第3章

基于神经网络的智能控制

基于神经网络的智能控制又称为神经控制。神经网络是在联结机制上模拟人脑右半球形象思维和神经推理功能的神经计算模型。神经元是构成神经网络的最小单元，它在细胞水平上模拟人的智能。神经元模型、神经网络模型和学习算法构成了神经网络的三要素。本章首先介绍神经元结构、功能及模型，神经网络结构、控制和识别中常用神经网络模型及学习算法；然后讲述神经控制系统的结构、原理、分类；最后阐述基于神经网络的系统辨识及基于神经网络的智能控制原理及应用。

3.1 神经网络系统基础

3.1.1 神经网络研究概述

神经网络(Neural Network，NN)是指用计算机模拟人脑神经系统的网络结构和功能，进行信息处理的人工神经网络(Artificial Neural Network，ANN)的简称。

神经网络的研究过程经历了初创期(1943—1969 年)、成长期(1970—1986 年)、发展期(1987—2005 年)和高潮期(2006 年至今)。

1943 年，心理学家麦卡洛克(McCulloch)和数学家匹茨(Pitts)首先提出了形式神经元 MP 模型，把神经元作为双态开关，用布尔逻辑研究客观事件的形式神经网络模拟。1949 年，心理学家赫布(Hebb)提出了神经元学习规则，赋予了神经网络的可塑性。Hebb 规则为后来的多种神经网络学习规则奠定了基础。1958 年，罗森布拉特(Rosenblatt)提出了感知器模型，指出感知过程具有统计分离性，利用教师信号对感知器进行训练，试图模拟人脑感知能力和学习能力。1969 年，明斯基(Minsky)和帕伯特(Papert)出版了《感知器》一书，严格地论证了简单线性感知器功能的局限性，并指出多层感知器还缺乏有效的计算方法，致

使神经网络的研究陷于困境。

1970—1986 年，经过辛顿(Hinton)、科霍恩(Kohonen)、格罗斯伯格(Grossberg)、甘利教授等坚持长期不懈地潜心研究，使得神经网络的研究逐渐摆脱了困境。1982 年，美国物理学家霍普菲尔德(Hopfield)通过引入能量函数研究神经网络的动态特性，并给出了网络的稳定性判据。Hopfield 网络具有联想记忆和优化计算功能，这是神经网络研究史上具有里程碑意义的重要成果。1986 年，鲁姆尔哈特(Rumelhart)和帕克(Parker)又重新发现、提出了反向传播算法，至今仍被广泛使用，并对研究神经网络的学习和训练产生了深远影响。

1987 年，在美国召开了第一届世界神经网络会议，在世界范围内推动了神经网络研究工作的发展。美国国防部于 1988 年开始了一项投资数亿美元的发展神经网络及其应用研究的八年计划。此后，许多国家也制定了相应计划推动神经网络研究。1989 年，切本科(Cybenko)和霍尼克(Hornik)等证明了神经网络的万能逼近定理：三层神经网络能以任意精度逼近任何函数。这一成果为研究神经网络优异性能提供了强大的理论依据，有力地推动了神经网络研究工作的进一步发展。1998 年，乐村等发明的卷积神经网络突破了图像识别复杂问题。此后的近十多年里，多种浅层网络遭遇了反向传播算法存在梯度消失问题的困扰。

2006 年，辛顿等提出了深度信念网络及其快速训练法，开启了深度学习的先河。本吉奥(Bengio)等证明了预训练方法也适用于无监督学习，以及波尔特尼等用基于能量模型来有效学习稀疏表示，这些成果奠定了深度学习的基础。2010 年，辛顿教授及其学生成功地帮助微软、IBM 和谷歌等公司突破了语音识别、图像识别的关键技术。2011 年，格洛特(Glorot)等提出的 ReLU 激活函数有效抑制梯度消失问题。2014 年以来，多芬(Dauphin)、齐罗曼斯卡(Choromanska)等证明了局部极小值问题不再是严重问题，消除了笼罩在神经网络上局部极值的阴霾。2015 年以来，谷歌 DeepMind 团队研发的基于深度学习的计算机围棋 AlphaGo、AlphaGo Zero 多次战胜了国际围棋大师。由此，在世界范围把深度神经网络和深度学习的研究推向了高潮。

纵观神经网络研究的发展史，不难看出多伦多大学辛顿教授始终发挥着引领作用，并做出了巨大的贡献。因此，他和乐村、本吉奥教授 3 人成功获得 2019 年计算机领域的诺贝尔奖——图灵奖。在当今大数据、网络化、智能化，并以人工智能技术为引擎的智能时代，人工神经网络作为人工智能的核心技术在新技术革命中将发挥越来越大的作用。

3.1.2 神经细胞结构与功能

人的智能来源于大脑，大脑是由一百几十亿个神经细胞构成的。一个神经细胞由细胞体、树突和轴突组成，其结构如图 3.1 所示。细胞体由细胞核、细胞质和细胞膜组成。细胞体外面的一层膜称为细胞膜，厚为 5～10nm，膜内有一个细胞核和细胞质。树突是细胞体向外伸出的许多树枝状长 1mm 左右的突起，用于接收其他神经细胞传入的神经冲动。

神经细胞在结构上具有如下两个重要的特征。

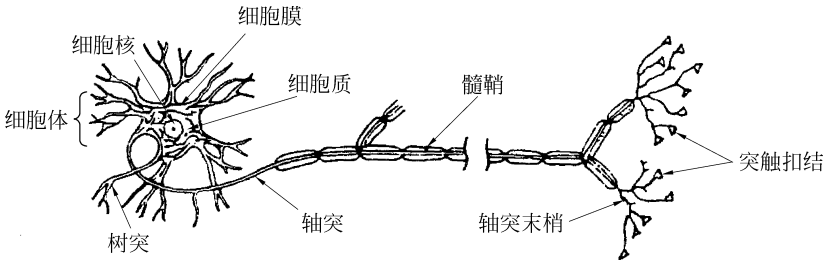


图 3.1 一个神经细胞的结构

1. 细胞膜具有选择的通透性

每个神经细胞用细胞膜和外部隔开,使细胞内、外有不同的电位。把没有输入信号的膜电位叫静止膜电位,约为−70mV。当有输入信号时(即其他神经细胞传入的兴奋信号)使膜电位比静止膜电位高 15mV 左右时,该神经细胞兴奋,而激发出宽度为 1ms、幅值为 100mV 电脉冲。兴奋的神经细胞一经发出一次脉冲,膜电位下降到比静止膜电位更低,然后再慢慢返回原值,约在几毫秒内神经细胞处于疲劳状态,即使再强大的输入信号也不能使神经细胞兴奋。

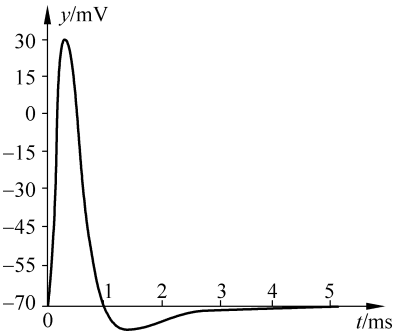


图 3.2 一个神经细胞兴奋发出的电脉冲

细胞膜具有选择的通透性使神经细胞具有阈值特性,如图 3.2 所示。

$$y=\begin{cases}\bar{y}, & u\geqslant\theta\\ 0, & u<\theta\end{cases} \tag{3.1}$$

其中,θ 是一个阈值,随着神经元的兴奋而变化,神经元兴奋时发出电脉冲具有突变性和饱和性。

2. 突触联结的可塑性

突触是指一个神经元轴突末梢和另一个神经元树突或细胞体之间微小的间隙,突触的直径为 0.5～2μm。突触起到了两个神经元之间传递信息的“接口”的作用。神经细胞之间通过突触相联结,这种联结强度根据输入和输出信号的强弱而可塑性变化。突触结合强度,即联结权重 w 不是一定的,根据输入和输出信号的强弱可塑性的变化,使得神经元具有长期记忆和学习功能。

3.1.3 人工神经元模型

从信息处理的角度,神经元是一个多输入单输出的信息处理单元。一个人工神经元的

形式化结构模型如图 3.3 所示，其中 $x_1, x_2, \cdots, x_n$ 表示来自其他神经元轴突的输入信号，$w_1, w_2, \cdots, w_n$ 分别为其他神经元与第 i 个神经元的突触联结强度，θ_i 为神经元 i 的兴奋阈值。

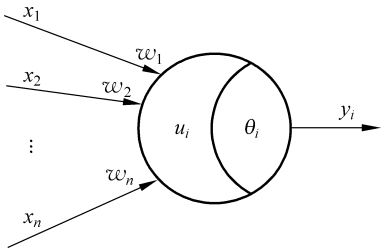


图 3.3　人工神经元的形式化结构模型

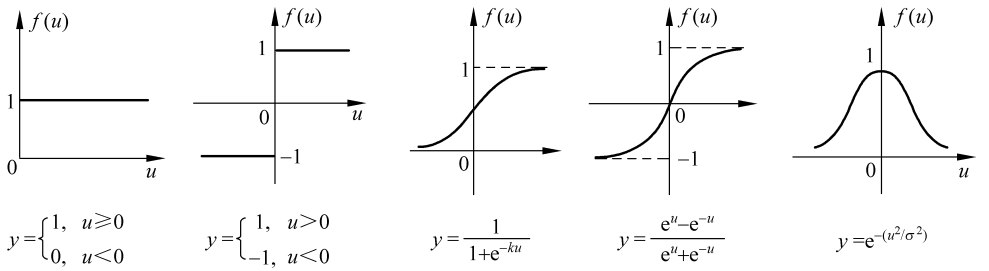
对于每个神经元信息处理过程可描述为

$$S_i = \sum_{j=1}^{n} w_j x_j - \theta_i \tag{3.2}$$

$$u_i = g(S_i) \tag{3.3}$$

$$y_i = f(u_i) \tag{3.4}$$

其中，S_i 表示神经元 i 的状态；u_i 表示神经元 i 膜电位的变化；y_i 表示神经元 i 的输出；$g(\cdot)$表示活性度函数；$f(\cdot)$为输出函数。输出函数又称为激活函数，如图 3.4 所示。



(a) 阶跃函数　(b) 符号函数　(c) S型函数　(d) 双曲正切　(e) 高斯函数

图 3.4　神经元常用的输出函数类型

上述输出函数均为非线性函数，都具有突变性和饱和性两个显著特征，这正是模拟神经细胞兴奋产生神经冲动性和疲劳特性。

3.1.4　神经网络的特点

大量的神经细胞通过突触联结成神经网络，而神经网络模型用于模拟人脑大量神经元活动的过程，其中包括对信息的加工、处理、存储和搜索等过程，它具有如下基本特点。

（1）具有分布式存储信息的特点。神经网络存储信息的方式与传统的计算机的思维方式是不同的，一个信息不是存储在一个地方，而是分布在不同的位置。网络的某一部分也不只存储一个信息，它的信息是分布式存储的。这种分布式存储方式即使当局部网络受损时，仍具有能够恢复原来信息的特点。因此，神经网络具有一定的容错能力。

（2）对信息的处理及推理具有并行的特点。每个神经元都可根据接收到的信息作独立的运算和处理，然后将结果传输出去，这体现了一种并行处理。神经网络对于一个特定的输

入模式,通过前向计算产生一个输出模式,各个输出结点代表的逻辑概念被同时计算出来。在输出模式中,通过输出结点的比较和本身信号的强弱而得到特定的解,同时排除其余的解。这体现了神经网络对信息并行推理的特点。

(3) 对信息的处理具有自组织、自学习的特点。神经网络各神经元之间的联结强度用权重大小来表示,这种权重可以事先定义,也可以为适应周围环境而不断地变化,这种过程体现了神经网络中神经元之间相互作用、相互协同、自组织的学习行为。神经网络所具有的自学习过程模拟了人的形象思维方法,这是与传统符号逻辑完全不同的一种非逻辑非语言的方法。

(4) 具有从输入到输出非常强的非线性映射能力。因为神经网络具有自组织、自学习的功能,所以通过学习它能实现从输入到输出的任意的非线性映射,即它能以任意精度逼近任意复杂的连续函数。

下面通过如图 3.5 所示的简单神经网络来说明人工神经网络的主要特点。

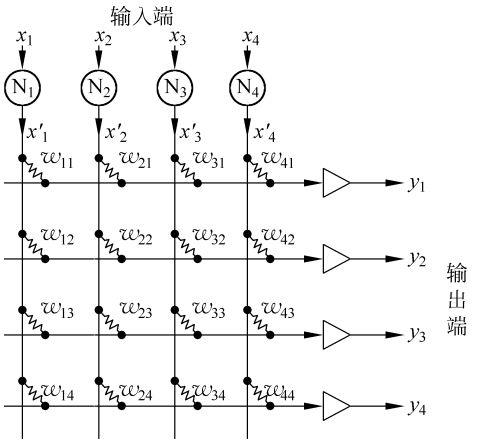


图 3.5 一个简单的神经网络结构

设 x_1,x_2,x_3,x_4 为神经网络输入,经神经元 N_1,N_2,N_3,N_4 的输出分别为 x'_1,x'_2,x'_3,x'_4,然后经过突触权 w_{ij} 连接到 y_1,y_2,y_3,y_4 的输入端,进行累加。

为简单起见,设 $\theta_i=0$,并将式(3.2)、式(3.3)、式(3.4)分别变为

$$S_i=\sum_{j=1}^{n}w_{ij}x'_j \tag{3.5}$$

$$u_i=S_i\cdot 1 \quad \text{(量纲变换)} \tag{3.6}$$

$$y_i=f(u_i)=\begin{cases}1, & u_i\geqslant 0\\ -1, & u_i<0\end{cases} \tag{3.7}$$

又设输入 $x'_j=\pm 1$ 为二值变量,且 $x'_j=x_j$,$j=1,2,3,4$。x_j 是感知器输入,用向量 $\boldsymbol{x}^1=(1,-1,-1,1)^{\mathrm{T}}$ 表示眼看到花,鼻子嗅到花香的感知输入,从 $\boldsymbol{x}^1$ 到 $\boldsymbol{y}^1$ 可通过一个连接矩阵

$$\boldsymbol{W}_1=\begin{bmatrix}-0.25 & +0.25 & +0.25 & -0.25\\ -0.25 & +0.25 & +0.25 & -0.25\\ +0.25 & -0.25 & -0.25 & +0.25\\ +0.25 & -0.25 & -0.25 & +0.25\end{bmatrix} \tag{3.8}$$

来得到。根据式(3.5)～式(3.7)可得

$$\boldsymbol{y}^1=f(\boldsymbol{W}_1\boldsymbol{x}^1)$$

经计算

$$\boldsymbol{y}^1=[-1,-1,+1,+1]^{\mathrm{T}}$$

这表示网络决策 $\boldsymbol{x}^1$ 为一朵花。

不难看出，从 $\boldsymbol{x}^1\rightarrow\boldsymbol{y}^1$ 不是串行计算得到的，而是并行计算得到的。因为 $\boldsymbol{W}_1$ 是可以用一个 VLSI 中电阻矩阵实现，而 $y_i=f(v_i)$ 也可以用一个简单运算放大器来模拟，不管 $\boldsymbol{x}^1$ 和 $\boldsymbol{y}^1$ 维数如何增加，整个计算只用了一个运算放大器的转换时间，显然网络的动作是并行的。

如果 $\boldsymbol{x}^2=[-1,+1,-1,+1]^{\mathrm{T}}$ 表示眼看到苹果、鼻嗅到苹果香味的感知器输入，通过矩阵

$$\boldsymbol{W}_2=\begin{bmatrix}+0.25 & -0.25 & +0.25 & -0.25\\ -0.25 & +0.25 & -0.25 & +0.25\\ -0.25 & +0.25 & -0.25 & +0.25\\ +0.25 & -0.25 & +0.25 & -0.25\end{bmatrix} \tag{3.9}$$

