


图 3.13　第 4 步代入调整后的直线方程为 $-1.4x_1+4x_2-3=0$

5	(0,0)	0	0	0	−3	−1.4	4
6	(0,1)	1	0	−1	**−3.6**	**−1.4**	**3.4**
7	(1,0)	0	0	0	−3.6	−1.4	3.4
8	(1,1)	0	1	1	**−3**	**−0.8**	**4**

续表

调整参数	样本输入	样本预设类别值 y	网络输出 y_{out}	误差 ϵ	调整后的 w_0	调整后的 w_1	调整后的 w_2
9	(0,0)	0	0	0	−3	−0.8	4
10	(0,1)	1	0	−1	**−3.6**	**−0.8**	**3.4**
11	(1,0)	0	0	0	−3.6	−0.8	3.4
12	(1,1)	0	1	1	**−3**	**−0.2**	**4**
13	(0,0)	0	0	0	−3	−0.2	4
14	(0,1)	1	0	−1	**−3.6**	**−0.2**	**3.4**
15	(1,0)	0	0	0	−3.6	−0.2	3.4
16	(1,1)	0	1	1	**−3**	**0.4**	**4**

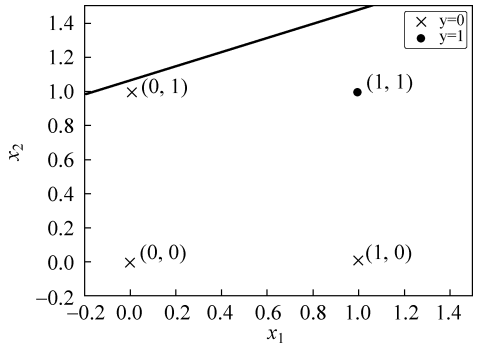


图 3.14 第 6 步代入调整后的直线方程为 $-1.4x_1+3.4x_2-3.6=0$

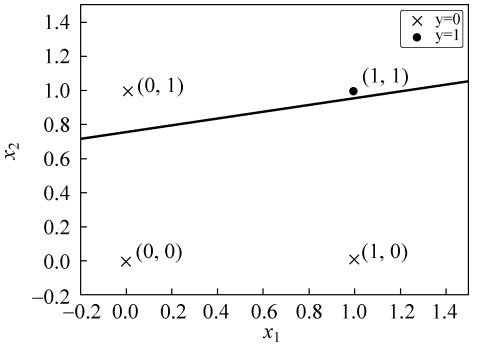


图 3.15 第 8 步代入调整后的直线方程为 $-0.8x_1+4x_2-3=0$

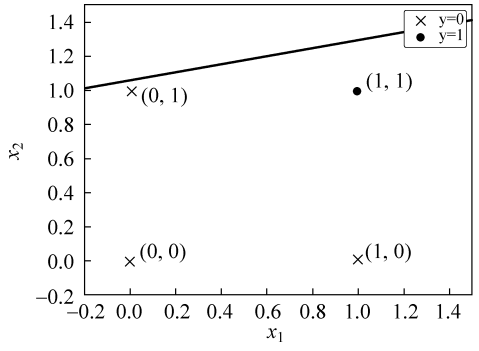


图 3.16 第 10 步代入调整后的直线方程为 $-0.8x_1+3.4x_2-3.6=0$

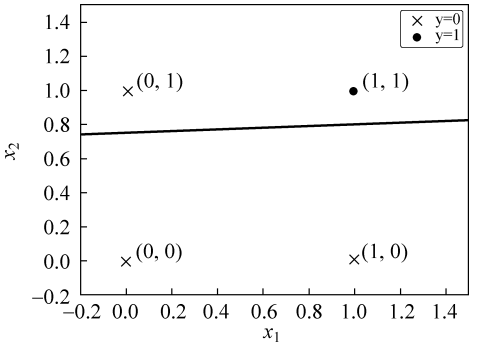


图 3.17 第 12 步代入调整后的直线方程为 $-0.2x_1+4x_2-3=0$

续表

调整参数	样本输入	样本预设类别值 y	网络输出 y_{out}	误差 ε	调整后的 w_0	调整后的 w_1	调整后的 w_2

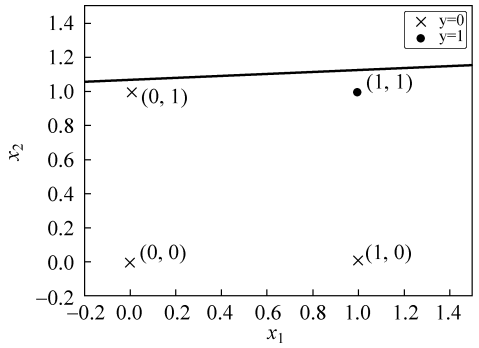


图 3.18　第 14 步代入调整后的直线方程为 $-0.2x_1+3.4x_2-3.6=0$

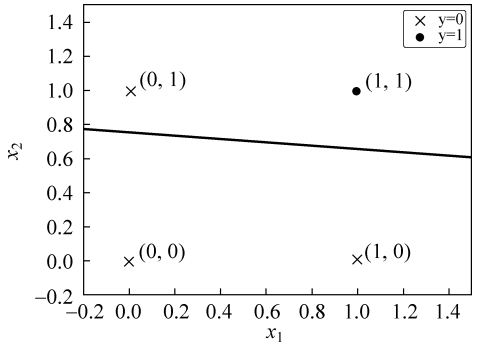


图 3.19　第 16 步代入调整后的直线方程为 $0.4x_1+4x_2-3=0$

17	(0,0)	0	0	0	−3	0.4	4
18	(0,1)	1	0	−1	**−3.6**	**0.4**	**3.4**
19	(1,0)	0	0	0	−3.6	0.4	3.4
20	(1,1)	1	1	0	−3.6	0.4	3.4

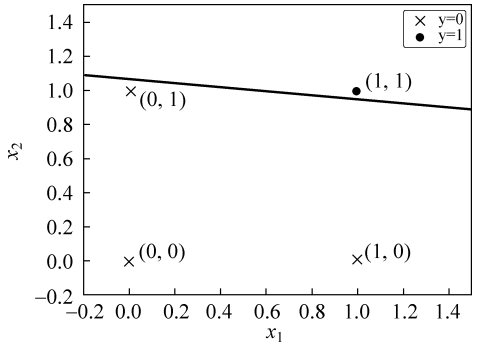


图 3.20　第 18 步代入调整后的直线方程为 $0.4x_1+3.4x_2-3.6=0$

21	(0,0)	0	0	0	−3.6	0.4	3.4
22	(0,1)	0	0	0	−3.6	0.4	3.4
23	(1,0)	0	0	0	−3.6	0.4	3.4
24	(1,1)	1	1	0	−3.6	0.4	3.4

续表

调整参数	样本输入	样本预设类别值 y	网络输出 y_{out}	误差 ε	调整后的 w_0	调整后的 w_1	调整后的 w_2

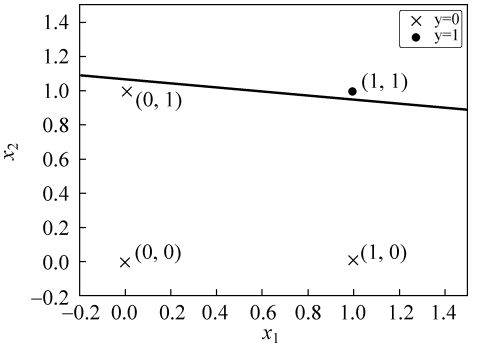


图 3.21 调整完毕的直线方程为 $0.4x_1+3.4x_2-3.6=0$

从上面的过程可知，感知机学习过程中，权值调整的方向是以让误差减小为依据的，因此能够有效地学习到合理的权值参数，而且还具有如下一些特征。

(1) 对某个样本错误进行权值参数调整，可能引起其他原本分类正确的点出错，例如在第 4 步中，代入(1,1)点输出错误，调整后(0,1)被错误分类在直线上方。所以每次调整之后，要把所有样本都要重新进行依次计算。

(2) 学习率有着重要作用。学习率大，每次调整力度大，可能让分割直线调整过头，引起其他点错误，其他点的错误又让直线调回来，从而引起分割直线来回振荡调整。学习率小，每次调整力度小，可能要反复训练多轮，才能把直线调整到合适位置，速度慢。合适的学习率对于训练来说非常重要。

(3) 一些点误差是 1，一些点误差是 -1，会让分隔直线来回移动。为了减少这种频繁调整，可以采用不是每个点计算出误差都调整权值，而是把所有点都计算一遍，把所有点的误差求和，然后再对权值进行一次调整的方法，实现每次调整让总体误差最小。

(4) 能线性分割这些点的直线非常多，根据初始参数和学习率，最终得到的权值参数结果不唯一。

(5) 如果这些点本身不是用一条直线可以分割的(非线性可分)，即没有一条直线能把这些点正确分开，那么这个学习过程无法终止。因此，它无法解决线性不可分问题。

3.3.4 感知机分类案例

对于输入数据是二维的情况，感知机的几何意义是找一条直线，将表示输入数据的平面中的点分开；如果输入数据是三维的，则相当于找一个平面，在三维空间中将它们分成两类；对于维度更高的数据，就是寻找对应的超平面。该平面和超平面方程都是由权值参数来确定。

接下来介绍一个区分苹果和香蕉的例子，更好地认识感知机的分类作用。首先用三个特征来对这两种水果进行描述，x_1 为颜色，x_2 为形状，x_3 为水分含量；其中水果颜色越红

则 x_1 数值越大、最大为 1，越黄则数值越小、最小为－1；当水果的形状越接近圆形时，x_2 数值越大、最大为 1，越接近长条形数值越小、最小为－1；含水分越多 x_3 数值越大、最大为 1，反之则数值越小、最小为－1；最终，某一苹果和某一香蕉的特征值如表 3.3 所示，即越红、越圆、含水越多的越可能是苹果(用类别 1 表示)，反之更可能是香蕉(用类别 0 表示)。

表 3.3　苹果、香蕉的特征表

特 征 参 数	苹果典型特征值 (输出用 1 表示苹果)	香蕉典型特征值 (输出用 0 表示香蕉)
x_1(颜色)：1 到－1 之间的值，表示由红到黄的程度	1	－1
x_2(形状)：1 到－1 之间的值，表示形状圆到长的程度	1	－1
x_3(含水量)：1 到－1 之间的值，表示含水程度	1	－1

可以建立一个简单的感知机模型 $y=f(w_1x_1+w_2x_2+w_3x_3)$，即存在三个输入和一个输出(为简单说明，这里设置偏置 b 为 0，因此该模型中需要训练的权值参数为 3 个：w_1、w_2、w_3，此处使用简单阶跃函数为感知机的激活函数)。

通过一定训练，得到网络的权值分别为 $w_1=1$，$w_2=1$，$w_3=1$(这个数据不唯一，可以通过训练得到)。

当输入 x_1、x_2、x_3 分别为 1，1，1 时，有

$$y=f(w_1x_1+w_2x_2+w_3x_3+b)=f(1\times1+1\times1+1\times1)=f(3)=1$$

则感知机输出 1；当输入 x_1、x_2、x_3 分别为－1，－1，－1 时，有

$$\begin{aligned}y&=f(w_1x_1+w_2x_2+w_3x_3+b)\\&=f(1-1+1\times-1+1\times-1)=f(-3)=0\end{aligned}$$

则感知机输出 0。

这样就实现了对样本的分类。在三维图 3.22 中可以清楚地看出这两点被一个平面分隔开，即感知机明确地区分开了苹果和香蕉。通过颜色、形状、含水量的特征，样本被分类为苹果或香蕉。

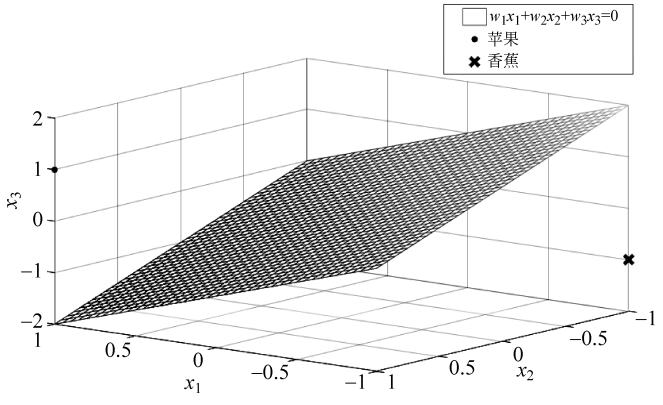


图 3.22　具有三个特征的数据的分类情况

由于香蕉和苹果的本质不同，相应的特征值会有所差距，相同品种的数据会趋于聚集，而不同品种的数据差距会相对大些，最终通过这些特征可以实现两者的分类。当用更多的苹果和香蕉的实际数据来训练感知机时，随着不断学习，这个平面将发生一定的变化，分类也会更加精确，如图 3.23 所示。

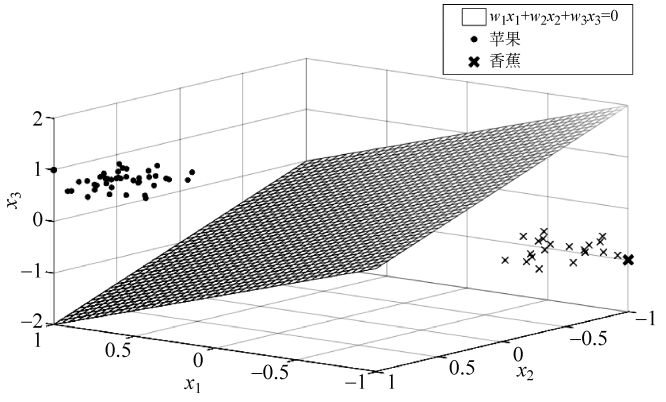


图 3.23　更多数据训练后的分类情况

3.3.5　多层感知机

由感知机的几何意义可以得知，单层感知机通过超平面来进行分类，无法解决线性不可分问题。这就是明斯基的质疑，单层感知机连异或问题都无法解决，从而让人们对感知机的学习能力产生了怀疑，造成了人工神经网络领域发展的长年停滞与低潮。如图 3.24 所示，无论直线怎么变动，也无法分割两种类型。

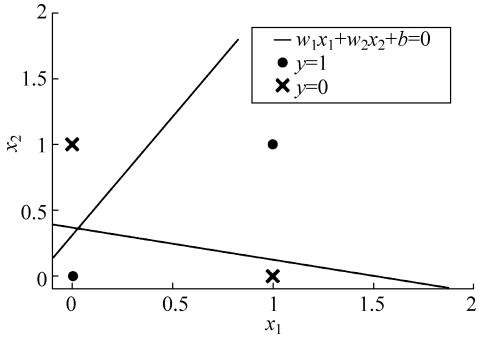


图 3.24　感知机无法进行分类的情况

随着研究的进行人们发现，在输入层与输出层之间增加隐含层，构成一种多层神经网络结构，这样的结构就可以解决非线性分类的问题，增强感知机的分类能力，这就是多层感知机（Multilayer Perceptrons，MLP）。如图 3.25 所示是一个 MLP 结构，包含了多个隐含层。