得到 $\boldsymbol{y}^2=[-1,+1,+1,-1]^{\mathrm{T}}$ 表示网络决策 $\boldsymbol{x}^2$ 为苹果。

从式(3.8)和式(3.9)的权矩阵来看，我们并不知道其输出结果是什么。从局部权的分布也很难看出 $\boldsymbol{W}$ 中存储了什么，这是因为信息是分布存储在权矩阵中，把式(3.8)和式(3.9)相加，得到一组新的权矩阵

$$\boldsymbol{W}=\boldsymbol{W}_1+\boldsymbol{W}_2=\begin{bmatrix}0 & 0 & 0.5 & -0.5\\ -0.5 & 0.5 & 0 & 0\\ 0 & 0 & -0.5 & 0.5\\ 0.5 & -0.5 & 0 & 0\end{bmatrix} \tag{3.10}$$

由 $\boldsymbol{x}^1$ 输入，通过权矩阵 $\boldsymbol{W}$ 运算可得到 $\boldsymbol{y}^1$，由 $\boldsymbol{x}^2$ 输入，通过权矩阵 $\boldsymbol{W}$ 运算可得到 $\boldsymbol{y}^2$，这说明 $\boldsymbol{W}$ 存储了两种信息，当然也可以存储多种信息。

如果感知器中某个元件损坏了一个，设第 3 个坏了，则 $\boldsymbol{x}^1=(1,-1,0,1)^{\mathrm{T}}$，经 $\boldsymbol{W}$ 算得 $\boldsymbol{y}^1=(-1,-1,+1,+1)^{\mathrm{T}}$，而 $\boldsymbol{x}^2=(-1,+1,0,1)^{\mathrm{T}}$，经 $\boldsymbol{W}$ 算得 $\boldsymbol{y}^2=(-1,1,1,-1)^{\mathrm{T}}$ 的结果和前面的一样，这说明人工神经网络具有一定的容错能力。

上面的例子已经表明，神经网络具有分布存储信息和对信息的处理及推理具有并行的特点。

有关人工神经网络的自学习功能和非线性映射能力将在后续的内容中加以研究。

3.1.5 神经网络结构模型

神经网络联结方式的拓扑结构是以神经元为结点、以结点间有向连接为边的一种图，其结构大体上可分为层状和网状两大类。层状结构的神经网络是由若干层组成，每层中有一定数量的神经元，相邻层中神经元单向联结，一般地同层内的神经元不能联结；网状结构的神经网络中，任何两个神经元之间都可能双向联结。下面介绍几种常见的神经网络结构。

1. 前馈神经网络

前馈网络包含输入层、隐层(一层或多层)和输出层，如图3.6所示为一个三层网络。这种网络特点是只有前后相邻两层之间神经元相互联结，各神经元之间没有反馈。每个神经元可以从前一层接收多个输入，并只有一个输出给下一层的各神经元。

2. 反馈神经网络

反馈网络指从输出层到输入层有反馈，即每一个结点同时接收外来输入和来自其他结点的反馈输入，其中也包括神经元输出信号引回到本身输入构成的自环反馈，如图3.7所示。

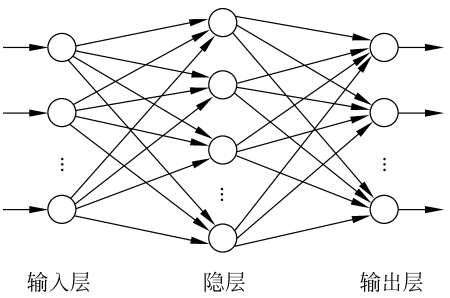


图3.6 前馈网络

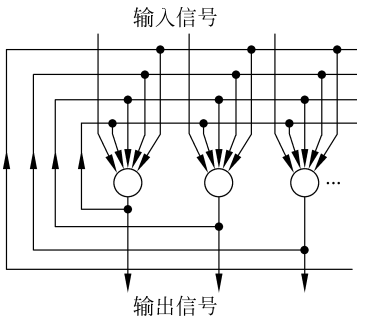


图3.7 反馈网络

3. 相互结合型神经网络

相互结合型网络属于网状结构，如图3.8所示。构成网络中各个神经元都可能相互双向联结，所有神经元既作输入也作输出。这种网络对信息处理与前馈网络不一样，如果在某一时刻从神经网络外部施加一个输入，各个神经元一边相互作用，一边进行信息处理，直到使网络所有神经元的活性度或输出值，收敛于某个平均值为止信息处理才结束。

4. 混合型神经网络

如图3.9所示，在前馈网络的同一层间神经元有互联的结构，称为混合型网络。这种在

同一层内的互联，目的是为了限制同层内神经元同时兴奋或抑制的神经元数目，以完成特定的功能。

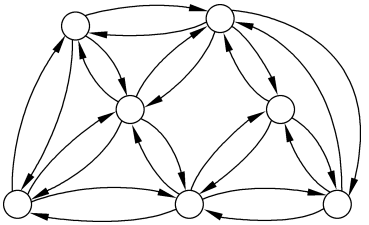


图 3.8　网状结构网络

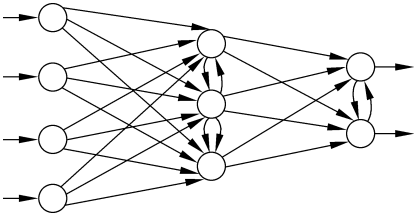


图 3.9　混合型网络

3.1.6　神经网络训练与学习

人脑中的神经元通过许多树突的精细结构，收集来自其他神经元的信息，并通过轴突发出电活性脉冲。轴突有上千条分支，在每条分支末端，通过突触的结构把来自轴突的电活性脉冲变为电作用，从而使与之相连的各种神经元的活性受到抑制或兴奋。

当一个神经元收到兴奋输入，而兴奋输入又比神经元的抑制输入足够大时，神经元把电活性脉冲向下传到它的轴突，改变轴突的有效性，从而使一个神经元对另一个神经元的影响改变，便产生了学习行为。因此，可以认为神经网络学习的本质特征在于神经细胞特殊的突触结构所具有的可塑性联结，而如何调整联结权重就构成了不同的学习算法。

神经网络按学习方式分为有教师学习和无教师学习两大类，如图 3.10 给出了这两种学习方式的直观示意。

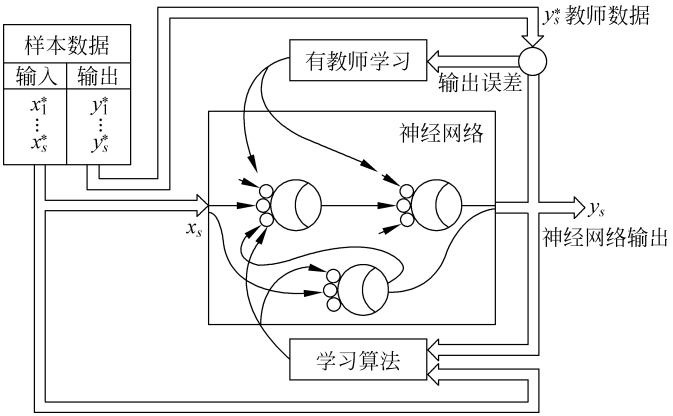


图 3.10　神经网络的训练与学习过程示意图

1. 神经网络的训练

为了应用神经网络解决工程实际问题,必须对它进行训练。从应用环境中选出一些输入输出样本数据,通过教师示教和监督来不断地调整神经元之间的联结强度,直到神经网络得到合适的输入输出关系为止,这个过程称为对神经网络的训练。这种有教师指导训练过程会使神经网络学习到隐含表示问题的知识,故又称这种方式为神经网络有教师学习,或监督学习。

如图3.10上半部分所示,在训练学习中教师提供样板数据集是成对的输入输出数据集$\{x_i^*, y_i^*\}$,它代表了实际问题输入输出关系。训练的过程就是根据网络输入的x_i^*和网络输出y_i^*的正误程度来反复调整权重的大小,直到网络的实际输出y_i^*全部等于期望的输出为止,训练过程便结束。

2. 神经网络的学习

神经网络的学习通常指神经网络无教师监督学习,如图3.10下半部分所示。神经网络学习的目的是根据实际输出数据和期望输出之间的误差,通过某种学习算法自动地、反复地去调整权重直到消除误差或小于允许的误差为止。

要使人工神经网络具有学习能力,需要使神经网络的知识结构变化,也就是使神经元间的结合模式变化,这同把联结权重用适当方法变化是等价的。因此,神经网络可以通过一定的学习算法实现对突触结合强度的调整,使其具有记忆、识别、分类、优化等信息处理功能。

3. 神经网络的泛化能力

神经网络通过样本数据集的训练后,当输入出现了样本数据集以外的新数据时,神经网络仍能通过学习获得新的输出,并能严格保持神经网络训练后所获得的输入输出映射关系的能力,称为神经网络的泛化能力。神经网络的结构、训练样本的数量和质量、不同的学习算法及其参数等都会影响神经网络的泛化能力。

4. 神经网络的生长与修剪

通过改变神经网络的结构和参数,从而改变了网络的规模大小,使之更适合于某个问题的求解,这样的过程称为神经网络的生长与修剪。例如对于前馈网络的生长算法,可以从单隐层的小网络开始,通过增加一个隐层重新训练,一直持续到再增加一个单元网络的性能不再改变为止。神经网络的修剪方法是从相对大的网络开始,然后逐渐剪去不必要的单元,直到获得满意的网络性能为止。

3.1.7　神经网络的学习规则

1. 联想式学习——Hebb 规则

1949 年，心理学家赫布(Hebb)提出了学习行为的突触联系和神经群理论，并认为突触前与突触后两个同时兴奋的神经元之间的联结强度将得到增强。后来研究者们将这一思想加以数学描述，被称为 Hebb 学习规则。如图 3.11 所示，从神经元 u_j 到神经元 u_i 的联结强度，即权重变化 ΔW_{ij} 表达为

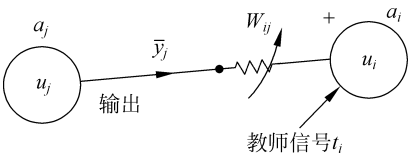


图 3.11　Hebb 学习规则

$$\Delta W_{ij} = G[a_i(t), t_i(t)] \times H[\bar{y}_j(t), W_{ij}] \tag{3.11}$$

其中，$t_i(t)$是神经元 u_i 的教师信号；G 是神经元 u_i 的活性度 $a_i(t)$和教师信号 $t_i(t)$的函数；H 是神经元 u_i 输出 $\bar{y}_j$ 和联结权重 W_{ij} 的函数。

输出 $\bar{y}_j(t)$与活性度 $a_i(t)$之间的关系为

$$\bar{y}_j(t) = f_j[a_i(t)] \tag{3.12}$$

其中，f_j 为非线性函数。

当上述的教师信号 $t_i(t)$没有给出时，函数 H 只与输出 $\bar{y}_j$ 成正比，于是式(3.11)可变为更简单的形式

$$\Delta W_{ij} = \eta a_i \bar{y}_j \tag{3.13}$$

其中，η 是学习率($\eta > 0$)。

上式表明，对一个神经元较大的输入或该神经元活性度大的情况，它们之间的联结权重会更大。Hebb 学习规则的哲学基础是联想，在这个规则基础上发展了许多非监督式联想学习模型。

2. 误差传播式学习

前述的函数 G 的值和教师信号 $t_i(t)$与神经元 u_i 实际的活性度 $a_i(t)$的差值成比例，即

$$G[a_i(t), t_i(t)] = \eta_1[t_i(t) - a_i(t)] \tag{3.14}$$

其中，η_1 为学习率；$[t_i(t) - a_i(t)]$表示差值，记为 δ。

函数 H 和神经元 u_j 的输出 $\bar{y}_j(t)$成比例，即

$$H[\bar{y}_j(t), W_{ij}] = \eta_2 \bar{y}_j(t) \tag{3.15}$$

其中，η_2 为学习率。

根据 Hebb 学习规则可得

$$\Delta W_{ij} = G[a_i(t), t_i(t)] \times H[\bar{y}_j(t), W_{ij}]$$

$$=\eta_1[t_i(t)-a_i(t)]\cdot\eta_2\bar{y}_j(t)$$
$$=\eta[t_i(t)-a_i(t)]\cdot\bar{y}_j(t) \tag{3.16}$$

其中,η 为学习率($\eta>0$)。

在式(3.16)中,如将教师信号 $t_i(t)$ 作为期望输出 d_i,而把 $a_i(t)$ 理解为实际输出 $\bar{y}_j$,则该式变为

$$\Delta W_{ij}=\eta[d_i-\bar{y}_j]\bar{y}_j(t)=\eta\cdot\delta\cdot\bar{y}_j(t) \tag{3.17}$$

其中,$\delta=d_i-\bar{y}_j$ 为期望输出与实际输出的差值,称式(3.17)为 δ 规则或误差修正规则。根据这个规则的学习算法,通过反复迭代运算,直至求出最佳的 ΔW_{ij} 值,使 δ 达到最小。

上述 δ 规则只适用于线性可分函数,不适用于多层网络非线性可分函数。1986 年,鲁姆尔哈特和辛顿等人系统地总结了误差传播式学习的研究成果,概括了具有普遍意义的 δ 规则,称为广义 δ 规则。根据广义 δ 规则,误差由输出层逐层反向传至输入层,而输出则是正向传播,直至给出网络的最终响应。因此,又称其为误差反向传播学习。

3. 概率式学习

从统计力学、分子热力学和概率论中关于系统稳态能量的标准出发,进行神经网络学习的方式,称为概率式学习。概率式学习的典型代表是 Boltzmann 机学习规则,它是基于模拟退火的统计优化方法,因此又称模拟退火算法。

Boltzmann 机模型是包括输入、输出和隐层的多层网络,但隐层间存在互联结构且网络层次不明显。对于这种网络训练的过程,就是根据下述规则对神经元 i、j 间的联结权重进行调整

$$\Delta W_{ij}=\eta(p_{ij}^{+}-p_{ij}^{-}) \tag{3.18}$$

其中,η 为学习率;p_{ij}^{+}、p_{ij}^{-} 分别是 i 与 j 两个神经元在系统中处于 α 状态和自由运转状态时实现联结的概率。调整权重的原则是,当 $p_{ij}^{+}>p_{ij}^{-}$ 时,则增加权重,否则减小权重。权重调整的这种规则就是 Boltzmann 机的学习规则。

4. 竞争式学习

在神经网络中的兴奋性或抑制性联结机制中引入竞争机制的学习方式,称为竞争式学习。它是属于无教师学习方式,它是利用不同层间的神经元发生兴奋性联结,以及同一层内距离很近的神经元间发生同样的兴奋性联结,而距离较远的神经元产生抑制性联结。竞争式学习的本质特征在于神经网络中高层次的神经元对低层次神经元输入模式进行竞争式识别。

1985 年,鲁姆尔哈特和兹普瑟提出了用于前馈网络竞争式学习规则。该网络结构设计成第一层为输入层,而以后的每一层都增加许多不重叠的组块,每一组块在特征识别中只有一个竞争优胜单元兴奋,其余单元受到抑制。

设 i 为输入层某单元，j 为获胜的特征识别单元，则它们之间的联结权重变为

$$\Delta W_{ij}=\eta(C_{ik}/nk-W_{ij}) \tag{3.19}$$

其中，η 为学习率；C_{ik} 为外部刺激 k 系列中第 i 项刺激成分；nk 为刺激 k 激励输入单元的总数。这种学习方式表明，在竞争中与输入单元间联结权重变化最大的优胜单元实现每一特征识别，而失败的单元 ΔW_{ij} 为零。科霍恩提出的自组织特征映射网络是采用竞争式学习机制的神经网络典型代表。

5. 强化学习

强化学习(Reinforcement Learning，RL)是 1961 年由明斯基(Minsky)提出的，又称再励学习、增强学习、评价学习。有别于监督学习、无监督学习，强化学习不要求预先给定任何数据，而是通过接收环境对动作的奖励(反馈)获得学习信息并更新模型参数 。