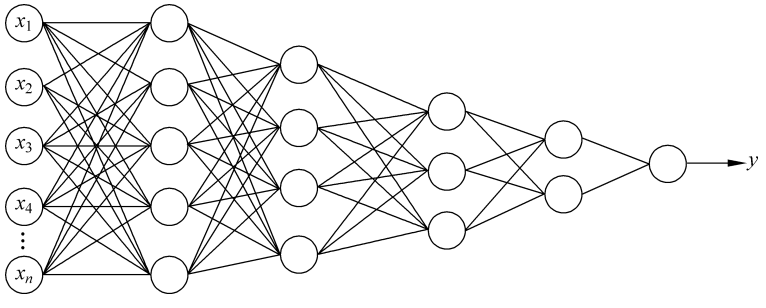


图 3.25　多层感知机结构

用一个如图 3.26 所示的两层感知机(输入层不算入神经网络层次),就可实现异或门电路的逻辑运算,其真值表如表 3.4 所示。

表 3.4　异或逻辑运算真值表

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

两层感知机的结构如图 3.26 所示,设置权值参数 $w_{11}=1, w_{21}=1, w_{12}=-1, w_{22}=-1, w_3=1, w_4=1, b_1=-0.5, b_2=1.5, b_3=-1.5$,激活函数 $f(x)$为阶跃函数。

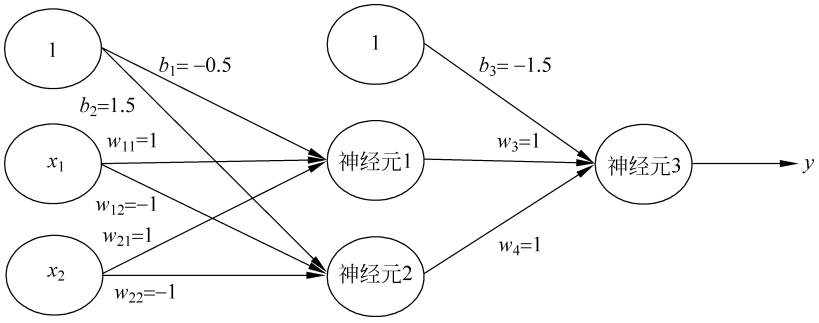


图 3.26　解决异或问题的两层感知机

将样本$(x_1, x_2)=(0,0)$输入模型,输出 $y=0$,计算过程如下:

神经元 1 输出 $s_1=f(w_{11}x_1+w_{21}x_2+b_1)=f(1\times0+1\times0-0.5)=0$

神经元 2 输出 $s_2=f(w_{12}x_1+w_{22}x_2+b_2)=f(-1\times0+(-1)\times0+1.5)=1$

神经元 3 输出 $y=f(w_3s_1+w_4s_2+b_3)=f(1\times0+1\times1-1.5)=0$

将样本$(x_1, x_2)=(0,1)$输入模型,输出 $y=1$,计算过程如下:

$$s_1=f(w_{11}x_1+w_{21}x_2+b_1)=f(1\times0+1\times1-0.5)=1$$

$$s_2=f(w_{12}x_1+w_{22}x_2+b_2)=f(-1\times0+(-1)\times1+1.5)=1$$

$$y=f(w_3s_1+w_4s_2+b_3)=f(1\times1+1\times1-1.5)=1$$

将样本$(x_1, x_2)=(1,0)$输入模型,输出 $y=1$,计算过程如下:

$$s_1=f(w_{11}x_1+w_{21}x_2+b_1)=f(1\times1+1\times0-0.5)=1$$

$$s_2=f(w_{12}x_1+w_{22}x_2+b_2)=f(-1\times1+(-1)\times0+1.5)=1$$

$$y=f(w_3s_1+w_4s_2+b_3)=f(1\times1+1\times1-1.5)=1$$

将样本$(x_1, x_2)=(1,1)$输入模型,输出 $y=1$,计算过程如下:

$$s_1=f(w_{11}x_1+w_{21}x_2+b_1)=f(1\times1+1\times1-0.5)=1$$

$$s_2=f(w_{12}x_1+w_{22}x_2+b_2)=f(-1\times1+(-1)\times1+1.5)=0$$

$$y=f(w_3s_1+w_4s_2+b_3)=f(1\times1+1\times0-1.5)=0$$

由上面计算可知,这样一个两层感知机的运算结果与异或门电路输出一致,实现了异或逻辑计算功能。但是感知机只给出了最后一层神经元权值的训练方法,而其他层的参数则只能人为设置。

3.4 多层神经网络

单层感知机能够根据已知数据来学习参数，在线性可分的问题领域具有很好的效果，但它不能处理线性不可分问题。而多层感知机只能训练最后一层参数，实用性有限。反向传播算法(BP 算法)的提出，使得多层神经网络的训练变得简单可行，证明了多层神经网络具有很强的学习能力。从而将神经网络的研究推向了新的高潮。

多层神经网络(Multilayer Neural Network，也称为多层感知机)的计算过程和感知机类似，通过输入数据来前向计算各个神经元的输出，并传递到下一层作为输入，最终得出结果。其核心在于采用反向传播算法(Back Propagation Algorithm，简称 BP 算法)来训练各层神经元的权值参数(即学习)。反向传播算法的关键是使用了一个重要的数学概念——梯度(Gradient)，而梯度又是由偏导数构成的，下面对这些概念进行介绍。

3.4.1 梯度

1. 导数概念

数学上，对于一元函数 $y=f(x)$来说，函数在某点的导数表示函数值在这一点附近的变化率。其计算方法为：对于某个点 x_0，当 x_0 增加 Δx 时，相应地函数值 y 也会发生变化，增量为 $\Delta y=f(x_0+\Delta x)-f(x_0)$；当 Δx 无限接近 0 时，如果 Δy 与 Δx 之比存在，则称函数 $y=f(x)$在点 x_0 处可导，并称这个 $\Delta y/\Delta x$ 为函数 $y=f(x)$在点 x_0 处的导数。

以一元二次函数 $y=x^2$ 为例，图 3.27 所示是该函数在二维平面中的图像，并在图上分别画出了(1,1)和(15,225)两点处的函数切线，每个点位置的导数值也就是在该点切线的斜率。该函数的导数用 y'表示，根据求导公式有 $y'=2x$，可以得知：在点 $x=15$ 处的导数值为 30；点 $x=1$ 处的导数值为 2。

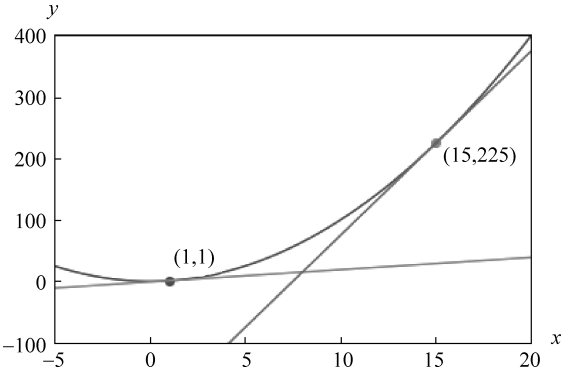


图 3.27 一元二次函数 $y=x^2$ 及在(1,1)和(15,225)处的切线

2. 偏导与梯度概念

对于多元函数 $y=f(x_1,x_2,x_3,\cdots,x_n)$，由于有多个自变量 x_i，各个自变量值的变化都会导致函数值发生变化，函数值的变化是由各个自变量的变化共同决定。因此，需要研究

函数值的变化与每个变量的变化之间的关系，这就是偏导数。

对任意一个变量 x_i，当其他变量为常数值时，函数 f 关于变量 x_i 的导数就称为偏导数，用符号∂表示。

以二元函数 $z=f(x,y)$ 为例，用 $\frac{\partial f(x,y)}{\partial x}$ 表示函数 $f(x,y)$ 关于 x 的偏导数，其几何含义就是在函数 $z=f(x,y)$ 的三维几何图中，y 固定为某个值时（y 方向不变），函数值 z 沿 x 轴方向的变化率；用 $\frac{\partial f(x,y)}{\partial y}$ 表示函数 f 关于 y 的偏导数，指在 x 固定为某个值时（x 方向不变），函数值 z 沿 y 轴方向的变化率。

梯度就是把多元函数的各个变量的偏导数放在一起构成的向量（也称为矢量）。梯度（矢量）既有方向，又有大小。梯度方向表示当函数的各个变量都按照各自偏导数的比例进行增加时，各个增加量合起来构成的方向（类似于力学中，对于某一个点，在多个不同方向上用不同的力来拉这个点，会合成为朝向某一个方向上的力）。而且数学上可以证明，各个变量的值按这样的比例变化时，函数值变化率最大（变化率为梯度的模），即在自变量的变化步长固定的情况下，各自变量增加值的比例等于各偏导数的比例时，函数值的增加量最大。函数 f 的梯度记为 grad(f)，表示为式(3.5)。

$$\mathrm{grad}(f)=\left(\frac{\partial f(x_1,x_2,\cdots,x_n)}{\partial x_1},\frac{\partial f(x_1,x_2,\cdots,x_n)}{\partial x_2},\cdots,\frac{\partial f(x_1,x_2,\cdots,x_n)}{\partial x_n}\right) \tag{3.5}$$

以二元函数 $z=f(x,y)=x^2+y^2+xy$ 为例，由式(3.5)可以得到 z 的梯度如下：

$$\mathrm{grad}(f)=\left(\frac{\partial f(x,y)}{\partial x},\frac{\partial f(x,y)}{\partial y}\right)$$

其中 $\frac{\partial f(x,y)}{\partial x}$ 和 $\frac{\partial f(x,y)}{\partial y}$ 分别表示函数 f 对于 x 和 y 的偏导数。各偏导数求解如下：

(1) 把 y 看作常数，对 x 求导得 $\frac{\partial f(x,y)}{\partial x}=2x+y$；

(2) 把 x 看作常数，对 y 求导得 $\frac{\partial f(x,y)}{\partial y}=x+2y$。

根据式(3.5)，该函数的梯度为 grad(f)=($2x+y$，$x+2y$)。

下面还是以二元函数 $z=f(x,y)=x^2+y^2+xy$ 为例来说明函数值随自变量的变化而变化的情况。图 3.28 是该函数的三维几何图。图中可以看出，不同的(x，y)坐标点，z（函数值）有着不同值。可以把这个三维函数图形象地理解为一座水池的表面，x 和 y 构成水平面，z 为水池表面上任意一个点的高度值。在图中有个 P 点(1,1,3)，其在 x 和 y 水平面上的坐标为(1,1)，对应的高度 z 值为 3。给该点的 x 和 y 分别加上一个数值 Δx 和 Δy，得到一个新点 P1，其在水平面上的坐标为($x+\Delta x$，$y+\Delta y$)，根据函数也可以计算出 P1 点对应的 z 值，那么在三维图中可以标记点 P1($x+\Delta x$，$y+\Delta y$，z_{new})。这个过程可以看作从 P 点移动到了 P1 点，如果 P1 的 z 值大于 P，则说明是在往高处走，反之则是往水池底部走，差值为高度变化值。

因为梯度方向是函数值增加最快的方向，如果要最快到达水池底部，就应该沿着梯度反方向移动，即在任意一个点，其水平坐标是(x，y)，每次变化时的 Δx 和 Δy 的比例都需要为 $(2x+y)/(x+2y)$，且 Δx 和 Δy 均取负值，而 Δx 和 Δy 的绝对值大小与步长相关。

下面进一步说明梯度方向。为了更清楚地演示函数的自变量 x、y 变化对函数值 z 的影响，图 3.29 是图 3.28 的水平投影，即图 3.29 中的每个(x,y)点就是图 3.28 中的(x,y)坐标。当 Δx 和 Δy 取不同值时，图中可以看出，任何一个新点对于 P 点都存在一个方向，图中的 P_t 和 P_x 可用看作是对 P 点的 x 和 y 坐标增加不同的 Δx 和 Δy 值得到。很显然，变化量 Δx 和 Δy 取不同的值，会让新点位于 P 的不同方向上，即这个方向由 Δx 和 Δy 的比例和符号决定，也可以用新点(如图中的点 P_x)到 P 点的直线与水平方向的夹角来描述(如图中的 θ)。

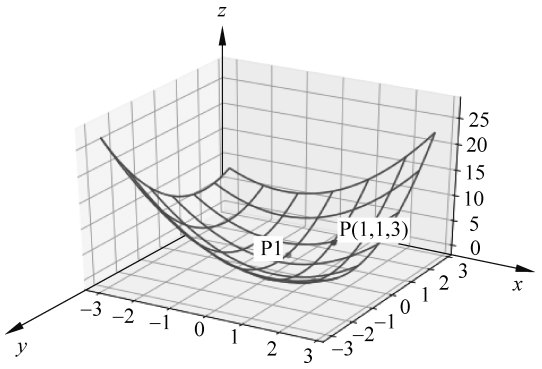


图 3.28　函数 $z=x^2+y^2+xy$ 的三维图

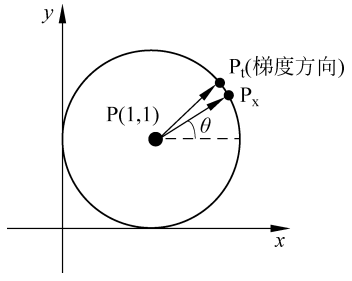


图 3.29　P 点在水平平面上的投影及与 P 点相距为 1 的动点轨迹

为了描述函数值 z 在 P 点往不同方向移动时的变化率，这里对 Δx 和 Δy 的变化设置一个限制条件：$(\Delta x)^2+(\Delta y)^2=1$。这个限制条件的含义是：这个值 1 代表步长，步长为 1 表示从 P 点在水平平面上往四周任意一个方向走一步距离，到达一个新点。对应到三维图中，通过函数 f 可以计算出任意新点对应的 z 值。显然，沿不同方向走一步，对应的 z 值不同，即此时的 z 值变化量不同；因为(x,y)的步长为 1，则该值也是函数值在该点的变化率。以平面中的 P 点(1,1)为例，一些新点相对于 P 的方向及对应的 z 值变化量如表 3.5 所示。

表 3.5　从 P 点往不同方向走 1 步时 z 值的变化量

新点	Δx	Δy	$(x+\Delta x, y+\Delta y)$	z_{new}	角度 θ	z 的变化量
P1	1	0	(2,1)	7	0°	4
P2	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\left(1+\frac{\sqrt{3}}{2},\frac{3}{2}\right)$	$\frac{11}{2}+\frac{7\sqrt{3}}{4}$	30°	$\frac{5}{2}+\frac{7\sqrt{3}}{4}\approx 4.975$
P3	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	$\left(1+\frac{\sqrt{2}}{2},1+\frac{\sqrt{2}}{2}\right)$	$\frac{9}{2}+3\sqrt{2}$	45°	$\frac{3}{2}+3\sqrt{2}\approx 5.743$
P4	$-\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\left(\frac{1}{2},1+\frac{\sqrt{3}}{2}\right)$	$\frac{5}{2}+\frac{5\sqrt{3}}{4}$	120°	$-\frac{1}{2}+\frac{7\sqrt{3}}{4}\approx 1.975$
P5	−1	0	(0,1)	2	180°	1

从该表可以看出，从 P 点往不同方向走一步，z 值的变化量不同。那么往哪个方向上走，z 的增加量最大呢？

从函数梯度角度来看，函数 f 在(1,1)点处，梯度 $\Delta f=(2x+y,2y+x)=(3,3)$。当 Δx 和 Δy 的变化比例与梯度中的偏导数比例相同时，即 $\frac{\Delta x}{\Delta y}=\frac{3}{3}=1$ 时，z 值增加最快，有

$\Delta x=\Delta y=\frac{\sqrt{2}}{2}$(45°方向,即 P3 点方向),这个方向就是 P(1,1)点的梯度方向。表 3.5 只是给出了一些示例数据,还可以带入其他数据验证,P3 点方向的 z 值变化量最大。