强化学习的灵感来源于心理学中的行为主义理论。强化学习的基本原理如图 3.12 所示，智能体在与环境的交互过程中，智能体对环境感知的当前状态为 s，从动作空间 A 中选择动作 a 执行。环境会根据智能体的动作提供一个反馈信号作为相应的奖赏 r，使智能体转移到新的状态 s'。智能体根据得到的奖励来调整自身的策略，并针对新的状态做出新的决策，以有效地适应环境。这种适应环境的行为便是学习，智能体通过最大化累积奖赏的方式来学习到最优策略。

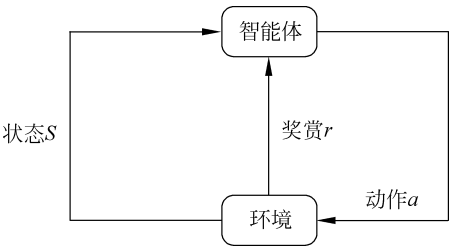


图 3.12　强化学习的基本原理

一个强化学习系统，除了智能体和环境，还包括 4 个要素：策略、奖赏函数、值函数及环境的模型。策略(决策函数)规定了智能体在每个可能的状态，应该采取的动作集合；奖赏函数是智能体在与环境的交互中，对所产生动作好坏作的一种评价；值函数(评价函数)是从长远的角度来考虑一个状态(状态-动作)的好坏。

智能体在当前状态 s 下，根据策略 π 来选择动作 a，执行该动作并以概率 $P^{a}_{s,s'}$ 转移到下一个状态 s'，同时接收到环境反馈回来的奖赏 r。策略 $\pi:S\rightarrow A$ 为从状态空间到动作空间的映射。强化学习的目标是通过调整策略来最大化累积奖赏，通常使用值函数估计某个策略 π 的优劣程度。

假设初始状态 $s_0=s$，策略 π 的状态值函数定义为

$$V^{\pi}(s)=\sum_{t=0}^{\infty}\gamma^{t}r(s_t,a_t)\mid s_0=s,a_t=\pi(s_t) \tag{3.20}$$

其中，$\gamma\in(0,1)$ 为衰减因子。由于最优策略是最大化值函数的策略，因此最优策略为

$$\pi^{*}=\arg\max_{\pi}V^{\pi}(s) \tag{3.21}$$

另一种形式的值函数是状态动作值函数，定义为

$$Q^{\pi}(s_t,a_t)=r(s_t,a_t)+\gamma V^{\pi}(s_{t+1}) \tag{3.22}$$

此时,得到的最优策略为

$$\pi^* = \arg\max_{a \in A} Q^{\pi}(s,a) \tag{3.23}$$

强化学习中常用算法包括:基于概率统计理论的蒙特卡罗方法;借助时间差分误差来更新值函数的Q-学习和SARAS学习;TD学习建立了蒙特卡罗和时间差分的统一描述,当$\gamma=1$时,TD(γ)变成了蒙特卡罗方法,$\gamma=0$时,TD(γ)变成了时间差分学习。

上面介绍的都是基于值函数的强化学习方法,它需要先求出值函数,再根据值函数来选择价值最大的动作执行。另一种基于策略的强化学习方法是一种直接逼近策略,优化策略,最终得到最优策略的方法。策略梯度法又可以分为确定策略梯度算法和随机策略梯度算法。

强化学习是一种无监督学习的重要方法,具有自学习、在线学习和普适性的优点,已广泛用于博弈论、控制论、运筹学、信息论、智能控制、多主体系统、机器人等领域。

6. 深度学习

深度学习(Deep Learning,DL)是2006年由辛顿教授等提出的一种深度信念网络的训练方法。"深度"是指从输入层到输出层之间所包含的隐层数很多,它反映了网络层次的深度。将具有这样深层次结构的神经网络称为深度网络,对深度网络的训练方法统称为深度学习算法。

辛顿提出的深度信念网络由多个含有两层的网络由下向上堆叠而成。最上面的两层之间为无向连接,其余的层间均为有向连接。如果同时对网络所有的层进行训练,时间复杂度就会太高;如果每次训练一层,偏差就会逐层传递,由于深度网络的神经元和参数太多,导致严重欠拟合。

为了解决深度神经网络在训练上的困难,辛顿提出把深度网络的训练过程分为预训练和调优两个阶段。

预训练过程首先从底层开始,自下而上的采用无监督学习方式逐层进行训练,利用数据每次训练一个单层网络参数,这一层的输出作为上一层的输入,直到所有的层都训练完为止。预训练过程,实际上是对网络各层神经元联结权重的初始化过程。

调优阶段是在预训练完成之后,将除最顶层外的其他层间的权重变为双向,这样最顶层仍然是一个单层神经网络。再用监督学习去调整所有层间权重,使网络参数进一步优化。

深度神经网络具有优异的特征学习能力,深度学习能够通过组合低层特征来形成更加抽象的高层表示属性类别或特征,以发现数据的分布式特征表示,这些特征对数据具有更本质的刻画的特点,有利于解决图像识别、语音识别和分类等复杂问题。

7. 深度强化学习

深度强化学习(Deep Reinforcement Learning,DRL)是2015年由姆尼(Mnih)等提出的。深度学习(DL)善于事物特征提取和感知,强化学习(RL)长于选择与决策以及寻找最

优策略。深度强化学习方法将深度学习的感知能力和强化学习的决策能力相结合，可以在复杂高维的状态空间中进行端到端的感知决策。深度强化学习原理如图 3.13 所示。

深度强化学习的过程如下：

(1) 每个时刻智能体在与环境交互中，得到一个高维度的观察，并利用深度学习方法来感知观察，以得到具体的状态特征表示；

(2) 基于预期回报来评价各动作的价值函数，并通过某种策略将当前状态映射为相应的动作；

(3) 环境对此动作做出反应，并得到下一个观察。

通过不断循环上述过程，最终可以获得实现目标的最优策略。

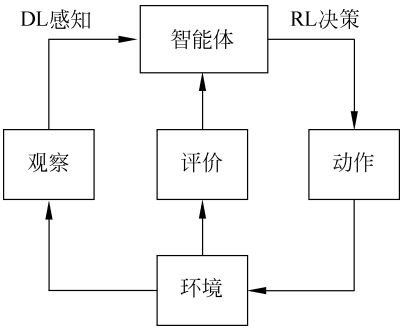


图 3.13　DRL 原理框图

深度强化学习也分为基于值函数的深度强化学习和基于策略梯度的深度强化学习两类。

(1) 基于值函数的深度强化学习

将卷积神经网络和 Q-学习结合，姆尼提出一种深度 Q 网络(Deep Q Network，DQN)，用于处理基于视觉感知的控制任务。DQN 采用时间上相邻的 4 帧游戏画面作为原始图像输入，经过深度卷积神经网络和全连接神经网络，输出状态动作函数 $Q(s,a;\theta)$ 去逼近目标值，其中 s 为状态，a 为动作，θ 为参数，从而实现了端到端的学习控制。

(2) 基于策略梯度的深度强化学习

策略梯度优化方法是通过不断计算策略期望总奖赏来对策略参数的梯度更新策略参数，最终收敛于最优策略。因此，在求解 DRL 问题时，可以采用参数为 θ 的深度神经网络来进行参数化表示策略，并利用策略梯度方法来优化策略。采取策略梯度优化方法能够直接优化策略的期望总奖赏，并以端对端的方式直接在策略空间中搜索最优策略，省去了烦琐的中间环节。

最常见的策略梯度思想是增加总奖赏较高情节出现的概率。在大规模状态的 DRL 任务中，可以通过深度神经网络参数化表示策略，并采用传统的策略梯度方法来求解最优策略。

3.2　控制和识别中的常用神经网络

3.2.1　感知器

1957 年，美国学者罗森布拉特提出了一种用于模式分类的最简单神经网络模型，称为感知器(perceptron)。它是由阈值元件组成且具有单层计算单元的神经网络，它具有学习功能。一个单元的模型如图 3.3 所示。

感知器信息处理的规则为

$$y(t)=f\left[\sum_{i=1}^{n}W_i(t)x_i-\theta\right] \tag{3.24}$$

其中，$y(t)$为 t 时刻输出；f 为阶跃函数；$W_i(t)$为 t 时刻第 i 个输入的加权；x_i 为输入向量的一个分量；θ 为阈值。

感知器的学习规则如下：

$$W_i(t+1)=W_i(t)+\eta\left[d_i-y(t)\right]x_i \tag{3.25}$$

其中，η 为学习率($0<\eta<1$)；d_i 为期望输出(又称教师信号)；$y(t)$是实际输出。通过不断调整权重，使得 W 对一切样本均保持稳定不变，学习过程就结束。

上述的单层感知器的输出是一个二值量，主要用于模式分类。能够解决一阶谓词逻辑问题，如逻辑与、逻辑或问题，但不能解决像异或问题的二阶谓词逻辑问题，感知器的学习算法保证收敛要求函数是线性可分的(指输入样本函数类成员分别位于直线分界线的两侧)，当输入函数不满足线性可分条件时，上述算法受到了限制。

3.2.2 前馈神经网络

前馈网络又称前向网络，由于采用误差反向传播学习算法又称 BP 网络。如图 3.14 所示，它包含输入层、隐层及输出层，隐层可以为一层或多层，每层上的神经元称为结点或单元。前馈网络的信息传播是由输入单元传到隐单元，最后传到输出单元。这种含有隐层的前馈网络的隐单元可以任意构成它们自身的输入表示，输入单元和隐单元间的权重决定每个隐单元何时是活性的，因此，借修改这些权重，可以选择一个隐单元代表什么。

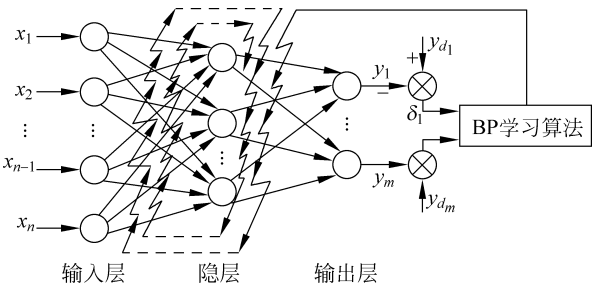


图 3.14 前馈网络结构及 BP 学习算法示意图

1. BP 网络误差反向传播学习算法

为了研究神经网络怎样从经验中学习，应该首先向网络提供一些训练例子，可以通过下述方法，教会一个三层前馈网络完成某个特定的任务。其方法步骤如下：

(1) 向网络提供训练例子，包括输入单元的活性模式和期望输出单元的活性模式；

(2) 确定网络的实际输出与期望输出之间允许的误差;

(3) 改变网络中所有联结权重,使网络产生的输出更接近于期望输出,直到满足给定的允许误差。

下面以网络识别手写数字为例说明上述方法。比如使用 256 个传感器组成传感器阵列,每个传感器记录一个数字的一小部分面积内是否有笔写的痕迹存在。因此,网络需要 256 个输入单元(每个传感器一个),10 个输出单元(每种数字一个)和许多隐单元。为了便于传感器记录每种数字,网络应在适当的输出单元中产生高的活性,并在其他输出单元中产生低的活性。

为了训练此网络,提供一幅某个数字的图像,并把 10 个输出单元的实际活性和期望活性加以比较,然后计算误差,将实际活性与期望活性差值的平方定义为误差。其次,改变网络中所有联结权重以减少误差。针对每种数字的多种不同图像重复训练,直到网络能对每个手写数字正确地归类为止。

实现上述训练过程的关键是必须改变每个权重,且其变化量应正比于权重改变时误差的变化率,此量称权重的误差导数,简记为 EW。但是,对 EW 的有效计算是十分复杂棘手的问题。如果采取稍许扰动每个权重,并观察误差如何变化,这种方法效率太低,因为要求对每一个权重都要单独加扰动。

1974 年前后,正在哈佛大学攻读博士的韦尔博斯(Werbos)发明了一种高效的计算 EW 的方法,即误差反向传播算法(BP 算法)。只因发明后多年未受到重视,也没有充分体会到它的用处,直到 1985 年鲁姆尔哈特和帕伯特两人各自再次发现该算法后才得以推广。

BP 算法的实质是以梯度法求取网络误差平方和使目标函数达到最小值。首先计算改变一个单元活性水平时误差的变化率 EA,再计算每个权重的误差导数 EW。对于输出单元,EA 只是实际输出和期望输出间的差值。为计算在输出层前面一层的一个隐单元的 EA,首先辨别该隐单元和与它相连的那些输出单元间的所有权重,然后把这些权重乘以这些输出单元的 EA 并求和,此和值即为所选定隐单元的 EA。将所有隐单元的 EA 计算出后,可用类似的方法计算其他层的 EA 值,计算的顺序恰好与活性传播过网络路径相反的方向逐层进行,故称为误差反向传播算法。计算出一个单元的 EA 后,再计算该单元每条输入联结的 EW,EW 是 EA 和经过该输入联结活性的乘积。

对于非线性单元,反向传播算法还包括另外一步,即在反向传播前,必须把 EA 变换为 EI,EI 是一个单元所收到的总输入变化时误差的变化率。

综上所述,为了训练前馈网络完成某项任务,必须调整每个单元的权重,即减小期望输出与实际输出间的误差,为此必须计算每个权重变化时误差的变化,即误差导数 EW,而反向传播算法是一种确定 EW 的最有效的方法。

2. BP 算法的计算机实现步骤

(1) 初始化,对所有权重赋以随机任意小值,并对阈值设定初值;

(2) 给定训练数据集,即提供输入向量 $\boldsymbol{X}$ 和期望输出 $\bar{y}$;

(3) 计算实际输出 y

$$y_j = f\left(\sum W_{ij} x_i\right) \tag{3.26}$$

其中,f 函数为 Sigmoid 函数

$$f(x) = \frac{1}{1+\mathrm{e}^{-(x-\theta)}} \tag{3.27}$$

(4) 调整权重,按误差反向传播方向,从输出结点开始返回到隐层修正权重为

$$W_{ij}(t+1) = W_{ij}(t) + \eta \delta_j y_j \tag{3.28}$$

其中,η 为大于零的学习率,δ_j 为结点 j 的实际活性与期望活性的差值。根据结点 j 的形式不同,δ_j 的计算为

$$\delta_j = \begin{cases} y_j(1-y_j)(\bar{y}_j - y), & \text{当 } j \text{ 为输出结点时} \\ y_j(1-y_j)\sum\limits_k \delta_k W_{jk}, & \text{当 } j \text{ 为隐结点时} \end{cases} \tag{3.29}$$

当使用冲量时,调整权重公式变为

$$W_{ij}(t+1) = W_{ij}(t) + \eta \delta_j y_j + \alpha\left[W_{ij}(t) - W_{ij}(t-1)\right] \tag{3.30}$$

其中,α 为动量因子,$0<\alpha<1$。

(5) 返回第(2)步重复,直至误差满足要求为止。

3. BP 算法的改进算法

BP 算法实质上是把一组样本输入输出问题转化为一个非线性优化问题,并通过梯度算法利用迭代运算求解权重问题的一种学习方法。已经证明,具有 Sigmoid 非线性函数的三层神经网络能够以任意精度逼近任何连续函数。但 BP 算法尚存在以下一些缺点:

(1) 由于采用非线性梯度优化算法,易形成局部极小而得不到整体最优。

(2) 算法迭代次数过多,致使学习效率低,收敛速度慢。

(3) BP 网络无反馈连接,影响信息交换速度和效率。

(4) 网络的输入结点、输出结点由问题而定,但隐结点的选取根据经验,缺乏理论指导。

(5) 在训练中学习新样本有遗忘旧样本的趋势,且要求每个样本的特征数目要相同。

针对 BP 算法的缺点,国内外学者提出了许多改进算法,如变步长、增加惯性因子、修改作用函数、与其他算法融合等。

4. 前馈网络及其 BP 算法的意义

感知器(感知机)是单层的前馈网络,它的神经元输出函数为阶跃函数,用于模式识别。BP 网络的神经元输出函数为 Sigmoid 函数,输出量是 0～1 的连续量。一个三层前馈网络就能实现从输入到输出的任意的非线性映射,因此可用于建模、优化、控制等。