3.4.2　多层神经网络的结构和工作过程

感知机只能训练最后一层神经元的权值参数,而反向传播算法则可以用来在多层神经网络的训练过程中调整各层神经元的权值参数。

1. 多层人工神经网络的结构

多层人工神经网络是由大量神经元按多个层次排列、经广泛互连而成的人工神经网络,用来模拟脑神经系统的结构和功能。网络中每一个神经元都和感知机模型一致,都包含对输入数据进行加权乘法的权值(含偏置)、求和运算、激活函数三个部分,激活函数的输出结果为该神经元的输出。图 3.30 所示是一个多层的人工神经网络(输入层只是数据而不是神经元,不算入神经网络层)。每个输入都连接到神经网络中第一层的所有神经元,每个神经元的输出也连接到下一层的所有神经元,因此也称为全连接神经网络。

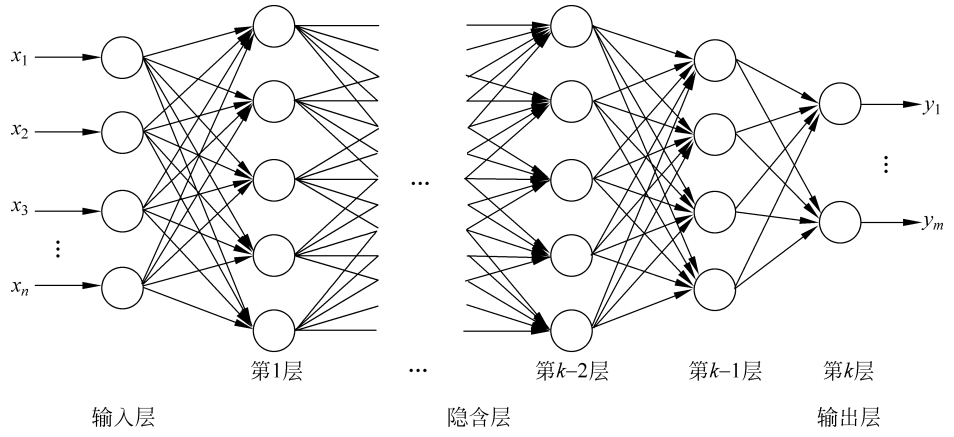


图 3.30　k 层的人工神经网络结构

2. 多层人工神经网络中神经元的激活函数

在多层神经网络中,为了便于采用反向传播(BP)算法进行权值参数学习,不使用感知机中的线性阶跃函数 sgn(x)作为激活函数,而是采用 Sigmoid 函数。Sigmoid 函数的数学描述如式(3.6)所示。

$$f(x)=\frac{1}{1+\mathrm{e}^{-x}} \tag{3.6}$$

其对应的函数图像如图 3.31 所示。它可以将任何输入数值压缩在(0,1)范围内。当神经网络用于进行二分类时,输出层可以只使用一个神经元,用 0 和 1 两个数值分别代表两个类别。在计算过程中,如果样本数据对应的类别是 1,那么希望整个神经网络的输出尽量接近 1;如果样本数据对应的类别是 0,那么希望整个神经网络的输出尽量接近 0。神经网络的学习过程(训练过程)就是尽量调整网络中各个神经元的权值参数,使输出分别接近 0 或者 1。但

通常在进行多分类时，有几个类别输出层就采用几个神经元，每个神经元的输出就代表输入数据为某一个类别的概率。采用两个神经元输出进行两个类别的分类时，一个神经元的输出代表输入数据为类别 0 的概率，另一个输出则代表输入数据为类别 1 的概率，取输出的概率值最大的那个神经元所对应的类别为整个神经网络判断的类别。

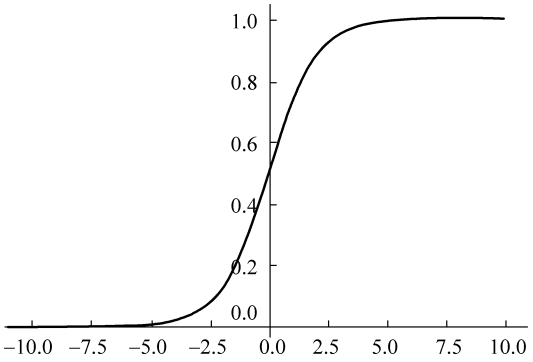


图 3.31 Sigmoid 函数图像

引入 Sigmoid 函数到网络中的原因是：在感知机中，输出数据仅仅由一个简单的线性函数计算得出，其复杂性有限，对数据的表示能力较弱，无法学习和处理图像、视频、音频等复杂类型的数据。而 Sigmoid 函数作为激活函数有如下优势：(1)把输出值限定在(0,1)之间，便于调节；(2)该函数的求导形式简单，为 $f'(x)=f(x)(1-f(x))$，便于计算梯度。

3. 多层人工神经网络的工作过程

多层人工神经网络的工作过程包含两个部分：正向(前向)传播得出网络输出；利用误差反向传播算法进行权值参数调整的学习过程。

1) 正向(前向)传播过程

每个神经元的正向计算过程和感知机模型神经元一致，只是激活函数替换为 Sigmoid 函数。每个神经元进行计算，如式(3.7)所示。

$$y=\text{Sigmoid}\left(\sum_{i=1}^{n}w_i x_i+b\right) \tag{3.7}$$

其中，y 表示该神经元的输出，n 表示该神经元共有 n 个输入(输入层不是神经网络层)，w_i 表示该神经元的第 i 个输入的权值参数，x_i 表示该神经元的第 i 个输入值，b 表示该神经元的偏置。

为了计算过程中的统一表示和简化公式，也把偏置 b 作为一个值固定为 1 的输入，用 x_0 表示，用 x_0 与一个可变的权值参数 w_0 的乘积来代替偏置 b，则计算公式等价变换为式(3.8)。

$$y=\text{Sigmoid}\left(\sum_{i=1}^{n}w_i x_i+w_0 x_0\right)=\text{Sigmoid}\left(\sum_{i=0}^{n}w_i x_i\right) \tag{3.8}$$

其中 x_0 为固定的输入值 1。

正向计算过程为：根据上面的公式，从第一层神经网络开始，逐层计算每个神经元的输出，最终得到最后一层神经元的输出值。在应用该神经网络时，就根据类别判定规则，把输

出值转换为类别结果。

2）利用误差反向传播算法进行权值参数学习

在多层神经网络最初建立的时候，各个权值参数都是未知的，通常被设置为随机值。显然，这时候正向计算出来的结果存在误差。误差反向传播算法就是利用这个误差来调整神经网络中的所有权值参数，让重新正向计算出来的误差减小。当误差值降低到可接受的程度就结束学习过程，这时候的神经网络就可以使用了。当然，这个神经网络不一定能对所有输入都得出百分之百正确的结果，其错误率符合预期、能被接受就可以。

误差反向传播算法的具体学习过程如下：

（1）确定训练数据集，其中的每一个样本都由输入信息和期望的输出结果两部分组成；

（2）从训练集中选取任一样本，把样本的输入信息作为网络输入；

（3）进行正向传播，逐层计算出各个神经元处理后的结果，最终得到最后一层神经元的输出结果；

（4）计算神经网络最后一层神经元的输出结果与期望的输出结果之间的误差；

与感知机有所不同，在误差反向传播算法中，不是直接使用期望的输出值减去网络输出值作为误差评价，而是使用一个损失函数来评价网络模型的效果，即评价该神经网络模型的期望值与真实值之间的差距。一般可以采用平方差函数，如式(3.9)所示。

$$E=\frac{1}{2}\sum(O_{\text{target}}-O_{\text{output}})^2 \tag{3.9}$$

其中 O_{target} 为期望输出值，而 O_{output} 为实际输出值；总误差为最后一层所有神经元的误差平方和；前面的 1/2 是为了便于求导计算，对整个误差值的缩放不影响网络运行。

（5）根据上面得到的总误差 E，采用梯度的概念对网络中所有权值参数进行调整，使得调整后的网络总误差减小。权值参数的调整基于函数梯度反方向为函数值（这里的误差）减小最快的方向这一规则，其基本原理分析如下。

因为总误差是由于网络中的所有权值参数设置不合理导致的，而在网络训练过程中，网络输入值是给定的，所有权值参数都可以改变，因此总误差与权值参数的关系可以看作一个关于各个权值参数的函数，如式(3.10)所示。

$$E=f(w_0,w_1,w_2,\cdots,w_{L-1}) \tag{3.10}$$

其中，L 为网络中权值参数的总个数。

网络训练的目的就是通过调整权值参数来让网络总误差减小到可接受的程度。根据前面对多元函数的梯度的介绍可知：为了让 E 最快减小，让各个权值参数同步按照梯度反方向来改变就可以实现，而梯度就是由函数对于各个权值参数的偏导数构成，那么就需要求各个参数的偏导数。

在实际计算中，由于网络层次深、权值参数多，神经网络中前面一些层次的权值参数的偏导很难直接计算，因此引入了数学中的链式求导法则，可以从后往前逐层累积偏导数并保存（偏导数与误差相关，因此称为误差反向传递），从而简化前面层次神经元权值参数的偏导数计算。

（6）对训练样本集中的每一个样本，重复执行步骤(3)～(5)，直到整个训练样本集的总误差达到要求时为止。

3.4.3 实现异或运算的多层神经网络案例

1. 异或运算介绍

异或运算是一种逻辑运算，硬件上可以用异或运算电路实现。单层感知机只能实现与运算和或运算，无法实现异或运算。异或运算的规则是：输入两个值，分别为 0 或 1，如果同为 0 或 1，结果为 0(表示假)；如果两个值不同，则结果为 1。其真值表如表 3.6 所示。

表 3.6 异或逻辑运算真值表

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

2. 多层神经网络结构以及应用过程

用于实现异或运算的多层神经网络可以设计为如图 3.32 所示的结构：输入层为两个输入数据 x_1、x_2；第一层包含两个神经元，分别是 $H1$、$H2$；第二层(输出层)也包含两个神经元，分别为 $Y1$、$Y2$，各自输出类别结果为假(0)和为真(1)的概率；并在输入层和除输出层之外的神经网络层中都设置固定的输入值 1，通过相应权值来取代下一层神经元中的偏置值。

每个输入数据为 0 或者 1；在网络训练时，用 $Y1$ 输出为 1 作为输入数据是类别 0 的期望输出值(标签值)，并同时期望此时 $Y2$ 输出尽量为 0，则类别 0 的期望输出是$[oy_1, oy_2]=[1,0]$，以此计算误差；用 $Y2$ 输出为 1 作为类别 1 的期望输出值(标签值)，并期望此时 $Y1$ 输出尽量为 0，则类别 1 的期望输出是$[oy_1, oy_2]=[0,1]$，以此计算误差。而在使用网络时，比较两个神经元输出值的大小，以输出值最大的神经元所对应的类别为分类结果。

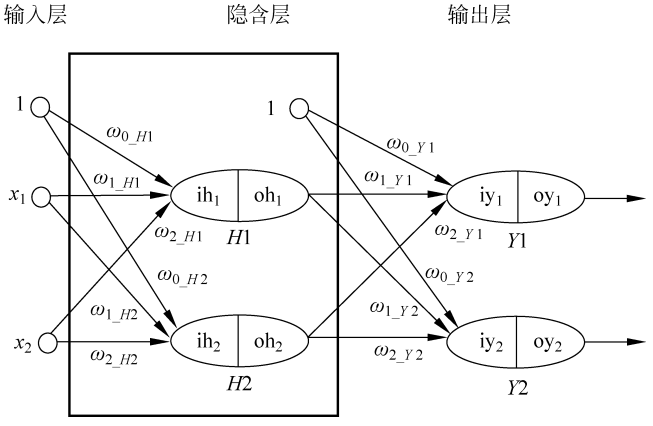


图 3.32 实现异或的多层神经网络及前向传播流程

该神经网络前向传播时，神经元采用 Sigmoid 函数作为激活函数，下面用 f 代替，即 $f(x)=\dfrac{1}{1+e^{-x}}$。如图 3.32 所示，根据神经元的输出计算公式，各个数据的计算过程如下。

(1) 从输入层到隐含层的输出计算为：

$$ih_1 = w_{0\_H1} + x_1 w_{1\_H1} + x_2 w_{2\_H1}$$
$$ih_2 = w_{0\_H2} + x_1 w_{1\_H2} + x_2 w_{2\_H2}$$
$$oh_1 = f(ih_1)$$
$$oh_2 = f(ih_2)$$

(2) 从隐含层到输出层的输出计算为：

$$iy_1 = w_{0\_Y1} + oh_1 w_{1\_Y1} + oh_2 w_{2\_Y1}$$
$$iy_2 = w_{0\_Y2} + oh_1 w_{1\_Y2} + oh_2 w_{2\_Y2}$$
$$oy_1 = f(iy_1)$$
$$oy_2 = f(iy_2)$$

到此时，前向传播已经完成。根据最后两个神经元的输出结果，可以判定预测结果。

以异或真值表中的第一个样本(0,0,0)为例，假设神经网络的权值参数为：$w_{0\_H1} = w_{0\_H2} = w_{0\_Y1} = w_{0\_Y2} = 0.3$，$w_{1\_H1} = w_{1\_H2} = w_{1\_Y1} = w_{1\_Y2} = 0.1$，$w_{2\_H1} = w_{2\_H2} = w_{2\_Y1} = w_{2\_Y2} = 0.2$。代入数据可得如下计算过程。

(1) 从输入层到隐含层：

$$ih_1 = w_{0\_H1} + x_1 w_{1\_H1} + x_2 w_{2\_H1} = 0.3$$
$$ih_2 = w_{0\_H2} + x_1 w_{1\_H2} + x_2 w_{2\_H2} = 0.3$$
$$oh_1 = f(ih_1) = 0.574$$
$$oh_2 = f(ih_2) = 0.574$$

(2)从隐含层到输出层：

$$iy_1 = w_{0\_Y1} + oh_1 w_{1\_Y1} + oh_2 w_{2\_Y1} = 0.472$$
$$iy_2 = w_{0\_Y2} + oh_1 w_{1\_Y2} + oh_2 w_{2\_Y2} = 0.472$$
$$oy_1 = f(iy_1) = 0.6159$$
$$oy_2 = f(iy_2) = 0.6159$$

在这个例子中，由于神经网络的参数不合理，两个神经元输出的值一样，无法正确判断输入数据对应的类别，说明当前存在较大误差。

3. 神经网络的权值参数训练过程(学习过程)

设置学习率为 0.5，初始神经网络的权值参数为：$w_{0\_H1} = w_{0\_H2} = w_{0\_Y1} = w_{0\_Y2} = 0.3$，$w_{1\_H1} = w_{1\_H2} = w_{1\_Y1} = w_{1\_Y2} = 0.1$，$w_{2\_H1} = w_{2\_H2} = w_{2\_Y1} = w_{2\_Y2} = 0.2$，进行 2000 轮训练(每轮均把 4 个样本依次送进去)。部分训练中间数据如下所示(计算过程较为复杂，这里只给出结果)。

1) 第一轮训练

第一个样本前向计算的结果如表 3.7(a)所示，反向调整权值参数的结果如表 3.7(b)所示。

表 3.7 第一个样本的相关计算

(a) 第一个样本前向计算的结果

序号	样本输入	样本值 y	$H1$ 输出	$H2$ 输出	$Y1$ 输出	$Y2$ 输出	总误差
1	(0,0)	0	0.5744	0.5744	0.6159	0.6159	0.2634

(b) 根据第一个样本的误差反向调整权值的结果

	$Y1$ 权值调整	$Y2$ 权值调整	$H1$ 权值调整	$H2$ 权值调整
调整前	(0.3, 0.1, 0.2)	(0.3, 0.1, 0.2)	(0.3, 0.1, 0.2)	(0.3, 0.1, 0.2)
调整后	(0.3136, 0.1261, 0.2261)	(0.2272, 0.0582, 0.1582)	(0.2998, 0.1, 0.2)	(0.2996, 0.1, 0.2)