前馈网络的重要意义在于，不仅它的网络结构成为后来设计 RBF 网络、深度网络等许多网络的基础，而且 BP 算法也成为后来许多网络训练、学习算法或推广算法应用的基础。

3.2.3　径向基神经网络

前馈网络的 BP 算法可以看作递归技术的应用，属于统计学中的随机逼近方法。此外，还可以把神经网络学习算法的设计看作一个高维空间的曲线拟合问题。这样，学习等价于寻找最佳拟合数据的曲面，而泛化等价于利用该曲面对测试数据进行插值。1985 年，鲍威尔(Powell)首先提出了实多变量插值的径向基函数(Radial Basis Function，RBF)方法。1988 年，布鲁姆里德(Broombead)和罗威(Lowe)将 RBF 用于神经网络设计，而穆迪(Moody)和达肯(Darken)提出了具有代表性的 RBF 网络学习算法。

1. RBF 神经网络模型

图 3.15 给出了一个三层结构的 RBF 网络模型，包括输入层、隐层(径向基层)和输出层(线性层)，其中每一层的作用各不相同。输入层由信号源结点组成，仅起到传入信号到隐层的作用，可将输入层和隐层之间看作权重为 1 的联结；隐层通过非线性优化策略对激活函数的参数进行调整，完成从输入层到隐层的非线性变换；输出层采用线性优化策略对线性权重进行调整，完成对输入层激活信号的响应。

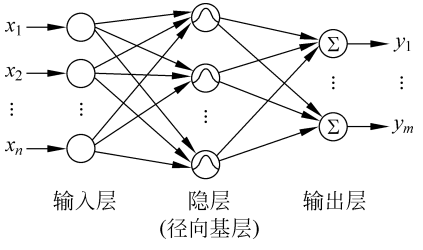


图 3.15　RBF 网络的三层结构

2. RBF 网络的工作原理

RBF 网络隐层的每个神经元实现一个径向基函数(RBF)，这些函数被称为核函数，通常为高斯型核函数。当输入向量加到输入端时，隐层每一个单元都输出一个值，代表输入向量与基函数中心的接近程度。隐层的各神经元在输入向量与 RBF 的中心向量接近时有较大的反应，也就是说，各个 RBF 只对特定的输入有反应。

如果输入向量与权重向量相差很多，则径向基层的输出接近 0，经过输出层线性神经元的输出也为 0；如果输入向量与权重向量很接近，则径向基层的输出接近于 1，经过线性神经元的输出值更靠近输出层的权重。

3. RBF 网络的学习算法

RBF 网络学习算法需要求解 3 个参数：基函数中心、宽度和隐层到输出层的权重。径向基函数通常采用高斯函数，根据对径向基函数中心选取方法的不同，提出了多种 RBF 网络学习算法，其中穆迪和达肯提出的两阶段学习算法较为常用。

第一阶段为非监督学习,采用K-均值聚类法决定隐层的RBF的中心和方差;第二阶段为监督学习,利用最小二乘法计算隐层到输出层的权重。

RBF神经网络的激活函数通常采用的高斯函数为

$$G_j(\boldsymbol{x}-\boldsymbol{c}_j)=\exp\left[-\frac{1}{2\sigma_j^2}\|\boldsymbol{x}-\boldsymbol{c}_j\|^2\right] \tag{3.31}$$

其中,$\boldsymbol{x}$ 是输入向量;$\boldsymbol{c}_j$ 是第 j 个神经元的RBF中心向量;σ_j 是第 j 个RBF的方差;$\|\cdot\|$ 表示欧氏范数。

RBF神经网络的输出为

$$y_k(\boldsymbol{x})=\sum_{j=1}^{J}w_{kj}G_j(\boldsymbol{x}-\boldsymbol{c}_j) \tag{3.32}$$

实现RBF网络学习算法的具体步骤如下:

(1) 随机选取训练样板数据作为聚类中心向量初始化,确定 J 个初始聚类中心向量。

(2) 将输入训练样板数据 x_i 按最邻近聚类原则选择最近的聚类 j^*。

$$c_{j^*}=\arg\min_j\|\boldsymbol{x}_i-\boldsymbol{c}_j\| \tag{3.33}$$

(3) 聚类中心向量更新。若全部聚类中心向量不再发生变化,则所得到的聚类中心即为RBF网络最终基函数中心,否则返回第(2)步。

(4) 采用较小的随机数对隐层和输出层间的权重初始化。

(5) 利用最小二乘法计算隐层和输出层之间神经元的联结权重 w_{kj}。

(6) 权重更新。首先求出各输出神经元中的误差

$$e_k=d_k-y_k(\boldsymbol{x}) \tag{3.34}$$

其中,d_k 是输出神经元 k 的期望输出。然后,再更新权重为

$$w_{kj}^{\text{new}}=w_{kj}^{\text{old}}+\eta e_k G_j(\boldsymbol{x}-\boldsymbol{c}_j) \tag{3.35}$$

其中,η 是学习率。

(7) 若满足终止条件,则结束;否则返回第(5)步。

4. RBF神经网络的特点

RBF网络同BP网络一样,都能以任意精度逼近任意连续函数。由于BP网络的隐结点使用的Sigmoid函数值在输入空间无穷大范围内为非零值,具有全局性。而RBF网络隐结点使用的高斯函数,使得RBF神经网络是一个局部逼近网络。RBF网络比BP网络的学习速度更快,这是因为BP网络必须同时学习全部权重,而RBF网络一般分两段学习,各自都能实现快速学习。

3.2.4 反馈神经网络

人的大脑没有设定有序的地址来存储记忆的内容,而是分布存储在神经网络的不同区

域，要取出时，当线索与回想的内容相同时，就可以取出；不相同时，就相互联想，分布存储在与线索贴近的内容的新区域。

1982 年，美国物理学家霍普菲尔德受到磁场具有记忆功能的启发，结合生物脑思维的机理，提出了一种由非线性元件构成的单层反馈网络系统，被称为 Hopfield 网络，具有模拟大脑的联想记忆功能，可用于模式识别和优化问题。

Hopfield 网络具有反馈，并引用了能量函数，它是一个非线性动力学系统。非线性动力学系统本身涉及稳定性、吸引子以及混沌现象等问题，因此研究反馈网络要比前馈网络（静态网络）复杂得多。在非线性动力学系统的相空间中，当 $t\to\infty$ 时，所有轨迹线都趋于一个稳定的不动点集，称为吸引子[15]。

1. Hopfield 网络模型

Hopfield 网络是一种网状网络，可分为离散和连续两种类型。离散网络的结点仅取 +1 和 −1（或 0 和 1）两个值，而连续网络取 0 和 1 之间任一实数。图 3.16 给出一种离散 Hopfield 网络的结构形式。

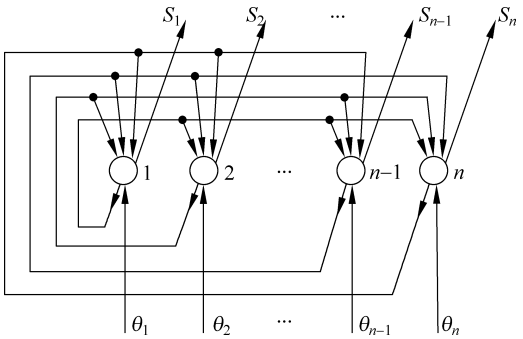


图 3.16　Hopfield 网络的结构

设此网络含有 n 个神经元，神经元 i 的状态 S_i 取 0 或 1，各神经元按下列规则随机地、异步地改变状态

$$S_i=\begin{cases}1, & \sum W_{ij}S_i+I_i-\theta_i>0\\ 0, & \sum W_{ij}S_i+I_i-\theta_i<0\end{cases}\tag{3.36}$$

其中，W_{ij} 为神经元 i 与 j 间的联结权重；I_i 为神经元 i 的偏流；θ_i 为其阈值。

Hopfield 根据系统动力学和统计力学的原理，在网络系统中引入能量函数的概念，给出了网络系统稳定性定理如下。

定理 3.1　设 (W,S) 是一个神经网络，若 $W_{ij}=W_{ji}$，且 $W_{ii}>0$，各神经元随机异步地改变状态，则此网络一定能收敛到稳定的状态。

证明　在网络中引入能量函数

$$E=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}W_{ij}S_iS_j-\sum_{i=1}^{n}I_iS_i+\sum_{i=1}^{n}\theta_iS_i \quad i\neq j \tag{3.37}$$

能量函数 E 随着状态 S_k 的变化为

$$\frac{\Delta E_k}{\Delta S_k}=-\frac{1}{2}\sum_{i=1}^{n}W_{ki}S_i-\frac{1}{2}\sum_{j=1}^{n}W_{jk}S_j-I_k+\theta_k$$

当 $W_{ik}=W_{ki}$ 时,则得

$$\Delta E_k=-\left(\sum_{j=1}^{n}W_{kj}S_j+I_k-\theta_k\right)\Delta S_k \tag{3.38}$$

若令 S_k 表示上式括号中部分,即 $S_k=\sum_{j=1}^{n}W_{kj}S_j+I_k-\theta_k$,则由式(3.36)可知,$S_k$ 与 ΔS_k 同号,再从式(3.38)可以看出,S_k 与 ΔS_k 之积大于零,故 $\Delta E_k<0$。这就表明网络系统总是朝着能量减小的方向变化,最终进入稳定状态。

Hopfield 网络模型的基本原理是:只要由神经元兴奋的算法和联结权重所决定的神经网络的状态,在适当给定的兴奋模式下尚未达到稳定状态,那么该状态就会一直变化下去,直到预先定义的一个必定减小的能量函数达到极小值时,状态才达到稳定而不再变化。

对于连续的 Hopfield 网络,通过引入能量函数,可以类同于离散模型的情况加以研究,它同样朝着能量减小的方向运行,最终到达一个稳定的状态,这里不再讨论。

2. Hopfield 网络的联想记忆功能

Hopfield 网络是一个非线性动力学系统。引入能量函数后,该网络系统在一定条件下总是朝着系统能量减小的方向变化,并最终达到能量函数的极小值。如果把这个极小值所对应的模式作为记忆模式,那么以后当给这个网络系统一个适当的激励时,它就能成为回想起已记忆模式的一种联想记忆装置,即 Hopfield 网络具有联想记忆功能。

Hopfield 网络的联想记忆过程可分为学习和联想两个阶段:在给定样本的条件下,按照 Hebb 学习规则,调整联结权重使得存储的样本成为动力学的吸引子,这个过程就是学习阶段;联想是指在已调整好权重不变的情况下,给出部分不全或受了干扰的信息,按照动力学规则改变神经元的状态,使系统最终收敛到动力学的吸引子。

Hopfield 网络模型的动力学规则是指若网络结点在 $S(0)$初始状态下,经过 t 步运行后将按下述规则达到 $S(t+1)$状态,即

$$S(t+1)=\mathrm{sgn}\left[\sum_{j=1}^{n}W_{ij}S_j(t)+I_i\right] \tag{3.39}$$

其中,sgn 为符号函数。

3. Hopfield 网络的联想记忆学习算法

在偏流 I 为零的情况下,本学习算法的具体步骤如下:

(1) 按照 Hebb 规则设置权重

$$W_{ij}=\begin{cases}\sum_{m=1}^{n}x_i^m x_j^m, & i\neq j \quad i,j=1,2,\cdots,n\\ 0, & i=j\end{cases} \tag{3.40}$$

其中,W_{ij} 是结点 i 到结点 j 的联结权重;x_i^m 表示样本集合 m 中的第 i 个元素,$x_i\in\{-1,+1\}$。

(2) 对未知样本初始化

$$S_i(0)=x_i, \quad i=1,2,\cdots,n \tag{3.41}$$

其中,$S_i(t)$是 t 时刻结点 i 的输出;x_i 是未知样本的第 i 个元素。

(3) 迭代计算

$$S_j(t+1)=\mathrm{sgn}\left[\sum_{i=1}^{n}W_{ij}S_i(t)\right], \quad j=1,2,\cdots,n \tag{3.42}$$

直至结点输出状态不改变时,迭代结束。此时结点的输出状态即为未知输入最佳匹配的样本。

(4) 返回第(2)步继续迭代。

4. Hopfield 网络的优化计算

Hopfield 网络用于优化问题的计算与用于联想记忆的计算过程是对偶的。在解决优化问题时,权重矩阵 $\boldsymbol{W}$ 已知,目的是求取最大能量 E 的稳定状态。通过将能量函数和代价函数相比较,求出能量函数中权重和偏流,并以此去调整相应的反馈权重和偏流,进行迭代计算,直到系统收敛到稳定状态为止。最后将所得到的稳定状态变换为实际优化问题的解。

旅行商问题(TSP)是给定 N 个城市,从某一城市出发不重复地走遍所有城市,再回到原出发地所经过的路径必须最短。1985 年,霍普菲尔德本人应用 900 个神经元组成连续的 Hopfield 网络,在 0.2s 内成功地解决了著名的旅行商问题。此问题用普通搜索法极其费时,而采用 Hopfield 网络在极短时间内找到虽不最短,但却是接近最短的最优近似解,充分显示出 Hopfield 网络优化计算的巨大潜力。

3.2.5　小脑模型神经网络

人的小脑具有管理和协调运动功能。小脑皮层的神经系统从肌肉、四肢、关节、皮肤等接收感觉信息,并感受反馈信息,然后将这些获得的信息整合到一特定的存储区域。当需要的时候,将这些存储的信息从中取出,作为驱动和协调肌肉运动的控制指令。当感受信息和反馈信息出现差异时,通过联想调整达到协调运动控制的目的,这一过程便是学习。

1975 年,阿布思(Albus)根据神经生理学小脑皮层结构特点,提出了一种小脑模型关联

控制器(Cerebellum Model Articulation Controller,CMAC),被称为CMAC网络。

1. CMAC网络的结构

CMAC网络是模拟小脑神经系统的协调功能而设计的,其结构如图3.17所示。CMAC的功能是通过多个映射实现的。第一个映射是从$S\rightarrow A$的映射,其中S为输入状态空间,A为概念存储器。映射的基本原则是:对相近的输入映射到A中有一定的重合,而不相近的输入在A中也相距较远。第二个映射是从$A\rightarrow M$的映射,其中A为概念存储器,M为实际存储器。由于要学习的问题不会是全部可能的输入,故M的存储容量要比A小得多,这就决定了从A到M的映射为多对一的随机映射,因此可用散列编码(hash coding)实现这种映射。

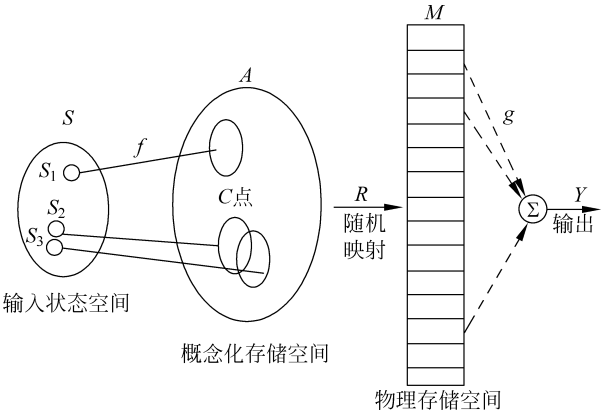


图3.17 CMAC网络的结构

2. CMAC网络的工作原理

图3.18给出了CMAC网络原理结构图。图中以输入状态空间的集合元素数等于3为例,输入S_1,S_2,S_3经过量化映射Q变换为q_1,q_2,q_3,输入量的最大值$S_{i\max}$,最小值$S_{i\min}$及量化等级$q_{i\max}$决定了分辨率q_i为

$$q_i=Q(S_i,S_{i\max},S_{i\min},q_{i\max}) \tag{3.43}$$

其中,输入集合数目$i=1,2,\cdots,m$,本例中$m=3$。由于量化等级可以根据需要而增加,因此CMAC网络可达到很高精度。

再对离散的输入量到虚地址进行映射,它相当于从$S\rightarrow A$的映射,映射后变为由3段合成的4个虚地址(V_1,V_2,V_3,V_4),即每一个V都由3段所组成,这样的映射使得CMAC网络具有泛化能力,即当某一输入量化值变化一个等级时,只有一个虚地址段变化为1,而其他虚地址段都保持不变,这样,就保证了在相邻量化值的指示权重用的4个虚地址中有3个是相同的。这就意味着,在输入空间中相近的输入量能给出相近的输出。

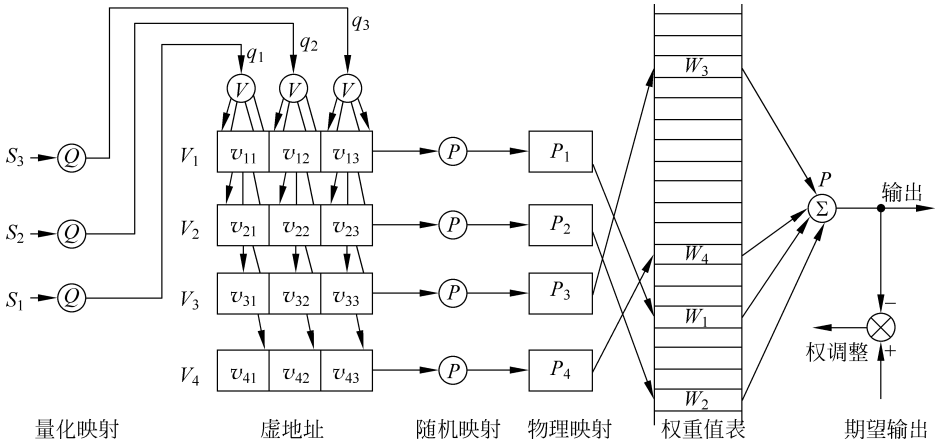


图 3.18　CMAC 网络原理结构图

对上述组合成的 3 段虚地址，经过一个多对一的随机映射 P 后，得到与输入量相对应的物理地址为

$$p_j = P(v_j), \quad j = 1,2,\cdots,g \tag{3.44}$$

其中，P 为一种以散列编码形成的映射。由得到的 p_j 可以通过输出权重值表获得相应的权重为

$$W_j = W(p_j) \tag{3.45}$$

输出为

$$y = \sum_{j=1}^{g} W_j \tag{3.46}$$

在训练过程中，如果网络输出 y 与期望输出 d 不同，则权重修正为

$$W_j(K+1) = W_j(K) + \beta\left(d - \sum_{j=1}^{m} W_j\right)/g \tag{3.47}$$

其中，β 为学习因子($\beta \leqslant 1$)；g 为推广能力，其值越大，相邻输入的共同虚地址越多，则 CMAC 的泛化能力越强。

综上所述，CMAC 网络是一种通过多种映射实现联想记忆网络，这种映射实质上是一种智能查表技术。它模拟小脑皮层神经系统感受信息、处理信息、存储信息，并通过联想利用信息的功能。它能实现无教师学习，具有在线学习能力，不仅学习速度快，而且精度高，可以处理不确定性知识。它在实时控制，尤其在机器人实时控制领域有着广泛的应用。

3.2.6　大脑模型自组织神经网络

脑神经学研究表明，人脑中大量的神经元处于空间的不同区域，有着不同的功能。它们

敏感着各自的输入信息模式的不同特征,这样就形成了大脑各种不同感知路径。在大脑皮质中,神经元之间信息交互的共同特征是最邻近的两个神经元互相激励而兴奋,较远的相互抑制,更远的又是弱激励,这种局部作用的交互关系形成一个墨西哥草帽形状的分布关系,如图 3.19 所示。

1987 年,芬兰科霍恩教授提出了一种模拟大脑神经系统自组织特征映射功能的神经网络,它是一种竞争式学习网络,在学习中能无监督地进行自组织自学习,因此又称为自组织神经网络,或称为 Kohonen 网络。

1. Kohonen 网络结构与功能

Kohonen 网络如图 3.20 所示,它是一个完全相互联结的神经元组成的二维点阵结构,在网络的一定邻域内各神经元间存在交互侧向反馈作用。每个神经元的输出都是网络中其他神经元的输入,而每个神经元又都有相同的输入形式。网络中有两种联结权重:一种是神经元对外部输入反应的联结权重;另一种是神经元之间的联结权重。它的大小控制着神经元之间的交互作用的大小。

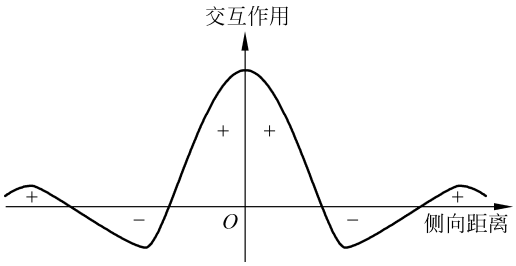


图 3.19 侧向交互作用关系

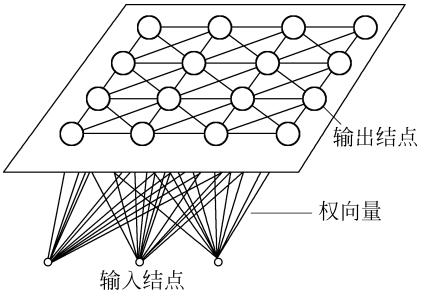


图 3.20 Kohonen 网络

科霍恩认为,一个神经网络接受外界输入模式,将分成不同区域,各区域中邻近的神经元通过交互作用,相互竞争,自适应地形成了对输入模式的不同响应检测器,这就是科霍恩提出的自组织特征映射网络的基本思想。

Kohonen 网络在结构上模拟了大脑皮质中的神经元呈二维空间点阵结构,在功能上通过网络中神经元间的交互作用和相互竞争,模拟了大脑信息处理的聚类功能、自组织、自学习功能。

2. Kohonen 网络学习规则

Kohonen 网络当外部输入模式出现后,网络所有神经元都同时工作,它们之间的突触联结权向量试图模仿输入信号,实现网络自组织的目标。这一自组织学习过程包括采用竞争机制选择最佳匹配神经元和权向量自适应更新两个过程。

1）选择最佳匹配神经元

当输入信号 $\boldsymbol{X}$ 与神经元 i 获得最佳匹配时，它们之间的欧氏距离为

$$\| \boldsymbol{X}-\boldsymbol{W}_c \| = \min_i \| \boldsymbol{X}-\boldsymbol{W}_i \| \tag{3.48}$$

其中，$\boldsymbol{X}$ 为输入空间 $\mathbf{R}^n$ 中的输入向量，$\boldsymbol{X}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$；$\boldsymbol{W}_i$ 为相应的权向量，$\boldsymbol{W}_i=[W_{i1},W_{i2},\cdots,W_{in}]^{\mathrm{T}}$。$\boldsymbol{X}$ 与 $\boldsymbol{W}_i$ 的匹配度等于它们的内积 $\boldsymbol{X}^{\mathrm{T}}\cdot\boldsymbol{W}_i$，其最大处正是由于神经元的不断交互作用，其输出分布所形成的“气泡”(bubble)中心 C。

因为 $\boldsymbol{X}$ 与 $\boldsymbol{W}_i$ 之间内积最大，就意味着它们之间的向量差的范数 $\| \boldsymbol{X}-\boldsymbol{W}_i \|$ 最小，这个最小的距离确定了神经元 C 在竞争中获胜。式(3.48)被称为匹配定律。当网络训练好之后，如果同样的输入模式出现时，某个神经元就兴奋起来，表示该神经元已经认识了这个模式。

2）权向量的自适应更新过程

当某一输入与被选神经元 j 的权重 W_j 有差异时，除该权重修正外，被选神经元的邻域 N_j 中的其他神经元也将根据它们的误差以及按照距离的大小作适当的调整，越靠近 j 的神经元调整得就越多，这样形成的邻域关系使得输入模式相近时，对应的神经元在位置上也靠近。这种思想来源于人脑内的感觉映射，Kohonen 网络中被选神经元 j 的邻域关系如图 3.21 所示。

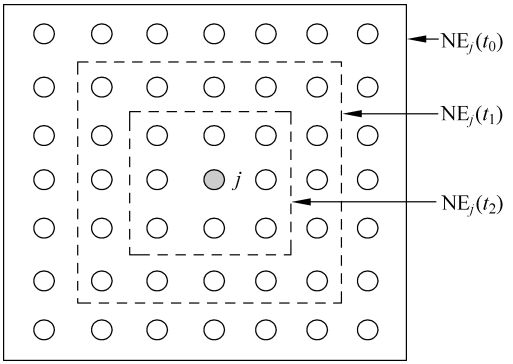


图 3.21　被选神经元 j 及其邻域变化($t_0<t_1<t_2$)

Kohonen 网络的学习过程也可以这样加以描述：对于每一个网络的输入，只调整一部分权重，使权向量更接近或更偏离输入向量，这一调整过程就是竞争学习。随着不断学习，所有权向量都在输入向量空间相互分离，形成了各自代表输入空间的一类模式，这就是 Kohonen 网络的特征自动识别的聚类功能。

3. Kohonen 网络学习算法

Kohonen 网络的学习算法实现步骤如下：

(1) 从 n 个输入到 m 个输出结点间随机赋任意小权重，并设定邻域的初始半径如

图 3.22 所示。

(2) 给定新的输入信号。

(3) 计算输入到各输出结点 j 之间距离

$$d_j=\sum_{i=0}^{n-1}[x_i(t)-W_{ij}(t)]^2 \tag{3.49}$$

(4) 选择 $d_{j^*}=\min_i d_j$ 的结点 j^* 作为输出结点。

(5) 对结点 j^* 和邻域内 $\mathrm{NE}_j(t)$所有结点按下式更新权重,其他结点保持不变。

$$W_{ij}(t+1)=W_{ij}(t)+\eta(t)[x_i(t)-W_{ij}(t)] \tag{3.50}$$

其中,$j\leqslant \mathrm{NE}_j(t)$,$0\leqslant i\leqslant n$;$\eta(t)$是自适应学习率,它随时间缓慢减小,$0\leqslant\eta(t)<1$。

(6) 满足终止条件,则结束;否则返回第(2)步重复计算。

应该指出,Kohonen 网络是一类重要的竞争学习的自组织网络,由 Kohonen 网络和其他形式网络组合,可以形成一大类神经网络,这类网络具有无教师学习特点,可以用于智能机器人控制、模式识别等系统。

3.2.7 Boltzmann 机

1985 年,加拿大多伦多大学教授辛顿等人基于统计物理学和 Boltzmann 概率分布,提出了一种模拟退火过程的神经网络模型,简称为 Boltzmann 机。

模拟退火的基本思想源于统计力学,统计力学是研究一个多自由度的系统在某个温度下达到热平衡时的行为特性。金属当高温熔化时,所有原子都处于高能的自由运动状态,随着温度的降低,原子的自由运动减弱,物体能量降低,只要在凝结温度附近,使温度下降足够慢,原子排列就越来越规整,而形成结晶,这一过程称退火过程。物体的上述结晶过程可对应多变量函数的优化过程,因此模拟退火过程,可以研究多变量的优化问题。

如果在退火过程的每一步,随机地改变原子的位置,导致整个系统能量改变 ΔE,若 $\Delta E<0$,则该原子就会处于新的位置,否则,该原子的位置可能不变或可能变到新的位置,而其改变的概率按 Boltzmann 分布变化。

1. Boltzmann 机的结构

Boltzmann 机是一个相互连接的神经网络模型,其主要特点是隐结点间具有相互结合的关系,每个神经元都根据自己的能量差 ΔE_i 随机地改变自己或为 1 或为 0 的状态,而单元 i 状态为 1 的概率为

$$P_i(\Delta E_i)=\frac{1}{1+\mathrm{e}^{-\Delta E_i/T}} \tag{3.51}$$

其中,T 为温度参数。当 $T>0$ 时,P_i 函数趋于阶跃函数;当 T 很大时,两种状态近于各半,而能量差为

$$\Delta E_i = E(S_i = 1) - E(S_i = 0) \tag{3.52}$$

假设网络的联结权重是对称的，引入能量函数

$$E = -\frac{1}{2}\sum_{i \neq j} W_{ij} S_i S_j \tag{3.53}$$

其中，S_i 是包括输入层、隐层和输出层中所有单元的状态。当系统达到平衡时，能量函数达到极小值，可以证明 Boltzmann 机是收敛的。

2. Boltzmann 机的训练步骤

(1) 设定初始高温并随机给定全部权重。

(2) 给定一输入向量，用已有的权重计算代价函数。

(3) 用 Boltzmann 分布 $P(x) = e^{-x^2/T^2}$ 随机改变每个权重。

(4) 重新计算代价函数，若减小，则将权重置成常数；否则返回第(3)步。

(5) 求新的温度值如下：

$$T(n+1) = T(0)/\log(1+n), \quad n \geqslant 1 \tag{3.54}$$

(6) 满足终止条件，则结束，否则重复第(3)～(6)步。

3. Boltzmann 机的学习算法

使用梯度下降法来计算权重 W_{ij} 的变化

$$W_{ij} = -\eta \frac{\partial G}{\partial W_{ij}} = -\eta \frac{1}{T}(P_{ij}^{+} - P_{ij}^{-}) \tag{3.55}$$

其中，P_{ij}^{+} 为网络输入、输出固定并对各种分布平均后，出现 S_i 和 S_j 同时为 1 的概率；P_{ij}^{-} 为网络仅固定输入时相应的概率，G 为两概率分布的测度。

经过反复调整 W_{ij}，直至 $P_{ij}^{+} = P_{ij}^{-}$，即 $\Delta W_{ij} = 0$ 时的 W_{ij} 即为所期望的权重。由于 Boltzmann 机模拟退火过程，虽然这种算法可以避免局部最小，但这种获得全局最小的收敛速度很慢，这是其不足。

3.2.8 深度信念网络

深度信念网络 (Deep Belief Network, DBN) 是 2006 年由辛顿教授等提出的，用于解决手写数字图像的识别问题。深度信念网络是以受限 Boltzmann 机为单元搭建的深层网络。

1. 受限 Boltzmann 机

受限 Boltzmann 机 (Restricted Boltzmann Machine, RBM) 是 1985 年由阿克雷(Ackley)和辛顿等提出的。受限就是把相互连接的 Boltzmann 机同层之间的连接取消，其

结构如图 3.22 所示,它包含一层可视层和一层隐层,两层的神经元彼此互连,同一层内神经元无连接。

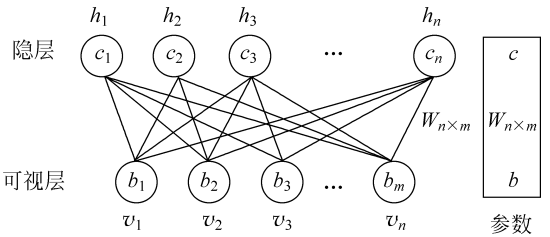


图 3.22 受限 Boltzmann 机

设可视层 v 为输入层,有 m 神经元,隐层 h 有 n 神经元,作为输出层,又称推断层,用作特征检测器。

在图 3.22 中,$\boldsymbol{W}_{n\times m}$ 为可视层与隐层间的连接权矩阵;$\boldsymbol{b}=(b_1,b_2,\cdots,b_m)$为可视结点的偏移向量;$\boldsymbol{c}=(c_1,c_2,\cdots,c_m)$为隐结点的偏移向量。可视层和隐层间的联合概率分布定义为

$$p(\boldsymbol{v},\boldsymbol{h})=\frac{1}{Z}\exp(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{h}+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{b}+\boldsymbol{h}^{\mathrm{T}}\boldsymbol{c}) \tag{3.56}$$

其中,Z 为归一化函数。对式(3.56)的优化目标是最小化式(3.56)的负对数:

$$E(\boldsymbol{p},\boldsymbol{h})=-\log P(\boldsymbol{v},\boldsymbol{h}) \tag{3.57}$$