第二个样本前向计算的结果如表 3.8(a)所示,反向调整权值参数如表 3.8(b)所示。

表 3.8 第二个样本的相关计算

(a) 第二个样本前向计算的结果

序号	样本输入	样本值 y	$H1$ 输出	$H2$ 输出	$Y1$ 输出	$Y2$ 输出	总误差
2	(0,1)	1	0.6224	0.6224	0.6301	0.6018	0.5413

(b) 根据第二个样本的误差反向调整权值的结果

	$Y1$ 权值调整	$Y2$ 权值调整	$H1$ 权值调整	$H2$ 权值调整
调整前	(0.3136, 0.1261, 0.2261)	(0.2272, 0.0582, 0.1582)	(0.2993, 0.1, 0.2)	(0.2986, 0.1, 0.2)
调整后	(0.2906, 0.0803, 0.1803)	(0.2914, 0.0878, 0.1878)	(0.2993, 0.1, 0.1985)	(0.2990, 0.1, 0.1978)

第三个样本前向计算的结果如表 3.9(a)所示,反向调整权值参数的结果如表 3.9(b)所示。

表 3.9 第三个样本的相关计算

(a) 第三个样本前向计算的结果

序号	样本输入	样本值 y	$H1$ 输出	$H2$ 输出	$Y1$ 输出	$Y2$ 输出	总误差
3	(1,0)	1	0.5985	0.5984	0.6098	0.6122	0.8024

(b) 根据第三个样本的误差反向调整权值的结果

	$Y1$ 权值调整	$Y2$ 权值调整	$H1$ 权值调整	$H2$ 权值调整
调整前	(0.2906, 0.0803, 0.1803)	(0.2914, 0.0878, 0.1878)	(0.2993, 0.1, 0.1985)	(0.2990, 0.1, 0.1978)
调整后	(0.2695, 0.0370, 0.1370)	(0.3048, 0.1154, 0.2154)	(0.2992, 0.0996, 0.1985)	(0.2986, 0.0989, 0.1979)

同样可获得第四个样本的前向计算结果，并根据其误差反向调整权值，具体数据略。

2）第 2 轮到第 1999 轮训练

每轮训练均与第 1 轮训练的方法相同，依次输入 4 个样本，获得前向计算结果，并根据其误差反向调整权值，具体数据略。

3）第 2000 轮训练

前 3 个样本输入和调整的具体数据略。

第四个样本前向计算的结果如表 3.10(a)所示，反向调整权值参数的结果如表 3.10(b)所示。

表 3.10　第四个样本的相关计算

(a) 第 2000 轮训练中第四个样本前向计算的结果

序号	样本输入	样本值 y	$H1$ 输出	$H2$ 输出	$Y1$ 输出	$Y2$ 输出	总误差
8000	(1,1)	0	0.0236	0.000	0.8683	0.5193	0.1590

(b) 第 2000 轮训练中根据第四个样本的误差反向调整权值的结果

	$Y1$ 权值调整	$Y2$ 权值调整	$H1$ 权值调整	$H2$ 权值调整
调整前	(2.0071, −5.1635, 6.5339)	(1.4572, 3.2810, −6.6959)	(7.3944, −5.5714, −5.5477)	(2.3438, −6.3456, −6.4506)
调整后	(2.0222, −5.1633, 6.5339)	(1.3628, 3.2795, −6.6959)	(7.3516, −5.5772, −5.5534)	(2.3439, −6.3456, −6.4506)

4. 神经网络的使用

通过上述 2000 次训练之后，得到整个网络的权值参数如表 3.10(b)所示。根据表中的权值参数，把所有数据依次送入网络进行前向传播计算，计算结果如表 3.11 所示，各个数据均能够得到正确的分类结果，实现了异或运算。

表 3.11　使用该神经网络得到的分类结果

	样本输入	$H1$ 输出 (oh_1)	$H2$ 输出 (oh_2)	$Y1$ 输出 (oy_1)	$Y2$ 输出 (oy_2)	较大概率	分类结果
1	(0,0)	0.9994	0.9124	0.9440	0.0556	0.9440	0
2	(0,1)	0.8579	0.0162	0.0910	0.9373	0.9373	1
3	(1,0)	0.8550	0.0180	0.0932	0.9361	0.9361	1
4	(1,1)	0.0223	0.000	0.8707	0.5183	0.8707	0

3.4.4 梯度消失与梯度爆炸问题

BP 算法基于梯度下降策略，以误差函数的负梯度方向对权值参数进行调整。输入数据通过各层神经元前向传递，各层逐层对其进行处理。在训练过程中，需要计算最终的误差评估函数对所有神经元权值参数的偏导数，而根据链式传递法则可知，误差评估函数对前面层次神经元的权值参数的偏导数是由误差评估函数对后面各层神经元输出计算公式的偏导数的连续乘积计算得来，当存在非常多的网络层次时，各层神经元函数的偏导数连续相乘的结果就会接近 0 或者非常大，这两种情况分别称为梯度消失和梯度爆炸，会造成整个网络训练不稳定而无法成功训练。

3.5 深度学习的卷积神经网络模型原理

卷积神经网络(CNN)主要模仿生物大脑皮层中神经网络工作的局部感受和分层处理的特点，构造多层/深层神经网络来处理图像等数据(现在也用于自然语言处理等各个领域)。卷积神经网络结构中包括卷积层、池化层、全连接层等不同类型的层次，其中每个神经元对上一层神经元的输出数据进行不同类型的处理。其主要特征是在卷积层采用局部连接和权值共享的方式进行连接，大大降低了权值参数的数量，而池化层可以大幅降低输入维度，从而降低网络复杂度，因此可以构造比较深的神经网络，实现深度学习。下面对这些概念和技术进行讨论。

3.5.1 计算机中的图像

计算机在处理图像时，首先将图像以点阵方式表示和存储。一幅图像的尺寸表示该图像在水平方向和垂直方向上分别有多少个像素(像素是图像处理的最小单位)。根据每个像素的颜色表示方式，计算机中的图像可以分为三类：二值图像、灰度图像、彩色图像。

二值图像是指图像中的点只有黑、白两种颜色，可以直接用 1 位二进制值 0 表示黑色点，1 位二进制值 1 表示白色点。一幅图像有多少行、列的像素，图像就可以用多少个二进制的 0 或 1 来排列成行、列的形式表示。

灰度图像是二值图像的“进化”版，将每个像素的颜色值从黑到白分为 256 级，黑色为 0，白色为 255，中间的值代表不同程度的灰色(也叫灰度值)，数值越大代表这个像素越白(或越亮)，数值越小代表像素越黑(或越暗)。

现实世界中的彩色由红、绿、蓝三原色构成。计算机在表示和存储彩色图像时，对每个像素采用了 3 个值来分别描述构成该像素的红(R)、绿(G)、蓝(B)3 个颜色。每个描述红色、绿色、蓝色的值也可以分别分为 256 级。

图 3.33 展示了人眼看到的灰度图像及其在计算机中存储的灰度值。左图方框框出来的区域中，左上角比较亮，其最左上角那个点的颜色值为 193；右上角比较暗，最右上角那个点的颜色值为 118。

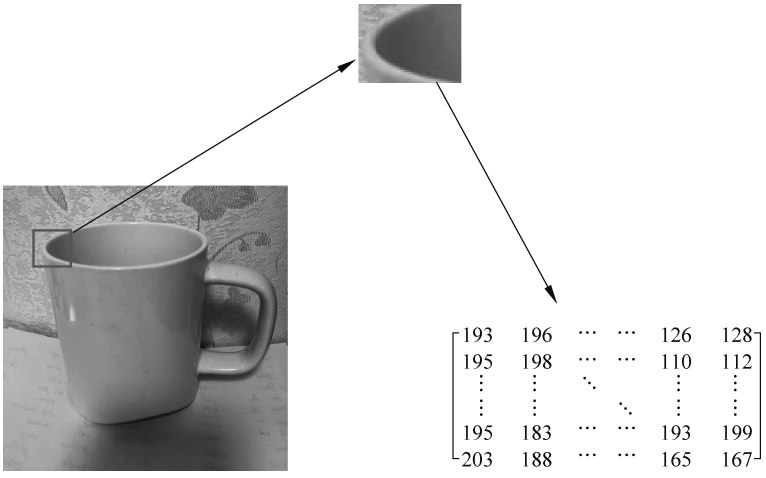


图 3.33　灰度图像在计算机中的表示

3.5.2　卷积运算

为了让图像中的某些特征更加显著，往往需要进行一些处理。例如，为了便于对图像进行分割或对图像的纹理特征、形状特征进行分析，需要强化图像的轮廓或边缘信息；为了让图像看起来更清晰，会对图像进行锐化处理；为了降低图像预处理时的噪声，会对图像进行平滑/模糊处理。具体处理效果如图 3.34 所示。

原图	效果图	效果说明
		边缘检测：突出图像的边缘信息，给出了图像轮廓的位置
		锐化操作：补偿图像的轮廓，增强图像的边缘及灰度跳变部分，使图像变得更加清晰
		模糊操作：图像中物体对象和内部边缘变得相对平滑，使得图像变得模糊

图 3.34　图像的各种处理效果

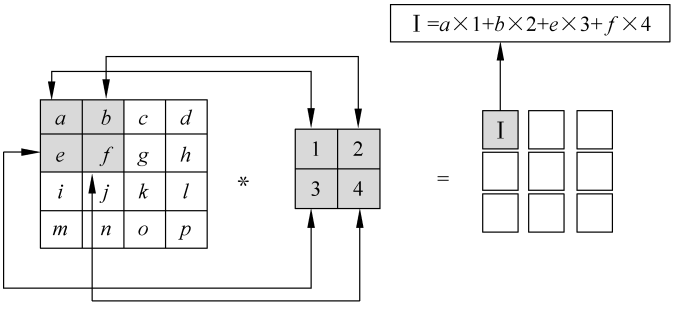
这些效果的实现通常采用一种称为卷积运算的方法，对原始图像的各个像素值进行运算得到，不同的效果就是采用了不同的卷积核参数进行运算。下面通过一个具体的案例，来对卷积核和卷积运算过程进行介绍。

首先假设有一个尺寸是 4×4 的灰度图像，用一个 4×4 矩阵表示，每个像素的灰度值用英文字母 a～p 来代表，如图 3.35(a)所示。

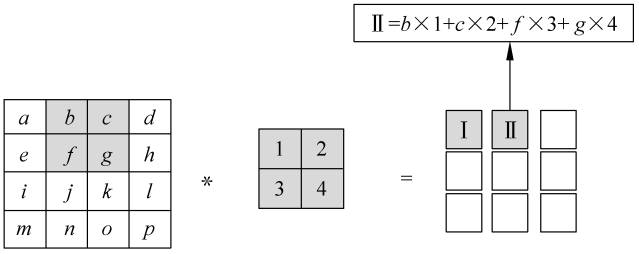
卷积核是一个矩阵，里面的参数一般是根据经验来人为设置。图 3.35 中是一个尺寸为 2×2 的卷积核，卷积核中的参数用数字编号 1～4 来代表。

卷积计算过程如图 3.35 所示，卷积核从图像的左上方开始，从左至右、从上到下依次滑动，每次滑动可以间隔一个或多个像素，称为步长。每滑动一次，就将卷积核中的各个参数与对应图像区域的像素值进行点乘运算(即对应点的数值相乘，最后对所有乘积相加)，得到新图像的一个像素值。随着滑动的进行，得到的像素值将构成一个新的图像，作为卷积计算的最终结果(也称为特征图)。还可以根据需要，对特征图中每个值都乘以一个系数来进行整体缩放。图 3.35 所示是按水平方向和垂直方向步长均为 1(每次滑动一个点)进行计算得到的特征图各像素值，也是用符号来表示。点乘计算中的对应点如图 3.35(a)中双向箭头所示，计算特征图左上角的像素值Ⅰ时，像素 a 与卷积核参数 1 相对应，像素 b 与卷积核参数 2 相对应，像素 e 与卷积核参数 3 相对应，像素 f 与卷积核参数 4 相对应，因此Ⅰ$=a\times1+b\times2+e\times3+f\times4$，以此类推。

从上述计算中可以看出，用卷积核每做一次计算，可以得出特征图中的一个像素，即这个像素包含了原始图像中对应 4 个像素的信息，因此可以理解为该像素包含了原始图像的部分信息。同时，卷积核的尺寸也意味着视觉感受区域(也称为感受野(Receptive Field))的大小。

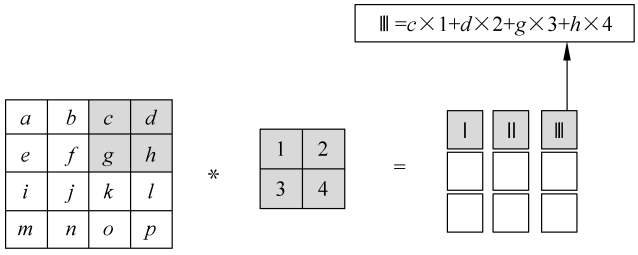


(a) 特征图中像素值Ⅰ的计算

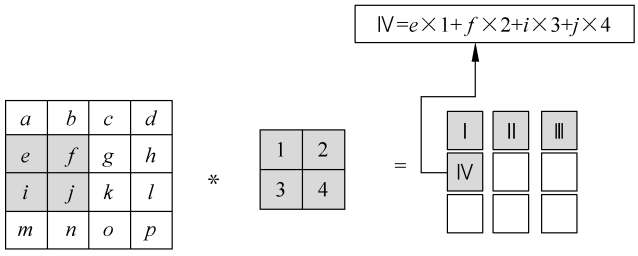


(b) 特征图中像素值Ⅱ的计算

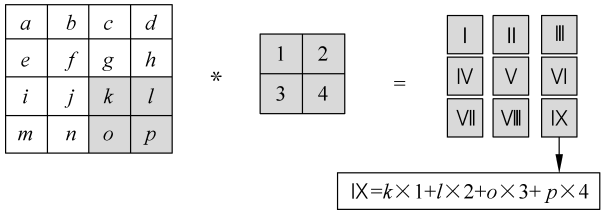
图 3.35　卷积计算过程



(c) 特征图中像素值Ⅲ的计算



(d) 特征图中像素值Ⅳ的计算



(e) 特征图中像素值Ⅸ的计算

图 3.35　（续）

对于由三原色(红、绿、蓝)构成的彩色图像,则需要把三种原色当作三个独立的图(也称为三个通道),各使用一个卷积核来处理,然后将处理后的三个特征图上的对应像素值直接通过求和方式叠加,得到该彩色图像的特征图。如图 3.36 所示,每个通道图像的尺寸是 3×3,卷积核大小是 2×2,卷积和叠加运算后的最终特征图的尺寸是 2×2。