2002 年,辛顿提出 RBM 的对比分歧(Contrastive Divergence,CD)快速算法,对参数 $\theta=(\boldsymbol{W},\boldsymbol{b},\boldsymbol{c})$最大似然求解,通常仅使用 $k=1$ 步 Gibbs 采样,就能得到足够好的近似解。针对一个样本的单步对比分歧算法的具体过程描述如下。

取一个训练样本$\boldsymbol{v}$,计算隐结点的概率,从中获取一个隐结点激活向量的样本 $\boldsymbol{h}$,计算$\boldsymbol{v}$和 $\boldsymbol{h}$ 的外积,称为“正梯度”;从 $\boldsymbol{h}$ 获取一个重构可视结点的激活向量样本$\boldsymbol{v}'$,再从$\boldsymbol{v}'$获得一个隐结点的激活向量样本 $\boldsymbol{h}'$,计算$\boldsymbol{v}'$和 $\boldsymbol{h}'$的外积,称为“负梯度”;使用正梯度和负梯度的差以一定的学习率 ε 更新权重为

$$\Delta W_{ij}=\varepsilon(\boldsymbol{v}\boldsymbol{h}^{\mathrm{T}}-\boldsymbol{v}'\boldsymbol{h}'^{\mathrm{T}}) \tag{3.58}$$

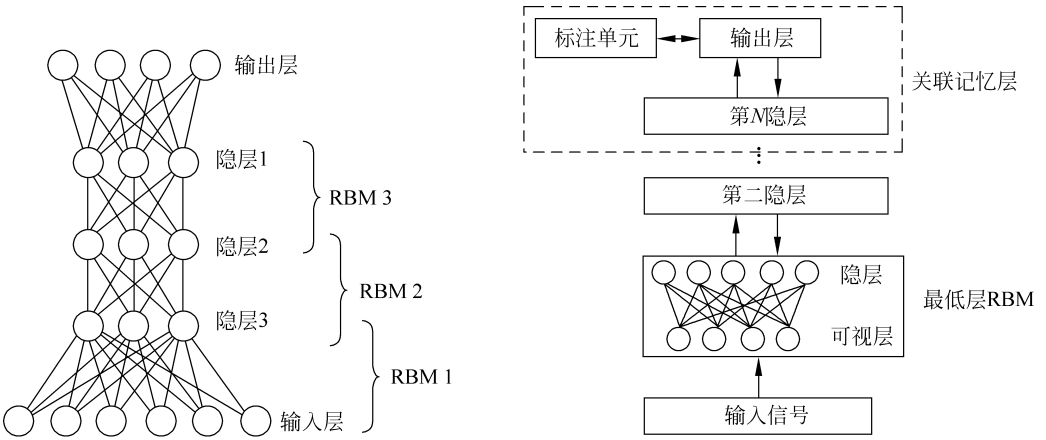
同样地,偏置 $\boldsymbol{b}$ 和 $\boldsymbol{c}$ 也可以使用类似的方法进行更新。

2. 深度信念网络的结构

深度信念网络(DBN)是由若干受限 Boltzmann 机(RBM)堆叠而成,即上一个 RBM 的隐层作为下一个 RBM 的可视层,上一个 RBM 的输出作为下一个 RBM 的输入,直到最后再连接一层分类层构成,如图 3.23 所示。

3. 深度信念网络的训练

DBN 的训练过程分为预训练和调优两个阶段。从低到高逐层采用无监督算法进行预



(a) 多个RBM堆叠成DBN的分层结构　　(b) 深度信念网络的结构

图 3.23　从受限 Boltzmann 机到深度信念网络的结构

训练，初始化的网络参数包括层与层之间的联结权重及各层神经元的偏置值，具体步骤如下：

(1) 低层(可视层)使用观测样本对 RBM 进行训练。

(2) 低层输出作为高层(隐层)的输入对 RBM 进行训练。

(3) 重复第(1)步和第(2)步，对所有的 RBM 进行训练，完成模型参数的初始化。

调优是在完成对模型参数初始化的预训练之后，再用传统学习算法对网络参数进行微调，达到进一步优化参数的目的。

深度信念网络采用了逐层初始化和整体反馈的方法，克服了深层网络难以训练的弊端。因此，它可以用于特征识别、特征分类及数据构造等。

3.2.9　卷积神经网络

卷积神经网络(Convolution Neural Network，CNN)是 1998 年由乐村等提出的[119]，用于手写字符的识别。为了解决全连接网络图像识别面临的许多问题，乐村等提出局部连接、局部感受野、权重共享、空间或时间子采样的新思想，设计了卷积神经网络 LeNet-5。

1. 卷积神经网络结构及设计原理

卷积神经网络是一种多层前向神经网络，如图 3.24 所示，除输入层外，有 3 个卷积层，2 个子采样层(池化层)，1 个全连接层和输出层，共 7 层。输入的原始图像大小是 32×32 像素，卷积层用来提取不同的图像特征，子采样层用作降维。卷积层和子采样层的输出均为特

征图。全连接层是一个分类器,输出层给出的是图像在各类别下的预测概率。

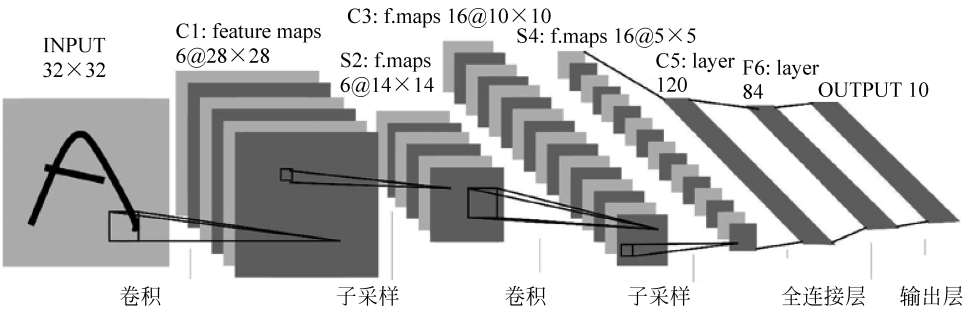


图 3.24 卷积神经网络 LeNet-5 的结构

卷积神经网络设计局部连接和权重共享的思想,源于自然图像存在相邻局部区域的统计特性具有相似性,因此,每个神经元就没必要全连接对全局图像进行感知,只需要局部连接对局部进行感知,从局部区域学习到的图像特征,同样适合于其他相邻的局部区域。这个局部区域称为卷积核(权矩阵)的局部感受野,这样的权矩阵就可以在其他区域共享了。

2. 卷积层与卷积运算

卷积神经网络的每一层都由多个二维向量组成,每个向量由多个独立神经元组成。每个二维向量可以视为一张图像,使用一个权矩阵表示单个像素与其邻域像素之间的关系。卷积层的作用是通过卷积运算提取输入图像的特征。

卷积是一种积分运算,用于求两个重叠区域面积。卷积运算就是用卷积核(权矩阵)作模板和一幅图像进行卷积,对于图像上的一个点,让模板的原点和该点重合,然后模板上的点和图像上对应的点相乘,再把各点的积相加,就得到了该点的卷积值。卷积运算是把一个点的像素用它周围点像素的加权平均代替,起到对图像线性滤波的作用。

卷积操作是将作为卷积核的权矩阵在输入特征图中,按从上到下、从左到右的顺序移动,从而得到整个图像的特征表示。

3. 池化层与池化操作

池化层用来降低数据的维数。类似于卷积操作,池化操作是将池化窗口在输入特征图中,按从上到下、从左到右的顺序移动。取池化窗口所覆盖的子矩阵元素的最大值或平均值作为输出的操作,分别称最大池化或平均池化。

上述的卷积运算和池化过程的示意如图 3.25 所示。输入图片首先利用卷积核(也称滤波器矩阵)$\boldsymbol{f}_x$ 对图像进行卷积运算,再赋予偏置项 $\boldsymbol{b}_x$,最后得到卷积层 $\boldsymbol{C}_x$ 的一个特征图。这个特征图被子采样后,先对每个邻域的 4 个像素求和,得到 1 个像素值,通过权重 $\boldsymbol{W}_{x+1}$ 对 x 加权,再加上偏置项 $\boldsymbol{b}_{x+1}$,最后利用 Sigmoid 激活函数得到几乎缩小为 1/4 倍的特征图 S_{x+1}。

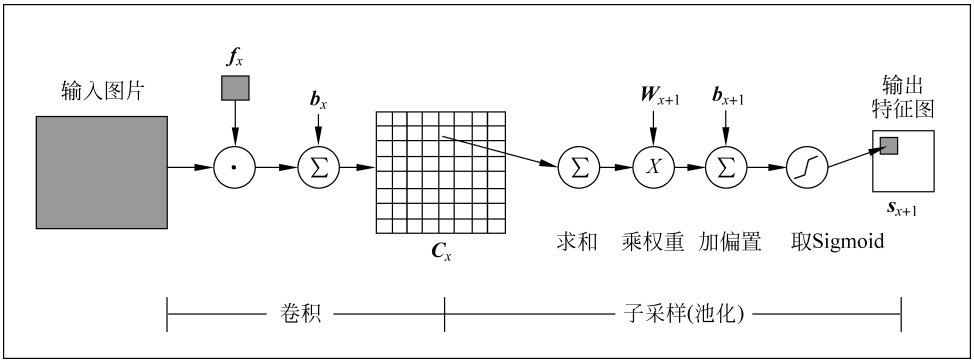


图 3.25　卷积和子采样过程示意图

卷积神经网络经过多个卷积层和池化层后，将获取图像的全部局部特征送给全连接层。

4. 全连接层

全连接层在整幅图像层面上整合卷积层（或者池化层）中局部的特征信息，将高维的大向量通过全连接变成合适维度的小向量，以形成便于分类的特征，并将这些特征映射到样本的标记空间中。

5. 卷积神经网络的训练

卷积神经网络的训练比起全连接神经网络的训练要复杂一些，但是训练的原理是一样的。都是利用链式求导计算损失函数对每个权重的偏导数（梯度），然后根据梯度下降公式更新权重。训练算法依然是反向传播算法。由于卷积神经网络的参数量较大，很容易发生过拟合，影响最终的测试性能。辛顿等人[120]提出通过在每次训练迭代中随机忽略一半的特征点来防止过拟合。此外还有一些改进方法，如动量法、权重衰变和数据增强等。

卷积神经网络 LeNet-5 已成为深度学习的经典模型，后来相继提出 AlexNet、ZFNet 和 VGGNet 等模型，主要应用在图像识别、人脸识别、计算机视觉、自然语言处理、机器学习等领域。

3.2.10　循环神经网络

循环神经网络（Recurrent Neural Network，RNN）是具有环路结构和记忆能力，能处理序列数据的一类动态神经网络。对循环神经网络的研究始于 20 世纪 80—90 年代，根据“循环”方式的不同和输入输出的变化，约旦（Jordan）、皮内达（Pineda）、威廉姆斯（Williams）等提出了多种 RNN 结构。

1. RNN 的结构与原理

循环神经网络的单元结构如图 3.26 所示,其中图左侧为包含输入单元、隐单元和输出单元的单元模型。如果去掉了隐单元的自身反馈环,它就变成了最普通的全连接神经网络。

图 3.26 中的右侧表示由单元模型按时间展开后的情况。网络在 t 时刻输入 x_t 后,隐单元的状态不仅取决于 s_t,还取决于 s_{t-1},s_t 和它的输出值 o_t 分别用下面的公式表示为

$$s_t = f(\boldsymbol{U} \cdot x_t + \boldsymbol{W} s_{t-1}) \tag{3.59}$$

$$o_t = g(\boldsymbol{V} \cdot s_t) \tag{3.60}$$

其中,$\boldsymbol{U}$ 为输入 x 的权矩阵;$\boldsymbol{V}$ 为输出层的权矩阵;g 和 f 分别为激活函数。

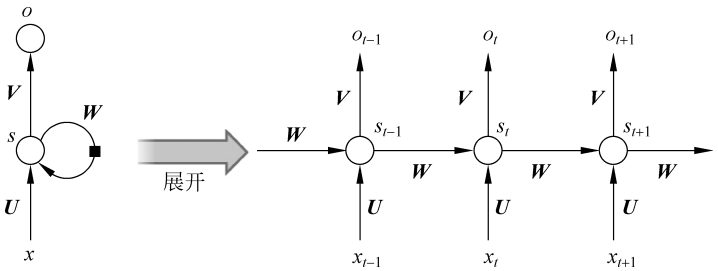


图 3.26 循环神经网络结构示意图

不难看出,输出层是一个全连接层,它的每个结点都和隐结点相连。这样能够通过获取输入层的输出和隐单元前一时刻的状态来计算当前时刻的隐单元输出,具有对过去信息进行记忆的功能。

2. RNN 的学习算法

1986 年,辛顿等人提出了一种循环神经网络训练参数的随时间反向传播(Back-Propagation Through Time,BPTT)算法。BPTT 算法可以看作反向传播算法由前馈神经网络向循环神经网络的推广,只不过 RNN 处理的是时间序列数据,所以是基于时间反向传播,故称随时间反向传播。

BPTT 算法将循环神经网络视为一个展开的多层前馈网络,其中“每一层”对应循环网络中的“每个时刻”。这样,循环神经网络就可以按照前馈网络中的反向传播算法进行计算参数梯度。在“展开”的前馈网络中,所有层的参数 $\boldsymbol{W}$、$\boldsymbol{U}$、$\boldsymbol{V}$ 都是共享的,训练的目的就是沿着需要优化参数的负梯度方向不断地寻找更优的点,直至收敛。

BPTT 算法对循环层的训练的具体步骤如下。

(1) 随时间前向传播计算每个神经元的输入值。

(2) 随时间反向传播计算每个神经元的误差值,将误差项向上一层传播。

(3) 计算每个权重的梯度。

(4) 用梯度下降的误差后向传播算法更新权重。

3. LSTM 网络的结构

长短时间记忆单元(Long Short-Term Memory,LSTM)的特殊结构循环神经网络是1997 年由德国的霍克赖特(Hochreiter)和施密特胡伯(Schmidhuber)提出的,它通过控制记忆状态时间对长短来解决 BPTT 算法当输入序列长时间存在的梯度消失问题。

LSTM 网络由多个带有 3 个门的单元模块组成,一个单元结构如图 3.27 所示,其中黑圆点表示门,含曲线的圆圈表示激活函数,中间的大圆圈表示该单元的结点。

每个单元模块包含输入门、输出门及遗忘门,它们分别通过不同的取值 1 或 0 对当前结点的信息进行控制。例如,输入门通过取值为 1(或 0)控制信息允许(或不允许)输入当前结点;输出门通过取值为 1(或 0)控制当前结点信息允许(或不允许)传递给下个结点;遗忘门通过取值为 1(或 0)控制当前结点允许(或不允许)保留历史时刻信息。

LSTM 的训练算法仍然是反向传播算法。LSTM 网络能够对数据点所在时刻的"历史"信息进行参考,更有利于分析数据。

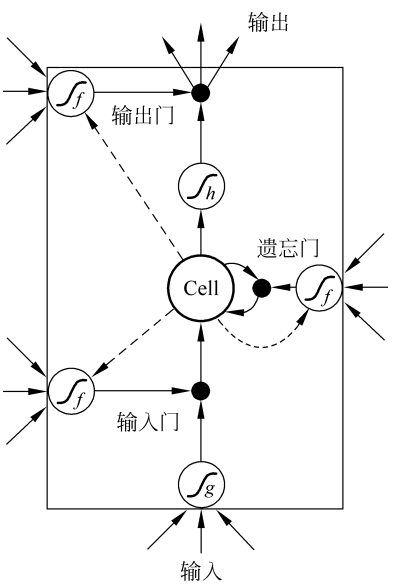


图 3.27　一个 LSTM 网络的单元结构

4. 双向循环神经网络的结构

双向循环神经网络(Bidirectional Recurrent Neural Network,Bi-RNN)是由两个独立并结构完全对称的循环神经网络叠加在一起组成的,其输出也是由这两个循环神经网络的输出拼接而成,其结构如图 3.28 所示。