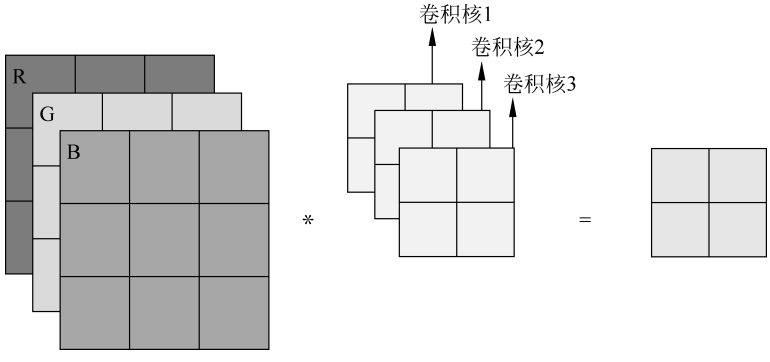


图 3.36　三个通道的卷积处理

图像处理中的卷积核通常通过经验来设计，图 3.34 所示的图像处理效果，可以通过分别对原始图像数据使用图 3.37 所示的卷积核得到。

通过这个案例不难发现，卷积运算是一种通过将图像的各点进行线性变换获取新值的过程，卷积核中的参数可以看作线性变换过程中的权重参数，因此，可以转换为对应的神经网络计算。以前面对 4×4 的灰度图像进行卷积运算为例，其对应的神经网络如图 3.38 所示。

(1) 图像中的每个像素值都是一个输入，因此有 16 个输入值，分别用字母 $a\sim p$ 表示；特征图的每个点值分别用一个神经元来计算，因此有 9 个神经元，输出值分别为Ⅰ～Ⅸ。

(2) 将图像中进行一次卷积计算用到的 4 个像素连接到对应的神经元(即每个神经元只有 4 个输入)，4 个连接各自的权值正好分别是卷积核的 4 个参数。

如图 3.38(a)所示，a，b，e，f 4 个点连接到输出Ⅰ的神经元(用 $a\rightarrow$Ⅰ表示 a 连接到输出Ⅰ的神经元)，则有 $a\rightarrow$Ⅰ的权值为编号 1，$b\rightarrow$Ⅰ的权值为编号 2，$e\rightarrow$Ⅰ的权值为编号 3，$f\rightarrow$Ⅰ的权值为编号 4。

神经元的激活函数取 $f(x)=x$，则根据神经网络的前向运算方法有Ⅰ$=a\times1+b\times2+e\times3+f\times4$，从而得出特征图中第一个点Ⅰ的值，与卷积运算完全一样。

同样，如图 3.38(b)所示，e，f，i，j 4 个点连接到输出Ⅳ的神经元，则有 $e\rightarrow$Ⅳ的权值为编号 1，$f\rightarrow$Ⅳ的权值为编号 2，$i\rightarrow$Ⅳ的权值为编号 3，$j\rightarrow$Ⅳ的权值为编号 4。则Ⅳ$=e\times1+f\times2+i\times3+j\times4$，与卷积运算完全一样。

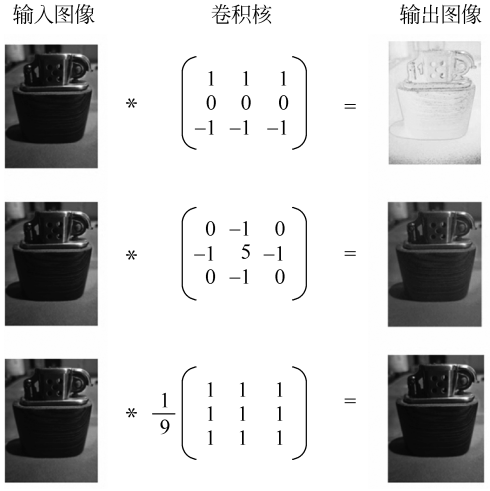
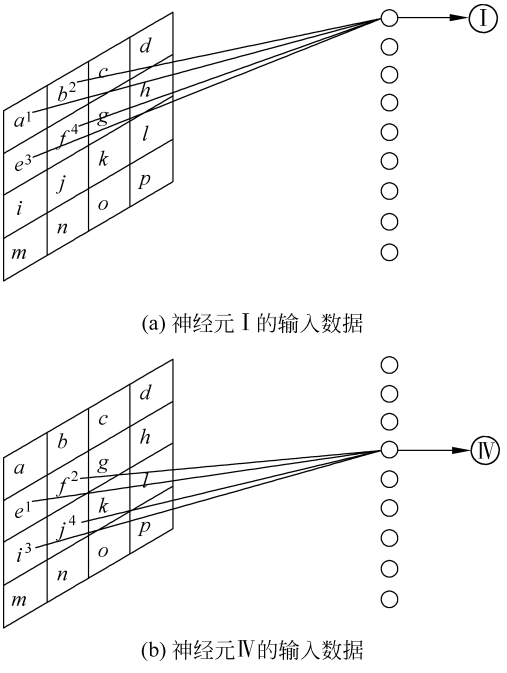


图 3.37 卷积处理效果

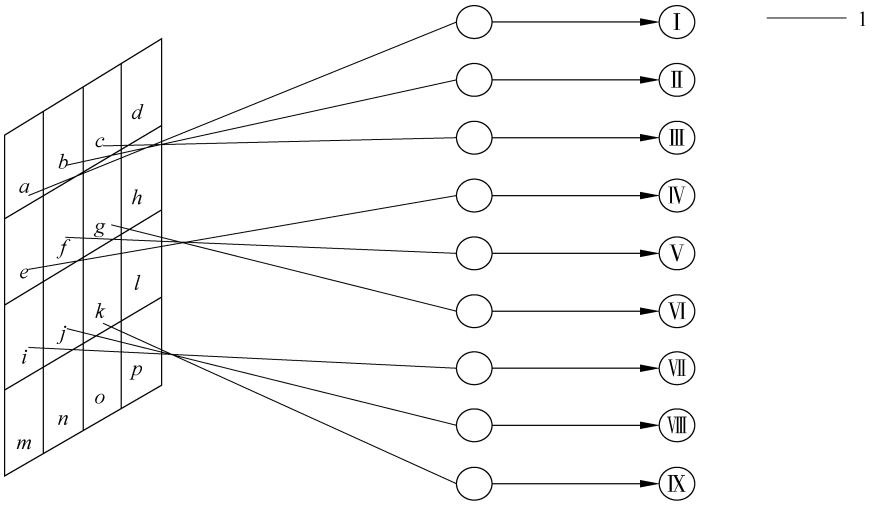


(a) 神经元Ⅰ的输入数据

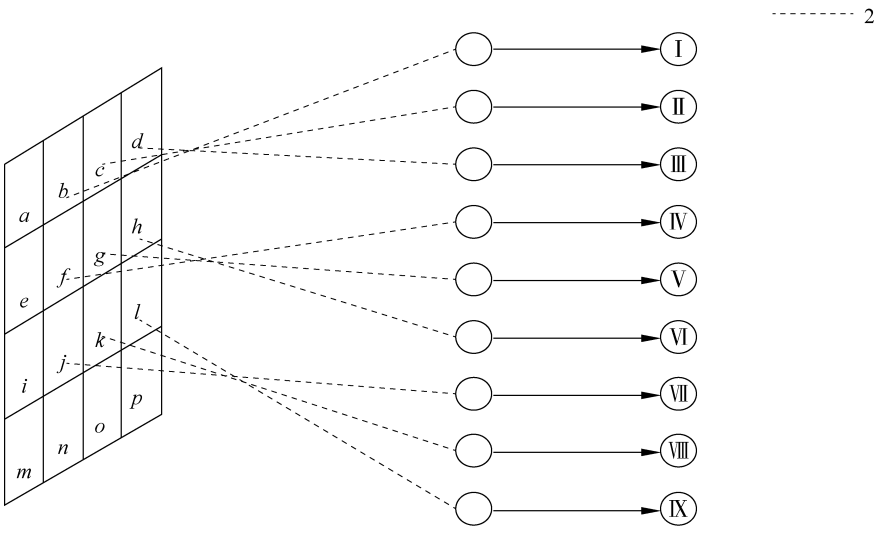
(b) 神经元Ⅳ的输入数据

图 3.38 卷积计算转换为神经网络计算

上述神经网络中，每个输入只连接到网络下一层的部分神经元，因此被称为局部连接；同时，输入到神经元的多个连接一共只使用了 4 个权值，很多连接的权值是相同的，被称为权值共享，如图 3.39 所示。



(a) 共享权值参数1的连接



(b) 共享权值参数2的连接

图 3.39　共享权值参数的示意图

可以看到，计算点Ⅰ、Ⅱ、Ⅲ、Ⅳ、Ⅴ、Ⅵ、Ⅶ、Ⅷ、Ⅸ时，分别对 a、b、c、e、f、g、i、j、k 输入使用了相同的权值参数 1。

其他多条连接分别使用相同权值参数 2、3、4 的情况与此类似。

因此，图像中的卷积计算可以对应成为神经网络计算，且这个神经网络计算具有非全连接和权值共享特征，其结果也同样是基于局部感受野的特征图，从而能够把卷积计算作为神经网络的一部分，称为卷积层。把神经网络卷积层与原来的全连接网络相连接进行计算，还有另外一个很大的好处：不用对卷积核参数（权值参数）进行预先设置，只要有了训练数据，采用 BP 算法反向传递能够自动学习并产生最合适的权值参数。

3.5.3 池化运算

由于高分辨率的图像数据量太大，卷积运算后的特征图点仍然很多，输入到全连接网络时还是存在权值参数多、高计算量的问题。为了减少输入数据量，通常还采用称为池化(pooling)的运算。池化运算实质上就是一种下采样操作，是将一定大小的区域中(称为池化窗口)的多个点值用一个值来代替。

池化运算一般有最大池化和平均池化。最大池化就是取出每块区域像素值里的最大值作为特征图中的一个点值，平均池化则是取区域像素值的平均值，构成特征图的一个点值。池化运算也可以设置步长，一般步长与池化窗口大小相同，如果步长小于池化窗口大小，则称为重叠池化。

下面通过案例来介绍池化运算的过程。如图 3.40 所示，输入一个 4×4 的数字图像，采用 2×2 大小的池化窗口，步长与池化窗口大小相同，则将原图像按 2×2 大小分块，共有 4 块。分别计算出 4 个值作为结果，运算方法为：图中第一块的像素值分别为 2、3、4、5，最大池化则采用其中的最大值 5 作为结果，平均池化则采用平均值 3.5 作为结果；同理可得到其他所有块的结果。最终得到的最大池化特征图如图 3.40 右上部分所示即 $\begin{bmatrix}5 & 9\\8 & 6\end{bmatrix}$，平均池化特征图则如图 3.40 右下部分所示，即 $\begin{bmatrix}3.5 & 7\\5.5 & 3.5\end{bmatrix}$。

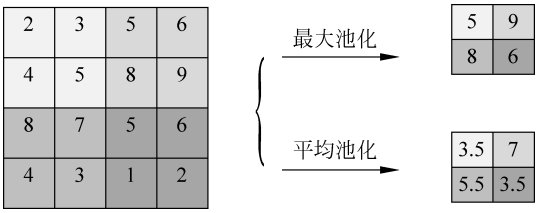


图 3.40 池化运算示例

池化运算能够在减少参数同时保留图像主要特征，还具有平移、旋转、尺度等不变性。如图 3.41 所示，有一个 12×12 的图像，里面有一个数字 7，经过 2×2 的池化窗口进行非重叠最大池化，变成了 6×6 的特征图。从图中可以看出，在保留该图像主要特征的情况下(还是能看出数字 7)，将图像缩小到了原图的 1/4 大小，大大减少了对其进行处理的神经网络中的输入，从而减少了权值参数数量。

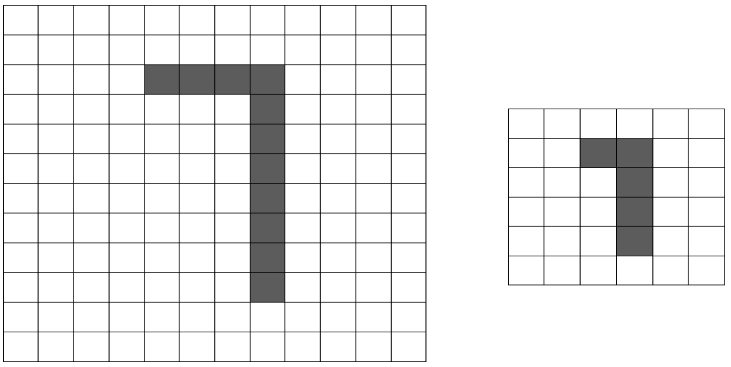


图 3.41 池化运算效果

3.5.4　卷积神经网络中的其他相关技术

1. ReLU 激活函数

在感知机中，神经元使用阶跃函数作为激活函数；基于 BP 算法的多层神经网络通常使用 Sigmoid 激活函数；而在卷积神经网络中，为了避免网络层次带来的梯度爆炸和梯度消失问题，也为了简化运算，采用了线性整流(Rectified Linear Unit，ReLU)函数作为激活函数。ReLU 函数公式非常简单，如式(3.11)所示：

$$f(x)=\max(0,x) \tag{3.11}$$

就是如果输入为负值则输出 0，输入为正值就直接输出。其函数图像如图 3.42 所示。

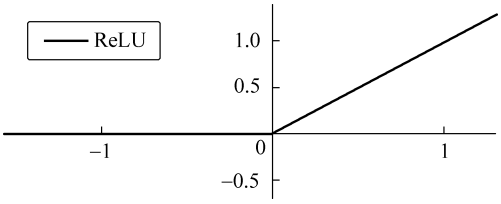


图 3.42　ReLU 函数图像

2. softmax 函数

在前面的反向传播神经网络中，采用 Sigmoid 函数作为激活函数，输出范围是(0,1)，可以直接作为输入数据属于某个类别的概率。而卷积神经网络采用 ReLU 函数作为激活函数，网络的输出值范围为 0 到正无穷，导致在进行网络训练的时候难以设置类别在训练时的标签值(没有一个合适的数值可以作为某个类别的标签值)。因此，对最后一层神经元的输出使用 softmax 函数进行归一化处理，得到 0～1 之间的概率，就可以用作属于某个类别的训练标签。

假设输出层有 K 个神经元(实现 K 个类别的分类)，每个神经元输出值为 a_i($0\leqslant a_i<\infty$)，则可以将这 K 个输出值采用如式(3.12)所示的 softmax 函数公式转换为概率 y_i，转换后的各个 y_i 的总和为 1。

$$y_i=\frac{e^{a_i}}{\sum_{i=1}^{K}e^{a_i}} \tag{3.12}$$

也就是把输出的具体数值转换为其在所有输出值的和里面所占的比例，因此可以理解为概率。在使用时，假设神经网络最后一层用 3 个输出神经元进行三分类，如果输出的 3 个数据为[188,10,2]，通过 softmax 函数得到[0.94,0.05,0.01]。在训练过程中，就可以用[1,0,0]作为该类别的标签值。根据对应的 softmax 函数的输出值[0.94,0.05,0.01]，计算出神经网络的总误差，然后通过反向传播算法来修改网络中所有神经元的权值；如果是在使用神经网络进行类别判断，因为当前 3 个数据中 0.94 是最大的，就用其作为本次分类的判别结果，即分类结果为[1,0,0]对应的类别，而如果实际上类别应该是[0,1,0]或[0,0,1]，就说明神经网络对该次输入数据的分类错误。

3. 交叉熵损失函数

在卷积神经网络中，使用 ReLU 函数作为激活函数，输出层中各个神经元的输出数据经过 softmax 函数转化为概率，各个概率之间有关系(总和为 1)。这时，采用交叉熵损失函数(cross entropy loss function)来进行整个网络输出的误差评估将更为有效：误差越大的

时候，梯度就越大，参数 w 调整得越快，训练速度也就越快。如果最后一层神经元数量为 K，每个训练样本输出的交叉熵损失函数如式(3.13)所示。

$$E=-\left(\sum_{i=1}^{K}(y_i\ln(a_i)+(1-y_i)\ln(1-a_i))\right) \tag{3.13}$$

如果神经网络采用批训练方式(即每次训练计算出所有样本的平均误差，进行一次权值参数调整)，如果一次训练采用 N 个样本，则计算每批训练的平均交叉熵损失的函数如式(3.14)所示。

$$E_{\text{avg}}=-\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K}(y_i\ln(a_i)+(1-y_i)\ln(1-a_i)) \tag{3.14}$$