同一个输入会同时提供给两个方向相反的循环神经网络,两个独立的网络独立进行计算,各自产生该时刻的新状态和输出;双向循环神经网络的最终输出是这两个单向循环神经网络输出的简单拼接。这样可以在同一网络结构中直接用两个相反时间方向的输入信息来减少代价函数的误差,而不需要额外的算法处理"未来"数据信息,能对输入数据的背景信息更加有效的利用,较前面的传统循环神经网络更为简便,并提高了识别率。

5. 深层循环神经网络

RNN 的环路结构使得在神经单元之间既有内部反馈连接,又有前馈连接。RNN 的内

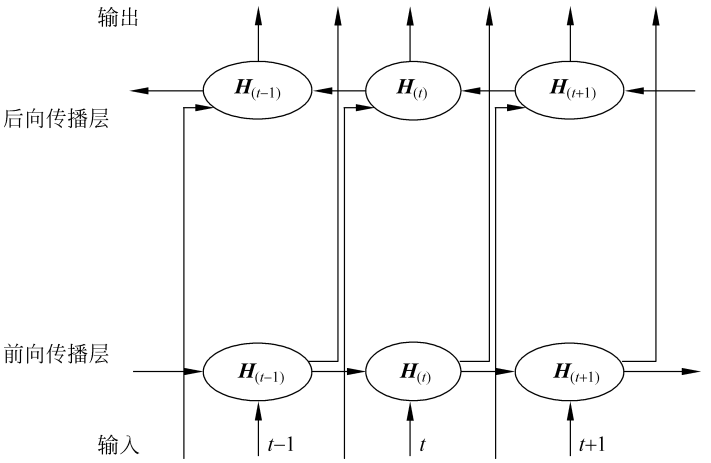


图 3.28 双向循环神经网络结构

部状态可以展示动态时序行为,可以利用它内部的记忆来处理任意时序的输入序列。RNN比前馈神经网络更加符合生物神经网络的结构,具有更强的动态行为和计算能力。

前面介绍的循环神经网络只有一个循环层,为了增强网络的信息表达和处理能力,在网络中设置多个循环层,将每层循环神经网络的输出传给下一层进行处理,这样网络就称为深层循环神经网络。深层循环神经网络广泛用于语音识别、语言模型以及自然语言生成等领域。

3.2.11 递归神经网络

随着大数据和云计算技术的发展,需要设计更为复杂的网络以便在时间和空间维度上解决表征、处理语音识别、图像识别等复杂问题。于是反馈神经网络的结构变得越来越复杂。复杂结构的网络在训练、学习过程中,通常采用递归的结构,一般把这类具有复杂结构的反馈网络称为递归神经网络。

1. 递归神经网络的结构原理

递归神经网络(Recursive Neural Network,RNN)是利用树状神经网络结构来递归构造更为复杂的深度神经网络。递归神经网络的结构上有利于在空间维度上展开,便于处理图像识别等问题。

根据实际应用场合对输入和输出序列数量的要求不同,RNN的结构有多种形式:单入单出,用于图像分类场景等;单入多出,可用于图像自动字幕等;多入单出,可用于文本情感分析等;多入多出,可用于翻译或聊天对话场景等。

下面介绍 RNN 怎样把一个树结构信息编码映射为一个向量。

设递归神经网络的输入有两个子结点，输出为这两个子结点编码后产生的父结点，父结点的维度和每个子结点相同，它们都用向量表示。子结点的每个神经元和父结点的每个神经元两两相连。两个子结点和两个父结点就组成一个全连接神经网络，用矩阵表示联结权重。

把产生的父结点向量和其他子结点向量再作为网络的输入，再次产生它们的父结点，如此递归下去，直至整棵树处理完毕。最终，将得到根结点的向量，它是对整棵树的表示。这样就实现了把树结构信息映射为一个向量。

2. 递归神经网络的训练

递归神经网络的训练算法和循环神经网络类似，都是采用辛顿等人在 1986 年提出的一种多层前馈网络训练的 BPTT 算法。但两者训练的区别在于，循环神经网络是将误差从当前时刻反向传播到初始时刻，而递归神经网络要将误差从根结点反向传播到各个子结点。

3. 递归神经网络与循环神经网络的关系

循环神经网络(Recurrent Neural Network，RNN)与递归神经网络(Recursive Neural Network，RNN)二者的英文缩写都是 RNN。英文中的 recurrent 和 recursive 都有递归的、循环的意思，而译成中文时前者译为循环，而后者译为递归。

循环神经网络指的是时间上的循环，用于在时间维度上展开，处理时间序列结构的信息。这样的信息在时间维度从前往后的传递和积累。递归神经网络指的是结构上的递归，用于在空间维度上展开，处理树/图类结构的信息。

用循环神经网络来建模的话，就是假设句子后面的词的信息和前面的词有关，而用递归神经网络来建模的话，就是假设句子是由几个句成分组成的一个树状结构，而每个部分又可以再分成几个小部分，即某一部分的信息由它的子树的信息组合而来，整句话的信息由组成这句话的几个部分组合而来。

递归神经网络和循环神经网络具有不同的计算图，递归神经网络的计算图的结构为一个深层树(无环图)，而不是循环神经网络的计算图为链式结构(有环图)。

除了上述介绍的神经网络模型外，还有许多种类的神经网络模型，如基于自适应共振理论(Adaptive Resonance Theory，ART)的自组织神经网络，基于非线性动力学稳定吸引子理论的双向联想记忆网络(Bidirectional Associative Memory，BAM)，基于细胞自动机理论的细胞神经网络(Cellular Neural Network，CNN)，模糊神经网络(Fuzzy Neural Network，FNN)，以及混沌神经网络(Chaotic Neural Network，CNN)等，在此不再赘述。

3.3 基于神经网络的系统辨识

3.3.1 神经网络的逼近能力

1989年,罗伯特·赫希特·尼尔森(Robert Hecht-Nielson)证明了对于任何在闭区间内的一个连续函数都可以用一个隐层的BP网络来逼近,因而一个三层的BP网络可以完成任意的 n 维到 m 维的映射。这个定理的证明是以数学上维尔斯特拉斯(Weierstrass)的以下两个逼近定理为依据的。

定理 3.2 任意给定一个连续函数 $g\in C(a,b)$ 及 $\varepsilon>0$,存在一个多项式 $P(x)$,使 $|g(x)-P(x)|<\varepsilon$,对每个 $x\in[a,b]$ 成立。

定理 3.3 任意给定一个函数 $g\in C_{2\pi}$($C_{2\pi}$ 是以 2π 为周期的连续函数)及 $\varepsilon>0$,存在三角函数多项式 $T(x)$,使得 $|g(x)-T(x)|<\varepsilon$,对每个 $x\in\mathbf{R}$ 成立。

推理 3.1 在 n 维空间中,任一向量 $\boldsymbol{x}$ 都可以由基集 $\{e_i\}$ 表示,$\boldsymbol{x}=C_1e_1+C_2e_2+\cdots+C_ne_n$,同样在有限区间内 $x\in[a,b]$ 的一个函数 $g(x)$,可以用一个正交函数序列 $\{\phi_i(x)\}$ 来表示。如果基函数可以扩展到任意大,那么

$$g(x)=C_1\phi_1(x)+C_2\phi_2(x)+\cdots+C_n\phi_n(x) \tag{3.61}$$

如果正交基函数是有限项,那么

$$g(x)=C_1\phi_1(x)+C_2\phi_2(x)+\cdots+C_n\phi_n(x)+\varepsilon \tag{3.62}$$

$\{\phi_i(x)\}$ 是正交的,可以用傅里叶级数的三角函数展开,$C_1,C_2,\cdots,C_n$ 为傅里叶级数的系数。

利用推理3.1,对于一个任意给定的一维连续函数 $g(x)$,$x\in[0,1]$,可以用一个傅里叶级数来近似,表示为

$$g_{\mathrm{F}}(x)=\sum_k C_k\exp(2\pi ikx) \tag{3.63}$$

其中

$$C_k=\int_{[0,1]}g(x)\exp(-2\pi ikx)\mathrm{d}x \tag{3.64}$$

则有 $|g(x)-g_{\mathrm{F}}(x)|<\varepsilon$,对每个 x 成立。

进一步考虑 $\boldsymbol{x}$ 为一个 n 维空间的向量,在 $[0,1]^n\in\mathbf{R}^n$ 进行映射 g':$[0,1]^n\in\mathbf{R}^n\to\mathbf{R}$,如果积分 $\int_{[0,1]^n}|g'(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x}$ 存在,那么根据傅里叶级数理论,仍旧存在一个级数:

$$\begin{aligned}g'_{\mathrm{F}}(\boldsymbol{x},N,g')&=\sum_{k_1=-N}^{N}\sum_{k_2=-N}^{N}\cdots\sum_{k_n=-N}^{N}C_{k_1k_2\cdots k_n}\exp\Big(2\pi i\sum_{i=1}^{n}k_ix_i\Big)\\&=\sum_k C_k\exp(2\pi i\,\boldsymbol{k}\cdot\boldsymbol{x})\end{aligned}$$

$$C_{k_1,k_2,\cdots,k_n}=C_k=\int_{[0,1]^n}g'(\boldsymbol{x})\cdot\exp(-2\pi ik\cdot\boldsymbol{x})\mathrm{d}\boldsymbol{x} \tag{3.65}$$

当 $N\to\infty$ 时，满足

$$\lim_{N\to\infty}\int_{[0,1]^n}|g(\boldsymbol{x})-g'_{\mathrm{F}}(\boldsymbol{x},N,g')|\mathrm{d}\boldsymbol{x}=0 \tag{3.66}$$

现考虑对一个任意多维函数的映射，给定一个函数 $h(\boldsymbol{x})$，$\boldsymbol{x}\in\mathbf{R}^n$，$[0,1]^n\subset\mathbf{R}^n\to\mathbf{R}^m$ 其 $h(\boldsymbol{x})=[h_1(\boldsymbol{x}),h_2(\boldsymbol{x}),\cdots,h_m(\boldsymbol{x})]^{\mathrm{T}}$，则 $\boldsymbol{h}$ 中的每一个分量也都可以用式(3.63)中的傅里叶级数来近似，那么可以得到下面的定理。

定理 3.4(映照定理) 给定任一个 $\varepsilon>0$，一个连续函数向量 $\boldsymbol{h}$，其向量中的每个元素满足 $\int_{[0,1]^n}|h_i(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x}$ 存在，$\boldsymbol{h}:[0,1]^n\subset\mathbf{R}^n\to\mathbf{R}^m$ 必定存在一个三层 BP 神经网络来逼近函数 $\boldsymbol{h}$，使逼近误差在 ε 之内。(证明略)

神经网络的万能逼近能力表明它具有从输入到输出的非常强的非线性映射能力。

3.3.2 神经网络系统辨识的原理

系统辨识是对难以通过机理和试验方法建模的复杂对象进行建模的一种方法。扎德把辨识定义为："辨识就是在系统输入和输出观测数据的基础上，从一组给定的模型中，确定一个与被识别的系统等价的模型。"这个定义指出了辨识应具有以下 3 个要素。

(1) 输入输出数据：能够观测到的按时间顺序排列的系统输入输出数据，简称为时序数据。

(2) 模型类：明确给定所要辨识的系统属于哪一类系统，因为属性完全不同的系统可能具有相同的模型结构。

(3) 等价准则：从一类模型中按所给定的性能等价准则，选择一个与实际系统最为接近的模型。等价准则就是用以衡量模型与实际系统接近程度的标准，一般表示为误差函数的泛函。

基于神经网络的系统辨识就是选择一个适当拓扑结构的神经网络模型，使该网络从待辨识的实际动态系统中不断地获得输入输出数据，神经网络通过学习算法自适应地调节神经元间的联结强度，从而获得利用神经网络逐渐逼近实际动态系统的模型(正模型)或逆模型(如果系统是可逆的)。这样的动态系统模型实际上是隐含在神经网络的权矩阵中。

3.3.3 基于 BP 网络的非线性系统模型辨识

多层前馈网络是系统辨识中常用的网络，下面介绍一种基于 BP 网络的非线性动态系统模型辨识算法[43]。

1. 前馈网络的结构设计

为简单起见,考虑单输入单输出动态系统

$$y(k+1)=f[y(k),y(k-1),\cdots,y(k-n+1);\ u(k),u(k-1),\cdots,u(k-m+1)] \tag{3.67}$$

其中,$u(k)$,$y(k)$分别为系统的输入、输出;m,n 分别为输入、输出的阶次。

辨识上述系统选用三层前馈网络:输入层神经元的数目为 $n_{\mathrm{I}}\geqslant m+n+1$;输出层神经元的数目 n_{O} 应为待辨识的输出维数,在此 $n_{\mathrm{O}}=1$;隐层的神经元数目 n_{H} 一般取 $n_{\mathrm{H}}>n_{\mathrm{I}}$。

如果对象的阶次 n,m 已知,那么 BP 网络的输入向量为

$$\boldsymbol{X}(k)=[x_1(k),x_2(k),\cdots,x_{n_{\mathrm{I}}}(k)]^{\mathrm{T}} \tag{3.68}$$

$$x_i(k)=\begin{cases}y(k-i), & 1\leqslant i\leqslant n\\ u(k-i-n+1), & n+1\leqslant i\leqslant n_{\mathrm{I}}\end{cases} \tag{3.69}$$

如果对象的阶次 n,m 未知,那么可通过选择 n,m 的不同组合,由好的性能指标决定。

2. 基于 BP 网络的系统辨识算法

设输入层到隐层的加权阵为$[v_{ji}]$,隐层到输出层的加权阵为$[w_i]$。对于设定的单输入单输出系统,神经网络从输入层到隐层的输入输出关系为

$$\mathrm{net}_i(k)=\sum_{j=0}^{n_{\mathrm{I}}}v_{ji}x_j(k) \tag{3.70}$$

$$I_i(k)=H[\mathrm{net}_i(k)] \tag{3.71}$$

$$H[x]=(1-\mathrm{e}^{-x})/(1+\mathrm{e}^{-x}) \tag{3.72}$$

其中,$x_0=1$ 为阈值对应的状态。

从隐层到输出层的输入输出关系为

$$\hat{y}(k)=\sum_{i=0}^{n_{\mathrm{H}}}W_iI_i(k) \tag{3.73}$$

其中,$I_0=1$ 为阈值对应的状态。

BP 网络的学习算法采用广义 δ 规则,使性能指标

$$J=\frac{1}{2}[y(k)-\hat{y}(k)]^2 \tag{3.74}$$

达到最小。为提高网络学习速度,采用带有惯性项的 δ 规则为

$$\Delta W_i(k)=a_1e(k)I_i(k)+a_2\Delta W_i(k-1) \tag{3.75}$$

$$\Delta v_{ji}(k)=a_1e(k)H'[\mathrm{net}_i(k)]W_i(k)x_i(k)+a_2\Delta v_{ji}(k-1) \tag{3.76}$$

$$e(k)=y(k)-\hat{y}(k) \tag{3.77}$$

$$H'[\mathrm{net}_i(k)]=\mathrm{net}_i(k)[1-\mathrm{net}_i(k)] \tag{3.78}$$

其中，$i=1,2,\cdots,n_{\mathrm{H}}$；$j=1,2,\cdots,n_{\mathrm{I}}$。

3. 基于 BP 网络的系统辨识步骤

(1) 初始化权重 $v_{ji}(0)$，$W_i(0)$为一小的随机值。

(2) 选择适当形式的输入信号 $u(k)$，如二进制伪随机序列，加入到系统式(3.67)。

(3) 采集输出信号 $y(k)$(若为仿真时，$y(k)$由式(3.67)计算)。

(4) 根据式(3.68)、式(3.69)构成输入向量 $\boldsymbol{X}(k)$，并计算误差 $e(k)$。

(5) 根据式(3.75)～式(3.77)修改加权重 $W_i(k)$，$v_{ji}(k)$。

(6) 将 $k\to k+1$，返回第(2)步。如果是离线辨识，需按预先给定的允许误差 ε 判断辨识算法的终止条件为

$$|e(k)|=|y(k)-\hat{y}(k)|<\varepsilon \tag{3.79}$$

3.4　基于神经网络的智能控制

3.4.1　神经控制的基本原理

控制系统的目的在于通过确定适当的控制量输入，使得系统获得期望的输出特性。图 3.29(a)给出一般反馈控制系统的原理图，图 3.29(b)是采用神经网络替代图 3.29(a)中的控制器完成同样的控制任务。