3.5.5 一个基本的多分类卷积神经网络结构

为了便于介绍卷积神经网络的工作过程，这里设计了一个基本的卷积神经网络，能够用来识别 0 和 1 两个数字的图像。下面将基于这个网络结构来讨论卷积神经网络的训练过程和使用。

1. 基本的卷积神经网络结构

基本的卷积神经网络结构如图 3.43 所示。使用 5×5 的黑白图像作为输入数据；第一层卷积层采用的卷积核大小为(2,2)，步长为(1,1)，使用 ReLU 激活函数，卷积计算得到的特征图大小为 4×4；对特征图进行 2×2 的非重叠最大池化运算，得到 2×2 的输出数据；把池化输出拉伸成 1×4 的特征向量，输入到全连接层；为了简化网络，这里的全连接层只设置了一层神经元；全连接层的输出通过 softmax 层转换，得出概率输出。由于待分类图像为字符 1 或者字符 0，softmax 层输出的两个概率值[x，y]中，x 表示图像为字符 0 的概率，y 表示图像为字符 1 的概率，取 x 和 y 中最大的值，就得到预测的图像分类结果。

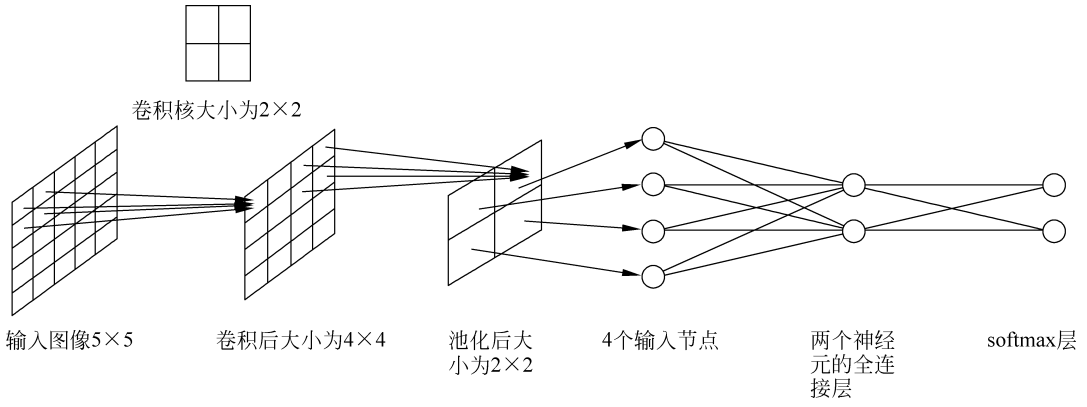


图 3.43 基本的卷积神经网络结构

2. 训练样本数据

采用分别为字符 0 和字符 1 的两幅图作为训练样本，其图像和数据表示如表 3.12 所示。样本图像为字符 0 时，采用[1,0]作为标签；样本图像为字符 1 时，采用[0,1]作为标签。

表 3.12　训练样本数据

图　像	各像素的值矩阵	对应的标签
	$\begin{bmatrix}1&1&1&1&1\\1&0&0&0&1\\1&0&0&0&1\\1&0&0&0&1\\1&1&1&1&1\end{bmatrix}$	[1, 0]
	$\begin{bmatrix}0&1&1&0&0\\0&1&1&0&0\\0&1&1&0&0\\0&1&1&0&0\\0&1&1&1&0\end{bmatrix}$	[0,1]

3. 第 1 次训练

1) 设置神经网络中各层的权值初值

设置神经网络中各层的权值初始值(实际算法中通常采用随机生成权值初始值的方法),如表 3.13 所示,卷积层共享一个偏置权值参数 Bias; 全连接层有两个神经元,每个神经元有一个代替偏置值的权值和 4 个输入权值。

表 3.13　训练参数数据

训　练	卷积核参数	全连接层参数
初始	$\text{Weight}=\begin{bmatrix}1&1\\1&1\end{bmatrix}$ $\text{Bias}=[1]$	$\text{Weight}=\begin{bmatrix}1&1&1&1\\1&1&1&1\end{bmatrix}$ $\text{Bias}=[1\quad 1]$

2) 训练过程的前向计算

图 3.44 中标识了字符 0 的训练图像在前向传播计算过程中的部分结果,下面详细介绍计算过程。

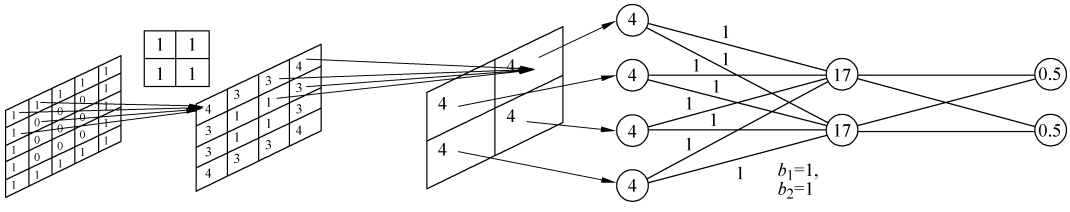


图 3.44　字符 0 的训练图像的前向传播计算

(1) 卷积层输出计算。

根据输入图像,最左上角的 4 个点值为 $\begin{bmatrix}1&1\\1&0\end{bmatrix}$,卷积核参数为 $\begin{bmatrix}1&1\\1&1\end{bmatrix}$,根据点乘计算方法,输出值为 $1\times1+1\times1+1\times1+1\times0=3$; 再加上偏置值 1,输出点值为 4。同理可计算出输出特征图各个点的值,如下所示:

$$\begin{bmatrix}1&1&1&1&1\\1&0&0&0&1\\1&0&0&0&1\\1&0&0&0&1\\1&1&1&1&1\end{bmatrix} * \begin{bmatrix}1&1\\1&1\end{bmatrix} + \text{Bias} = \begin{bmatrix}4&3&3&4\\3&1&1&3\\3&1&1&3\\4&3&3&4\end{bmatrix}$$

(2) 池化层输出计算。

采用 2×2 最大池化，最左上角 $\begin{bmatrix}4&3\\3&1\end{bmatrix}$ 的最大池化输出值为 4。同样，可计算出本层输出为 $\begin{bmatrix}4&4\\4&4\end{bmatrix}$。

(3) 全连接层输出计算。

对于第 1 个神经元，4 个权值参数均为 1，4 个输入均为 4，偏置为 1，采用 ReLU 作为激活函数，输出为 $y=17$，计算如下：

$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right) = f(4\times1+4\times1+4\times1+4\times1+1) = 17$$

同样可得，第二个神经元的输出也是 17。

(4) softmax 层输出计算。

根据 softmax 计算公式可得 $y_1=0.5$，计算如下：

$$y_1 = \frac{e^{a_1}}{\sum_{i=1}^{K} e^{a_i}} = \frac{e^{17}}{e^{17}+e^{17}} = 0.5$$

同样计算得到 softmax 输出为[0.5,0.5]。

(5) 本次训练输出的误差采用交叉熵评估函数计算如下：

$$E = -\left(\sum_{i=1}^{K}(y_i \ln(a_i) + (1-y_i)\ln(1-a_i))\right) = -(\ln(0.5)) = 0.6931$$

这里采用批训练，输入两张图像的计算中间过程的数据如表 3.14 所示。

表 3.14 训练结果数据(第 1 次)

训练	输入	卷积激活结果	池化输出	全连接输出	softmax 输出
1	图像 0	$\begin{bmatrix}4&3&3&4\\3&1&1&3\\3&1&1&3\\4&3&3&4\end{bmatrix}$	$\begin{bmatrix}4&4\\4&4\end{bmatrix}$	[17 17]	[0.5 0.5]
	图像 1	$\begin{bmatrix}3&5&3&1\\3&5&3&1\\3&5&3&1\\3&5&4&2\end{bmatrix}$	$\begin{bmatrix}5&3\\5&4\end{bmatrix}$	[18 18]	[0.5 0.5]

因为两个训练样本图像的误差均为 0.6931，因此总平均误差为 0.6931。

(6) 采用 BP 算法进行误差反向传播，调整各层参数。

整个神经网络只在卷积层和全连接层有权值参数，采用 BP 算法进行误差反向传播，调

整这些权值参数(计算过程略),调整后的参数如表 3.15 所示。

表 3.15　训练参数数据(第 1 次调整)

训练调整	卷积核参数	全连接层参数
1	Weight= $\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$ Bias= [1]	Weight= $\begin{bmatrix}0.95 & 1.05 & 0.95 & 1\\ 1.05 & 0.95 & 1.05 & 1\end{bmatrix}$ Bias= [1　1]

4. 第 2 次训练

根据第 1 次训练调整后的权值参数,再次进行训练,计算过程的数据如表 3.16 所示。

表 3.16　训练结果数据(第 2 次)

训练	输入	卷积激活结果	池化输出	全连接输出	softmax 输出
2	图像 0	$\begin{bmatrix}4 & 3 & 3 & 4\\ 3 & 1 & 1 & 3\\ 3 & 1 & 1 & 3\\ 4 & 3 & 3 & 4\end{bmatrix}$	$\begin{bmatrix}4 & 4\\ 4 & 4\end{bmatrix}$	[16.8　17.2]	[0.4013　0.5987]
	图像 1	$\begin{bmatrix}3 & 5 & 3 & 1\\ 3 & 5 & 3 & 1\\ 3 & 5 & 3 & 1\\ 3 & 5 & 4 & 2\end{bmatrix}$	$\begin{bmatrix}5 & 3\\ 5 & 4\end{bmatrix}$	[17.65　18.35]	[0.3318　0.6682]

计算总平均误差为 0.6679。采用 BP 算法进行误差反向传播,调整各层参数,调整后的参数如表 3.17 所示。

表 3.17　训练参数数据(第 2 次调整)

训练调整	卷积核参数	全连接层参数
2	Weight= $\begin{bmatrix}0.9628 & 1.0372\\ 0.9628 & 1.0372\end{bmatrix}$ Bias= [0.9628]	Weight= $\begin{bmatrix}0.9377 & 1.0982 & 0.9377 & 1.0372\\ 1.0623 & 0.9018 & 1.0623 & 0.9628\end{bmatrix}$ Bias= [1.0372　0.9628]

5. 第 3 次训练

根据第 2 次训练调整后的权值参数,再次进行训练,计算过程的数据如表 3.18 所示。

表 3.18　训练结果数据(第 3 次)

训练	输入	卷积激活结果	池 化 输 出
3	图像 0	$\begin{bmatrix}3.9256 & 2.96283 & 2.9628 & 4\\ 2.8884 & 0.9628 & 0.9628 & 3.0372\\ 2.8884 & 0.9628 & 0.9628 & 3.0372\\ 3.9256 & 2.9628 & 2.9628 & 4\end{bmatrix}$	$\begin{bmatrix}3.9256 & 4\\ 3.9256 & 4\end{bmatrix}$
	图像 1	$\begin{bmatrix}3.0372 & 4.9628 & 2.8884 & 0.9628\\ 3.0372 & 4.9628 & 2.8884 & 0.9628\\ 3.0372 & 4.9628 & 2.8884 & 0.9628\\ 3.0372 & 4.9628 & 3.9256 & 1.9256\end{bmatrix}$	$\begin{bmatrix}4.9628 & 2.8884\\ 4.9628 & 3.9256\end{bmatrix}$

续表

训练	输入	全连接输出	softmax 输出
3	图像 0	[16.9412 16.7612]	[0.5449 0.4551]
	图像 1	[17.5884 17.8907]	[0.4250 0.5750]

计算总误差为 0.6351。采用 BP 算法进行误差反向传播，调整各层参数，调整后的参数如表 3.19 所示。

表 3.19 训练参数数据(第 3 次调整)

训练调整	卷积核参数	全连接层参数
3	Weight= $\begin{bmatrix}0.9219 & 1.0777\\ 0.9200 & 1.0786\end{bmatrix}$ Bias= [0.9362]	Weight= $\begin{bmatrix}0.9155 & 1.1467 & 0.9155 & 1.0750\\ 1.0845 & 0.8533 & 1.0845 & 0.9250\end{bmatrix}$ Bias= [1.0700 0.9300]

6. 第 20 次训练

根据前面训练调整后的权值参数，进行 20 次训练后，得到的权值参数如表 3.20 所示。

表 3.20 训练结果数据(第 20 次)

训练	输入	卷积激活结果	池化输出
20	图像 0	$\begin{bmatrix}3.0094 & 2.7924 & 2.7924 & 4.5758\\ 1.2266 & 0.7971 & 0.7971 & 4.3632\\ 1.2266 & 0.7971 & 0.7971 & 4.3632\\ 3.0099 & 2.7974 & 2.7974 & 4.5802\end{bmatrix}$	$\begin{bmatrix}3.0094 & 4.5758\\ 3.0099 & 4.5802\end{bmatrix}$
	图像 1	$\begin{bmatrix}4.3632 & 4.7927 & 1.2266 & 0.7971\\ 4.3632 & 4.7927 & 1.2266 & 0.7971\\ 4.3632 & 4.7927 & 1.2266 & 0.7971\\ 4.3632 & 4.7927 & 3.0099 & 1.0141\end{bmatrix}$	$\begin{bmatrix}4.7927 & 1.2266\\ 4.7927 & 3.0099\end{bmatrix}$

训练	输入	全连接输出	softmax 输出
20	图像 0	[18.8222 13.4685]	[0.9950 0.0050]
	图像 1	[12.1052 17.5389]	[0.0043 0.9957]

计算总误差为 0.3158。采用 BP 算法进行误差反向传播，调整各层参数，调整后的参数如表 3.21 所示。

表 3.21 训练参数数据(第 20 次调整)

训练调整	卷积核参数	全连接层参数
20	Weight= $\begin{bmatrix}0.1819 & 1.8122\\ 0.1819 & 1.8125\end{bmatrix}$ Bias= [0.7976]	Weight= $\begin{bmatrix}0.4155 & 1.7915 & 0.4161 & 1.5049\\ 1.5845 & 0.2085 & 1.5839 & 0.4951\end{bmatrix}$ Bias= [1.3170 0.6830]

从上述计算过程可以看出，损失的误差随着训练次数的增加在不断减小。如果使用更多的图像，进行更多次的训练，就能得到一个很好的、鉴别图像是不是数字 1 的分类器。

7. 使用训练好的 CNN 网络

当输入如表 3.22 所示的字符 0 的图像进行识别时，将图像的矩阵数据输入前面训练好的网络模型中(参数值如表 3.21 所示)，可以计算得到如表 3.23 所示的结果。

表 3.22　待识别图像“0”(测试数据)

图像	存储的值矩阵
	$\begin{bmatrix}0&1&1&1&0\\0&1&0&1&0\\0&1&0&1&0\\0&1&1&1&0\\0&0&0&0&0\end{bmatrix}$

表 3.23　测试结果

输 入 数 据	卷积激活结果	池 化 输 出
$\begin{bmatrix}0&1&1&1&0\\0&1&0&1&0\\0&1&0&1&0\\0&1&1&1&0\\0&0&0&0&0\end{bmatrix}$	$\begin{bmatrix}4.4223&2.9799&4.6042&1.1677\\4.4223&1.1677&4.4223&1.1677\\4.4223&2.9801&4.6104&1.1677\\2.6098&2.7917&2.7917&0.9795\end{bmatrix}$	$\begin{bmatrix}4.4223&4.6042\\4.4223&4.6104\end{bmatrix}$
全连接输出	softmax 输出	
[20.1814　17.9369]	[0.9042　0.0958]	

根据 softmax 输出结果，图像为字符 0 的概率为 0.9042，大于图像为 1 的概率，则分类结果为字符 0，分类正确。

当输入如表 3.24 所示的字符 1 图像进行识别时，将图像的矩阵数据输入前面训练好的网络模型中，可以计算得到如表 3.25 所示的结果。

表 3.24　待识别图像“1”(测试数据)

图像	存储的值矩阵
	$\begin{bmatrix}0&1&0&0&0\\0&1&0&0&0\\0&1&0&0&0\\0&1&0&0&0\\0&0&0&0&0\end{bmatrix}$

表 3.25 测试结果

原始数据	卷积激活结果	池化输出
图像 0	$\begin{bmatrix} 4.4223 & 1.1677 & 0.7976 & 0.7976 \\ 4.4223 & 1.1677 & 0.7976 & 0.7976 \\ 4.4223 & 1.1677 & 0.7976 & 0.7976 \\ 2.6098 & 0.9795 & 0.7976 & 0.7976 \end{bmatrix}$	$\begin{bmatrix} 4.4223 & 0.7976 \\ 4.4223 & 0.7976 \end{bmatrix}$

全连接输出	softmax 输出
[7.6237 15.2557]	[0.0005 0.9995]

根据 softmax 输出结果，图像为字符 1 的概率约为 0.9995，大于图像为字符 0 的概率，则分类结果为字符 1，分类正确。

3.5.6 经典卷积神经网络 LeNet-5 模型

Yann LeCun 在 1998 年提出的用于文字识别的 LeNet-5 模型是非常经典的模型，它是第一个成功大规模应用的卷积神经网络，在 MNIST 数据集中的正确率可以高达 99.2%，其网络结构如图 3.45 所示。

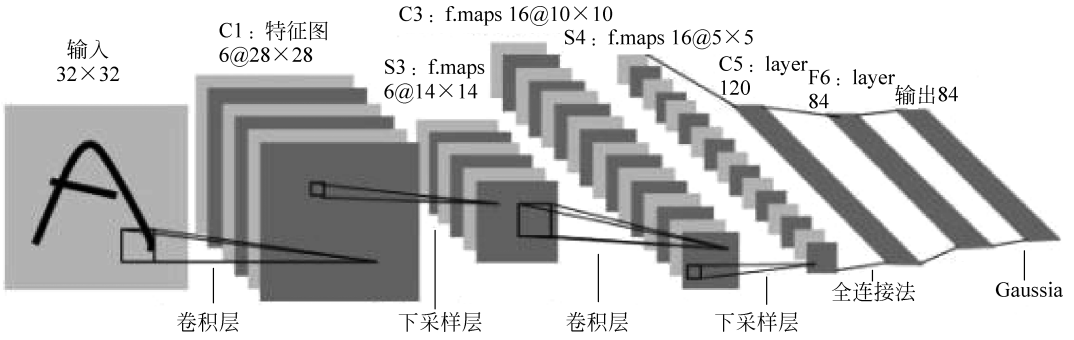


图 3.45 LeNet-5 网络结构[6]

LeNet-5 卷积神经网络模型一共有 7 层，包含卷积层、池化层（下采样层）、全连接层等。首先需要把含手写字符的原始图像处理成包含 32×32 个像素点的图像，作为输入；后面的神经网络层采用卷积层和池化层交替分布的方式。第 1 层（C1）是卷积层，分别采用了 6 个不同的卷积核，每个卷积核的尺寸均为 5×5，对 32×32 的输入数据进行纵向、横向步长均为 1 的卷积计算，得到 6 个 28×28 的特征图，每个特征图中的 28×28 个神经元共享这 25 个卷积核权值参数。通过卷积运算，原始信号的特征增强，同时也降低了噪声，不同的卷积核能够提取到图像中的不同特征。第 2 层（S2）是一个 2×2 的池化层，对 6 个特征图分别进行池化，得到 6 个 14×14 的特征图。第 3 层（C3）又是一个卷积层，这次采用了 16 个 5×5 的卷积核，得到 16 个 10×10 的特征图，而且本层产生不同特征图数据的每个神经元并不是和 S2 层中的所有 6 个特征图连接，而是只连接其中某几个特征图，这样可以让不同的特征图抽取出不同的局部特征。第 4 层（S4）是池化层，同样采用 2×2 的池化，对 16 个 C3 层的

特征图处理，得到16个5×5的特征图。第5层(C5)是一个包含120个神经元的卷积层，采用5×5的卷积核；因为S4层的特征图是5×5大小，每个特征图与卷积核运算得到一个数值，将每个特征图与120个神经元进行全连接，即每个神经元有5×5×16个连接。第6层(F6)则包含84个神经元，与C5层进行全连接，每个神经元经过激活函数、产生数据，输出给最后一层。因为是对10个数字字符进行识别，最后一层设置了10个神经元来获得分类结果，每个神经元的输出对应输入为某一个数字字符的概率。

3.6　深度学习的RNN模型介绍

在前面介绍的多层神经网络和卷积神经网络中，样本之间也没有明确的顺序关系，每个样本被独立处理，没有对数据本身存在的先后顺序关系进行建模。循环神经网络(Recurrent Neural Network，RNN)则能够在处理时对样本的先后顺序进行建模。

RNN中神经元的输出数据可以在下一个运算中又作为自身输入数据的一部分参与运算，即该神经元t时刻的输出是其t时刻的外部输入和其$t-1$时刻的输出共同作用的结果。RNN在实践中被证明能有效完成自然语言(词与词之间存在顺序关系)等多个领域的任务，当前被广泛应用。例如用于机器翻译(Machine Translation)，实现将中文语句转换成语义相同的英文语句；用于文本情感分析(Emotion Analysis)，对输入文本的情感极性进行分类，广泛应用于电商评论的情感分析和大众舆情分析；用于自动问答(Auto Q&A)，针对用户的问题，根据文档找出相应的答案；用于语音识别(Speech Recognition)，根据给出的一段声音信号，翻译出该语音对应的某种语言语句。

3.6.1　基本RNN的网络结构和工作过程

在RNN中，一般采用tanh函数作为激活函数。tanh函数定义如式(3.15)所示。

$$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3.15}$$

其函数图像如图3.46所示。观察图像可知，tanh激活函数的特点在于可以将数值压缩到(−1，1)之间，从而可以调节网络中神经元输出值的范围，防止数值过大影响后续的计算。

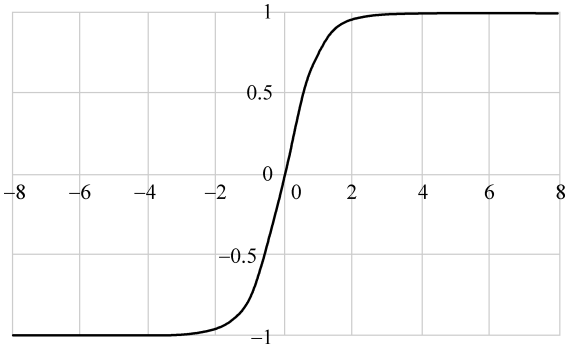


图3.46　tanh激活函数

RNN 的基本结构和工作过程如图 3.47 所示，图中箭头左边的部分是 RNN 的基本结构图，箭头右边的部分为 RNN 按每次输入数据展开的工作过程状态图。基本结构中，该神经网络只包含一个神经元，该神经元的结构和工作方式与感知机神经元类似，包含输入数据 X（可以是单个数值，也可以是一组数据）、权值参数 W、求和计算、激活函数；但是其输出数据 Y 会反馈回去，作为下一次神经元运行的输入数据。

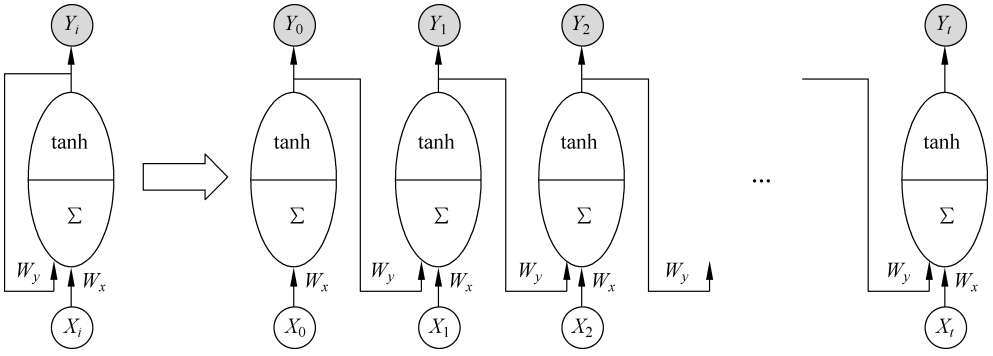


图 3.47　RNN 基本结构

RNN 的运行过程为：第一次（称为 T_0 时刻）输入的数据（用 X_0 表示，单个数值或一组数据），与权值 W_x 相乘，然后求和（通常每个神经元也包含一个偏置），对求和结果采用激活函数处理，产生输出 Y_0；该输出 Y_0 既作为网络输出向后传递，又反馈回到输入端，与第二次（称为 T_1 时刻）输入的数据 X_1 一起作为该神经元的输入。对于反馈回来的 Y_0，有一个权值参数 W_y，而对于第二次输入数据仍然使用原来的权值 W_x，神经元进行同样的求和和激活，然后产生输出 Y_1；同样过程继续运行，直到输入结束。

RNN 按照其输入、输出的对应关系，划分为以下 4 种类型，如图 3.48 所示，图中每个类型的底层为输入数据，中间为神经元，顶层为输出数据，用阴影填充的输入和输出为无实际意义的数据。

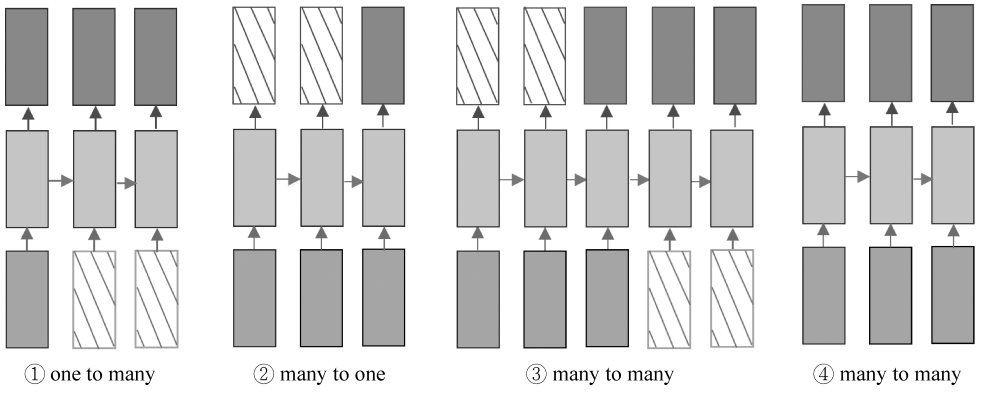


图 3.48　RNN 的类型

① one to many：输入单个值，输出序列。例如输入一个主旨词，输出一段描述该主旨的文本。

② many to one：输入序列，输出单个值。例如情感分析任务，输入一句话，返回其情感极性。

③ many to many：输入、输出均为序列，但输入的有效数据与输出的有效数据数量不一致。例如机器翻译任务，输入一句话，翻译为另外一种语言的一句话，两句话的词语数量很可能不一样多。

④ many to many：另一种输入、输出均为序列的情况，输入的有效数据与输出的有效数据数量一致。例如实时语音识别任务。

当用这样的 RNN 模型通过上下文来预测下一个单词时，如果预测的内容和关键信息之间的距离较近，RNN 可以较好地利用前文的信息实现预测，例如输入“24 hours in a (　　)”这句话，RNN 模型可以很轻松地预测出下一个单词为(day)。但是，当预测内容和关键信息间的距离较远时，RNN 模型就很难掌握这种长距离的信息。例如输入“My motherland is China，…，the capital is(　　)”，希望预测出(Beijing)，这个文本比较长，提示信息与当前要预测的单词距离较远，利用 RNN 模型进行预测的准确率就比较低。

3.6.2　LSTM 的结构和工作过程

传统的 RNN 由于只反馈当前输出作为下一次的输入，导致对前面一些时刻输入的信息记忆较短(更前面一些时刻输入的信息留存在当前输出中的部分比较少)。塞普·霍克赖特(Sepp Hochreiter)和于尔根·施米德胡贝(Jürgen Schmidhuber)于 1997 年提出了长短时记忆网络(Long Short Term Memory networks，LSTM)，旨在解决信息的长时期(长距离)依赖问题(The Problem of Long-Term Dependencies)。

LSTM 的核心思想：是神经元不是简单把当前输入和上一次反馈回来的结果一起处理之后就输出，而是模仿人的记忆过程，每次接收到新数据之后，对原来相关的记忆信息进行更新，然后视情况对更新后的记忆信息按一定比例输出；下一次有信息到来时，再同样地更新记忆和输出记忆信息。

LSTM 网络的基本单元不是一个简单的神经元，而是一个包含多组神经元的、称为细胞(cell)的结构，如图 3.49 所示。为了实现记忆更新和输出需要的各种控制比例，LSTM 细胞的内部结构较为复杂，通过不同组神经元来计算各种数据，并设置内部循环的细胞状态。在 t 时刻，LSTM 细胞反馈到下一次计算的除了当前的输出向量 Y_t(一个向量中包含多个数值)，还增加了反馈传递当前的状态向量 C_t；上一次的输出 Y_{t-1} 反馈传递回来，和当前输入向量 X_t 拼接在一起，构成 t 时刻的输入。为了更清楚地表示，图 3.49 中将这两部分输入数据分开画线(计算的时候会将两部分输入拼接在一个向量中)，因此对这两部分输入对应的权值参数也分开标记，例如分别用 W_{fy} 和 W_{fx} 分开标记，公式中则直接用 W_f 表示对所有输入数据的权值向量。

LSTM 细胞内部有 5 个实现不同功能的结构，主要分为 3 类：“门结构”(一共 3 个控制门)、“候选信息生成”结构和“输出信息生成”结构。3 个控制门和“候选信息生成”结构内部均包含有数量与状态向量 C_t 维度相等的神经元，且每个神经元都使用 Y_{t-1} 和 X_t 作为输入，但使用不同的权值参数。门结构用 Sigmoid(图中用 σ 表示)作为激活函数，输出值范围为(0，1)，用于控制信息向前传输的比例；信息生成结构采用 tanh 作为激活函数，取值范围为(−1，1)。

LSTM 细胞内部的三个控制门分别是：

遗忘门——控制原来记忆的信息(状态)保留多少；

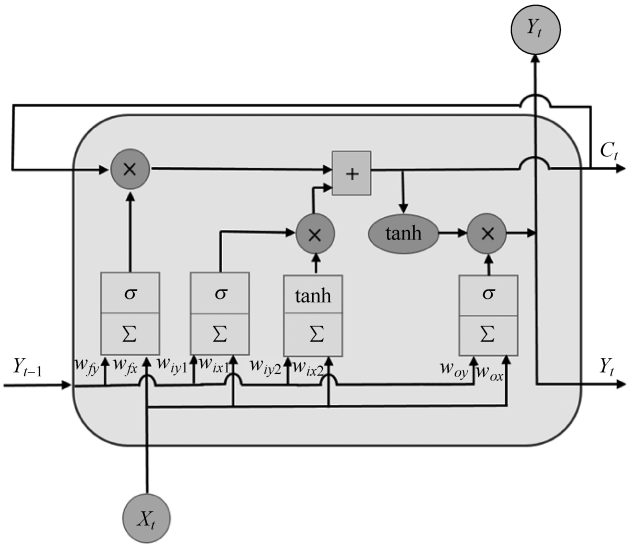


图 3.49 LSTM 的基本单元——细胞结构

输入门——控制新产生的信息有多少能进入到记忆(状态)中；

输出门——控制经过处理后的当前记忆信息(状态)按多大比例输出。

整个 LSTM 网络可以由多个这种细胞连接构成。下面对 LSTM 细胞的内部结构按工作过程进行介绍。

(1) LSTM 细胞的状态(cell state)，记为 C_t。

如图 3.50 所示，细胞状态 C_t 可以理解为在 t 时刻该 LSTM 细胞记住的信息，将反馈回去构成下一时刻细胞状态的一部分；同时乘以一个控制比例，输出给下一层 LSTM 细胞，并反馈回本细胞作为下一次输入数据的一部分，记为 Y_t。由于细胞状态不输出给其他 LSTM 细胞，只是在本细胞内部循环，因此也被称为隐藏信息或隐藏状态。

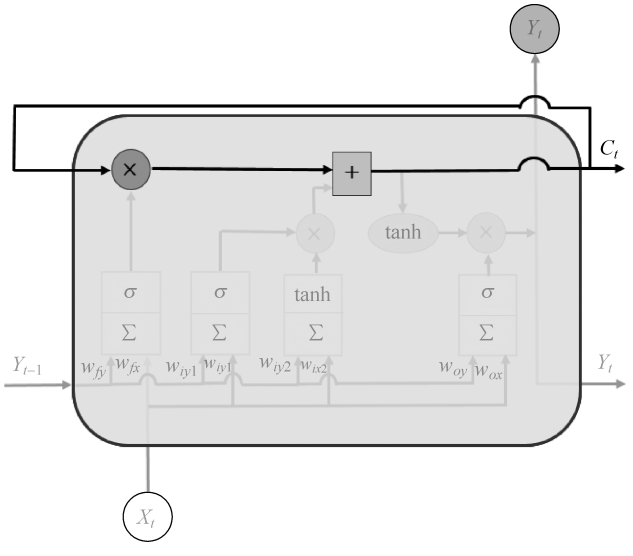


图 3.50 细胞状态(记忆的信息)

(2) 遗忘门。

每次输入数据到来时，LSTM 会通过“遗忘门”来控制保留多少此前时刻记住的信息，也即遗忘掉多少比例的信息。遗忘门通常包含多个神经元，每个神经元都对上次输出和本次新输入数据进行加权求和，使用 Sigmoid 作为激活函数，产生一个(0,1)之间的数值，表示模型需要记住或者遗忘其对应的原记忆信息的比例。Sigmoid 值为 0 时，表示这部分信息全部遗忘，Sigmoid 值为 1 时表示信息全部保存下来。

如图 3.51 所示，遗忘门的输入向量为前一时刻输出反馈回来的 Y_{t-1} 和当前的输入向量 X_t，将这两部分数据送入带有 Sigmoid 激活函数的神经元组处理，得到比例向量 f_t，从而让模型能自行判断对之前时刻的记忆需要遗忘多少，其计算如式(3.16)所示。

$$f_t=\sigma(W_f[Y_{t-1},X_t]+b_f) \tag{3.16}$$

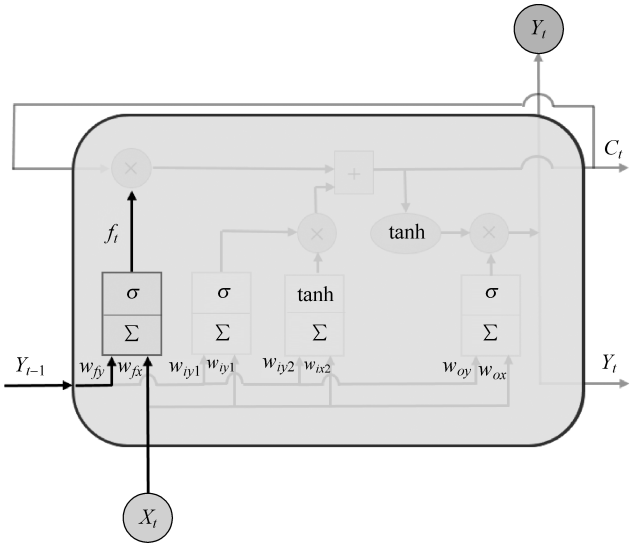


图 3.51　遗忘门内部结构

(3) 候选信息生成。

如图 3.52 所示，将前一时刻输出端反馈回来的信息 Y_{t-1} 和当前的输入向量 X_t 加权求和后，采用 tanh 激活函数生成一个新的候选信息 $\widetilde{C}_t$，用来更新记忆。

(4) 输入门。

输入门用来控制新产生的候选信息有多少被保留下来。如图 3.52 所示，它也是将前一时刻反馈回来的信息 Y_{t-1} 和当前的输入向量 X_t 采用不同的权值参数进行加权求和后，用 Sigmoid 激活函数输出，得到输入控制向量 i_t，用于决定候选信息 $\widetilde{C}_t$ 是否重要，需要将它的多大比例保留到记忆中。i_t 和 $\widetilde{C}_t$ 的计算公式分别如式(3.17)和(3.18)所示。

$$i_t=\sigma(W_i[Y_{t-1},X_t]+b_i) \tag{3.17}$$

$$\widetilde{C}_t=\tanh(W_C[Y_{t-1},X_t]+b_C) \tag{3.18}$$

之后将对细胞状态进行更新，如图 3.53 所示。使用遗忘门得到的 f_t 和前一时刻的细胞状态 C_{t-1} 相乘来得到需要保留的记忆部分，将输入门产生的比例 i_t 与新的候选值 $\widetilde{C}_t$ 相

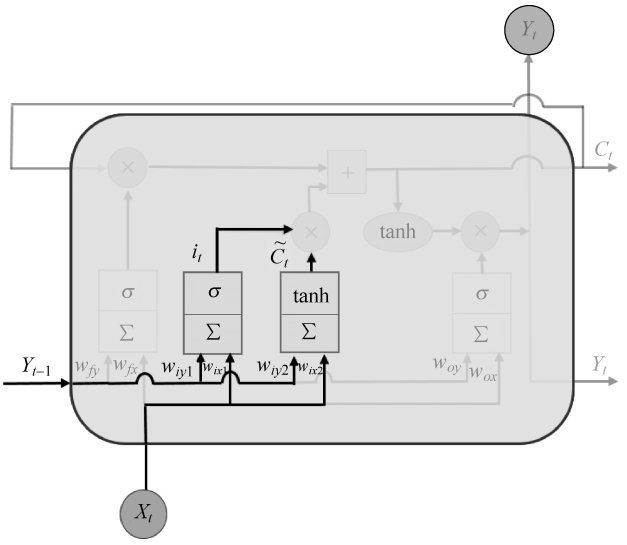


图 3.52 输入门内部结构

乘来得到 $\widetilde{C}_t$ 中应该进入记忆的部分，两者相加得到新的记忆信息 C_t（细胞状态）。其计算如式(3.19)所示。

$$C_t = f_t C_{t-1} + i_t \widetilde{C}_t \tag{3.19}$$

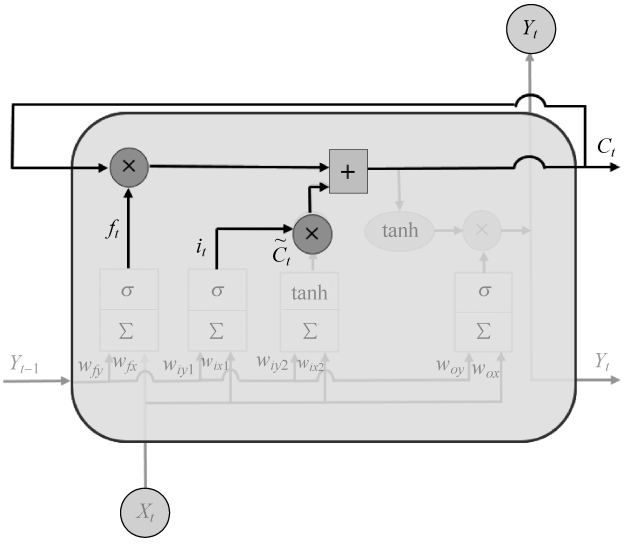


图 3.53 细胞状态更新

(5) 输出门。

记忆信息并不直接输出，而是采用 tanh 激活函数规范到(−1,1)之间，作为准备输出的信息。但实际输出多少还需要使用输出门来产生一个 0 到 1 之间的比例来控制。如图 3.54 所示，同样是将前一时刻输出端反馈回来的信息 Y_{t-1} 和当前的输入向量 X_t 加权求和(采用与其他神经元不同的权值参数)，然后用 Sigmoid 激活函数产生输出控制向量

O_t，如式(3.20)所示。将上述两部分相乘来最终确定当前神经元的输出 Y_t。计算过程如式(3.21)所示。

$$o_t = \sigma(W_o[Y_{t-1}, X_t] + b_o) \tag{3.20}$$

$$Y_t = o_t \cdot \tanh(C_t) \tag{3.21}$$

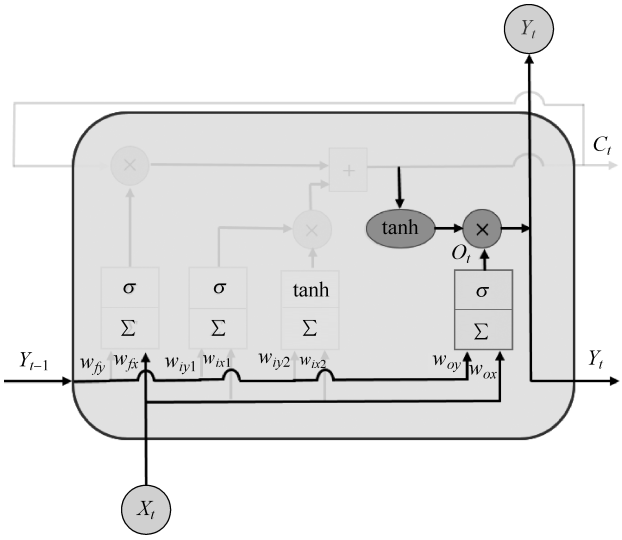


图 3.54　输出门内部结构

由于 LSTM 的神经元可以通过自动学习来调节各个门产生的控制比例，因此即使是较早时刻产生的重要信息，也能够通过状态来保存，并传递到其后比较远的时刻，从而在结构上克服了传统 RNN 带来的“长距离依赖问题”。

参考文献

[1] 高文. 人工智能——螺旋上升的 60 年. CNCC2016 大会特邀报告[EB/OL]. (2016-10-22)[2020-05-27]. http://www.360doc.com/content/16/1022/22/11548039_600591144.shtml.

[2] 张玉宏. 深度学习之美：AI 时代的数据处理与最佳实践[M]. 北京：电子工业出版社，2018.

[3] 李德毅，于剑. 人工智能导论[M]. 北京：中国科学技术出版社，2018.

[4] 韩力群. 人工神经网络理论、设计及应用[M]. 2 版. 北京：化学工业出版社，2007.

[5] 周志华. 机器学习[M]. 北京：清华大学出版社，2016.

[6] LECUN Y，BOTTOU L. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE，1998，86(11)：2278-2324.

扩展阅读

[1] 张觉非. 深入理解神经网络 从逻辑回归到 CNN[M]. 北京：人民邮电出版社，2019.

[2] 江永红. 深入浅出人工神经网络[M]. 北京：人民邮电出版社，2019.

[3] 山下隆义. 图解深度学习[M]. 张弥，译. 北京：人民邮电出版社，2018.

[4] 深度学习中文社区[EB/OL]. [2020-05-31]. http://dl.tustcs.com/.

习题 3

一、单项选择题

1. 以下关于人工神经网络的描述正确的是(　　)。
   A. 任何两个神经元之间都是有连接的
   B. 前馈神经网络(FNN)是带有反馈的人工神经网络
   C. 带反馈的人工神经网络比不带反馈的人工神经网络高级
   D. 神经元的激活函数具有多种形式,不同的激活函数得到的性能不同
2. 人工神经网络的层数增加会导致梯度消失现象,其本质原因是(　　)。
   A. 各层误差梯度相加　　B. 各层误差梯度相减
   C. 各层误差梯度相乘　　D. 误差趋于饱和
3. 以下关于深度学习的描述不正确的是(　　)。
   A. 深度神经网络的层数多于浅层神经网络,具有更强的表达能力
   B. 卷积神经网络可以不需要人工提取特征参数
   C. 深度学习是大数据时代的必然产物
   D. 以上都不正确
4. 以下关于感知器的说法错误的是(　　)。
   A. 单层感知器可以解决异或问题
   B. 感知器分类的原理是通过调整权重使两类样本经过感知机模型后的输出不同
   C. 单层感知器只能针对线性可分的数据集分类
   D. 学习率可以控制每次权值调整力度

二、多项选择题

1. 卷积神经网络的结构主要包括(　　)。
   A. 卷积层　　B. 池化层　　C. 全连接层　　D. 输入层
2. 多层神经网络主要包括(　　)。
   A. 输入层　　B. 物理层　　C. 隐藏层　　D. 输出层
3. 人工神经网络由许多神经元构成,M-P 模型的主要特征包括(　　)。
   A. 多输入、单输出　　B. 对输入加权求和
   C. 具有树突和轴突　　D. 具有激活函数

三、判断题

1. 人工神经网络的层数是固定的,每层的神经元个数是不固定的。(　　)
2. BP 神经网络的误差是从前往后传播的。(　　)
3. 卷积神经网络的层数一般大于 3。(　　)

四、简答题

1. 感知机是如何实现从数据中学习的?
2. 什么是梯度?什么是梯度的方向?
3. 有 A 类和 B 类两类物品,均有两个类似的特征值。现有三个属于 A 类物品的样本,

每个样本的特征值分别为[0.1,1]、[0.2,0.7]、[0.4,0.8],样本标签用1表示；有三个属于B类物品的样本,其特征值分别为[0.8,0.3]、[0.9,0.2]、[1.0,0.5],样本标签用0表示。现在要训练一个感知机,使其能将两类物品正确地区分,可以初始化感知机的三个权值参数分别为 $w_1=1$,$w_2=1$,$b=-1$(也可随机初始化),学习率为0.4。请在下表中写出这六个样本循环送入感知机进行训练时,对三个参数进行调整的情况。

调整参数	样本输入	样本预设类别值 y	网络输出 y_{out}	误差 ε	调整后的 w_1	调整后的 w_2	调整后的 b
初始					1	1	−1
1	(0.1,1)	1	1	0	1	1	−1
2	(0.2,0.7)	−1	1	2	1.08	1.28	−0.6
3	(0.4,0.8)	1					
4	(0.8,0.3)	0					
5	(0.9,0.2)	0					
6	(1.0,0.5)	0					

4. 对图像进行卷积运算,输入图像与卷积核如下所示,请给出按步长为1进行卷积后的特征图。

输入图像：

4	5	9	9	10
7	13	2	8	3
3	8	3	4	2
12	8	10	13	10
13	16	6	11	15

卷积核：

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

5. 对如下图像采用2×2的池化窗口进行步长与窗口大小相同的最大池化运算,请给出池化后的图像。

4	5	9	9
7	13	2	8
3	8	3	3
12	8	10	14