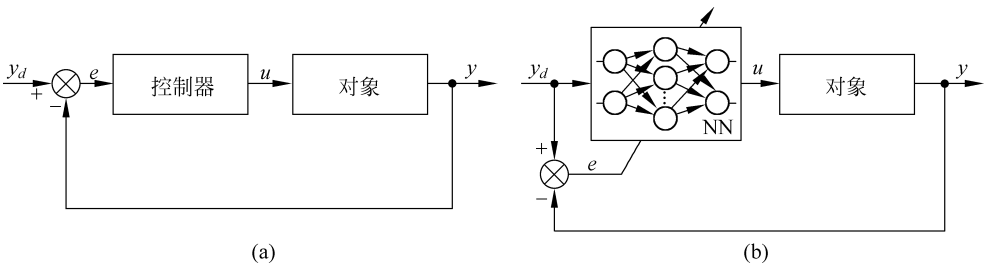


图 3.29　反馈控制与神经控制的对比

下面分析神经网络是如何工作的。设被控制对象的输入 u 和系统输出 y 之间满足如下非线性函数关系

$$y=g(u) \tag{3.80}$$

控制的目的是确定最佳的控制量输入 u，使系统的实际输出 y 等于期望的输出 y_d。在该系统中，把神经网络的功能看作从输入到输出的某种映射，或称函数变换，并设它的函数关系为

$$u = f(y_d) \tag{3.81}$$

为了满足系统输出 y 等于期望的输出 y_d，将式(3.81)代入式(3.80)，可得

$$y = g[f(y_d)] \tag{3.82}$$

显然，当 $f(\cdot)=g^{-1}(\cdot)$ 时，满足 $y=y_d$ 的要求。

由于要采用神经网络控制的被控对象一般是复杂的且多具有不确定性，因此非线性函数 $g(\cdot)$ 是难以建立的，可以利用神经网络具有逼近非线性函数的能力来模拟 $g(\cdot)$。尽管 $g(\cdot)$ 的形式未知，但根据系统的实际输出 y 与期望输出 y_d 之间的误差，通过神经网络学习算法调整神经网络联结权重直至误差

$$e = y_d - y \to 0 \tag{3.83}$$

这样的过程就是神经网络逼近 $g^{-1}(\cdot)$ 的过程，实际上是对被控对象的一种求逆过程。由神经网络的学习算法实现逼近被控对象逆模型的过程，就是神经网络实现直接控制的基本思想。

3.4.2 基于神经网络智能控制的类型

在控制系统中，应用神经网络的非线性映射能力可对难以精确描述的复杂非线性对象进行建模，或充当控制器，或优化计算，或进行推理，或故障诊断等，或同时兼有上述某些功能的适当组合等。

基于神经网络的智能控制本书指只由神经网络单独进行控制或由神经网络同其他智能控制方式相融合的控制的统称，属于这类控制的主要有以下形式。

1. 神经网络直接反馈控制

这是只使用神经网络直接实现智能控制的一种方式。在这种控制方式中，神经网络直接作为控制器，利用反馈等算法实现自学习控制。

2. 神经网络专家系统控制

专家系统善于表达知识和逻辑推理，神经网络长于非线性映射和直觉推理，将二者相结合发挥各自的优势，就会获得更好的控制效果。

图 3.30 是一种神经网络专家系统的结构方案，这是一种将神经网络和专家系统相结合用于智能机器人的控制系统结构。EC 是对动态系统 P 进行控制的专家控制器。神经网络控制器 NC 将接受小脑模型神经网络 CMAC 的训练，每当运行条件变化使神经控制器性能下降到某一限度时，运行监控器 EM 将调整系统工作状态，使神经网络处于学习状态，此时 EC 将保证系统的正常运行。该系统运行共有 3 种状态：EC 单独运行，EC 和 NC 同时运行，NC 单独运行。监控器 EM 负责管理它们之间运行的切换。

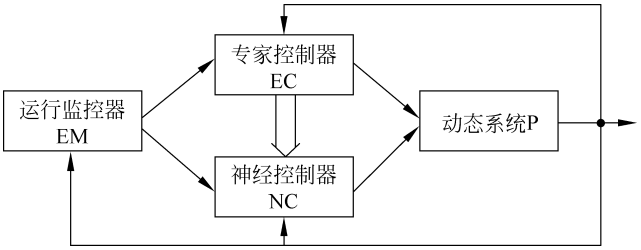


图 3.30　神经网络专家系统

对复杂系统可采用递阶控制结构，如图 3.31(a)所示，下层为 NC，上层为 EC，利用 NC 的映射能力和运算能力进行实时控制，EC 则用于知识推理、决策、规划、协调。图 3.31(b)表示一种分级结构，EC1 帮助 NC 进行训练等；NC 用于决策、求解问题；EC2 用来解释 NC 的输出结果并驱动执行机构对系统 P 进行控制。图 3.31(c)是利用神经网络控制完成专家系统中最耗费时间的模式匹配工作，以加速专家系统的执行。由上面的结构不难看出，神经控制和专家系统的结合具有这样的特点：在分层结构中，EC 在上层，NC 在下层；在分级结构中，EC 在前级，NC 在后级。

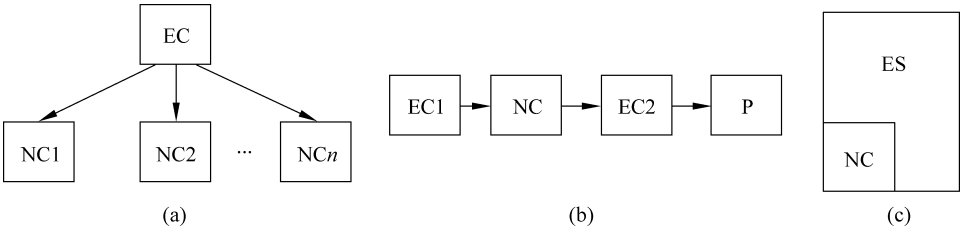


图 3.31　神经网络专家系统的递阶分级结构

3. 神经网络模糊逻辑控制

模糊系统善于直接表示逻辑，适于直接表示知识，神经网络长于学习通过数据隐含表达知识。前者适于自上而下的表达，后者适于自下而上的学习过程，二者存在一定的互补、关联性。因此，它们的融合可以取长补短，可以更好地提高控制系统的智能性。神经网络和模糊逻辑相结合有以下几种方式。

1）用神经网络驱动模糊推理的模糊控制

这种方法是利用神经网络直接设计多元的隶属函数，把神经网络作为隶属函数生成器组合在模糊控制系统中。

2）用神经网络记忆模糊规则的控制

通过一组神经元不同程度的兴奋表达一个抽象的概念值，由此将抽象的经验规则转化成多层神经网络的输入输出样本，通过神经网络如 BP 网络记忆这些样本，控制器以联想记

忆方式使用这些经验进行控制,这在一定意义上模拟了人的联想记忆思维方式。

3) 用神经网络优化模糊控制器的参数

在模糊控制系统中对控制性能影响的因素除上述的隶属函数、模糊规则外,还有控制参数,如误差、误差变化的量化因子及输出的比例因子,利用神经网络的优化计算功能可优化这些参数,改善模糊控制系统的性能。

4. 神经网络滑模控制

变结构控制可以视为模糊控制的特例,因此它属于智能控制的范畴。将神经网络和滑模控制相结合就构成神经网络滑模控制。这种方法将系统的控制或状态分类,根据系统和环境的变化进行切换和选择,利用神经网络具有的学习能力,在不确定的环境下通过自学习来改进滑模开关曲线,进而改善滑模控制的效果。

3.4.3 基于传统控制理论的神经控制

将神经网络作为传统控制系统中的一个环节或多个环节,用来充当辨识器,或对象模型,或控制器,或估计器,或优化计算等。这种方式很多,常见的一些方式归纳如下。

1. 神经逆动态控制

设系统的状态观测值为 $x(t)$,它与控制信号 $u(t)$的关系为 $x(t)=F(u(t),x(t-1))$,F 可能是未知的,假设 F 是可逆的,即 $u(t)$可从 $x(t)$,$x(t-1)$求出,通过训练神经网络的动态响应为 $u(t)=H(x(t),x(t-1))$,H 即为 F 的逆动态。

2. 神经 PID 控制

将神经元或神经网络和常规 PID 控制相结合,根据被控对象的动态特性变化情况,利用神经元或神经网络的学习算法,在控制过程中对 PID 控制参数进行实时优化调整,以达到在线优化 PID 控制性能的目的。上述这样的复合控制形式统称为神经元 PID 控制或神经 PID 控制。

3. 模型参考神经自适应控制

在传统模型参考自适应控制系统中,利用神经网络充当对象模型,或充当控制器,或充当自适应机构,或优化控制参数,或兼而有之等,这样的系统统称为模型参考神经自适应控制。

4. 神经自校正控制

这种控制结构的一种形式如同前面介绍的双神经网络间接学习控制结构。对于单神经

网络可以有如图 3.32 所示的控制结构形式，评价函数一般取为 $e=y_d-y$，或采用下述形式：

$$e(t)=\boldsymbol{M}_y[y_d(t)-y(t)]+\boldsymbol{M}_u u(t) \tag{3.84}$$

其中，$\boldsymbol{M}_y$ 和 $\boldsymbol{M}_u$ 为适当维数的矩阵。该方法的有效性在水下机器人姿态控制中得到了证实。

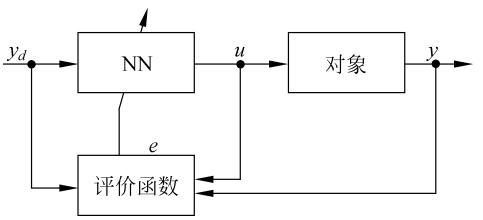


图 3.32　神经自校正控制的一种结构

此外，神经网络和传统控制的结合形式，还有神经内膜控制、神经预测控制、神经最优决策控制等，不再赘述。

3.5　神经元 PID 控制

将神经网络和传统 PID 控制融合，主要是通过神经网络学习算法在线优化 PID 控制器的控制参数，与通过模糊推理在线优化 PID 控制参数是类似的。将单个神经元和 PID 控制融合构成所谓的神经元 PID 控制，由于控制算法相对简单，所以获得了较多的应用。

有关通过神经网络学习算法在线优化 PID 控制器的控制参数的内容放在本书第 7 章介绍。本节介绍一般的神经元 PID 控制和自适应神经元 PID 控制两种形式。

3.5.1　神经元 PID 控制

基于单个神经元的 PID 控制系统如图 3.33 所示，简称为神经元 PID 控制。其中，神经元输入的 3 个状态变量 $x_i(t)(i=1,2,3)$ 分别为

$$\begin{cases} x_1(t)=e(t) \\ x_2(t)=\sum_{i=0}^{t} e(i)T \\ x_3(t)=\Delta e(t)/T \end{cases} \tag{3.85}$$

控制器输出为

$$u(t)=K\sum_{i=1}^{3} w_i(t)x_i(t)/\|\boldsymbol{W}\| \tag{3.86}$$

$$\|\boldsymbol{W}\|=\sum_{i=1}^{3}|w_i(t)| \tag{3.87}$$

$$u_g(t)=U_{\max}\frac{1-e^{-u(t)}}{1+e^{-u(t)}} \tag{3.88}$$

式中，$U_{\max}$ 为最大控制量。

控制系统的性能指标为

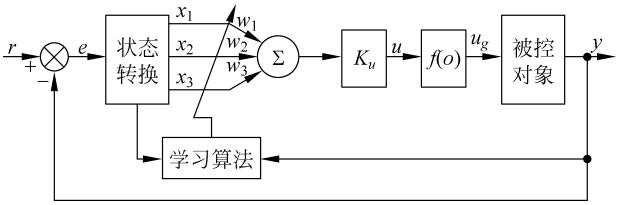


图 3.33 单个神经元 PID 控制系统

$$J(t)=\frac{1}{2}[r(t)-y(t)]^2=\frac{1}{2}e^2(t) \tag{3.89}$$

根据上述性能指标,控制器的加权重学习算法为

$$w_i(t+1)=w_i(t)+\Delta w_i(t) \tag{3.90}$$

$$\Delta w_i(t)=-d_i\frac{\partial J}{\partial w_i(t-1)}=-d_i\frac{\partial J}{\partial y(t)}\frac{\partial y(t)}{\partial u_g(t-1)}\frac{\partial u_g(t-1)}{\partial u(t-1)}\frac{\partial u(t-1)}{\partial w_i(t-1)} \tag{3.91}$$

由于 $\|\boldsymbol{W}\|$ 变化比较平缓,在求导过程中,可认为 $\|\boldsymbol{W}\|$ 近似为一常数,故有

$$\Delta w_i(t)=d_iU_{\max}[1-U_g^2(t-1)/U_{\max}^2]\times e(t-1)x_i(t-1)\frac{\partial y(t)}{\partial u_g(t-1)} \tag{3.92}$$

在对象 $\partial y(t)/\partial u_g(t-1)$ 未知时,有几种近似处理方法:一是利用符号信息 $\operatorname{sgn}[\partial y(t)/\partial u_g(t-1)]$ 近似 $\partial y(t)/\partial u_g(t-1)$;另一个方法是利用差分近似导数

$$\frac{\partial y(t)}{\partial u_g(t-1)}\approx\frac{y(t)-y(t-1)}{u_g(t-1)-u_g(t-2)} \tag{3.93}$$

假定 $\operatorname{sgn}[\partial y(t)/\partial u_g(t-1)]=1$,下面分析一下闭环系统的稳定性。取 Lyapunov 函数为 $V(t)=\frac{1}{2}e^2(t)$,则

$$\Delta V(t)=\frac{1}{2}e^2(t+1)-\frac{1}{2}e^2(t) \tag{3.94}$$

$$e(t+1)=e(t)+\frac{\partial e(t)}{\partial \boldsymbol{W}}\Delta\boldsymbol{W}^{\mathrm{T}}=e(t)+\Delta e(t) \tag{3.95}$$

将上式代入式(3.94),可得

$$\Delta V(t)=\frac{1}{2}[2e(t)\Delta e(t)+\Delta^2 e(t)] \tag{3.96}$$

设 $\boldsymbol{P}=[\partial e(t)/\partial w_1,\partial e(t)/\partial w_2,\partial e(t)/\partial w_3]^{\mathrm{T}}$,$\boldsymbol{D}=\operatorname{diag}\{d_1,d_2,d_3\}$,则有

$$\Delta e(t)=-e(t)\boldsymbol{P}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{P} \tag{3.97}$$

于是有

$$\Delta V(t)=-\frac{1}{2}[e(t)\boldsymbol{P}]^{\mathrm{T}}(2\boldsymbol{D}-\boldsymbol{D}\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{D})[e(t)\boldsymbol{P}] \tag{3.98}$$

当满足

$$\boldsymbol{0} < \boldsymbol{D} < 2(\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}})^{-1} \tag{3.99}$$

时，有 $\Delta V(t)<0$，整个闭环系统是稳定的。因此，当步长 d_i 取得比较小时，有 $\Delta V(t)<0$，这表明随 t 增大，$e(t)\to 0$，控制算法收敛。

由于 $\boldsymbol{P}$ 不是一个常数，故式(3.99)只能从性质上说明步长 d 的取值范围，实际步长可从实验中确定。

3.5.2 自适应神经元 PID 控制

自适应神经元控制的学习算法为

$$\begin{cases} u(t)=k\sum_{i=1}^{n} w_i(t)x_i(t) \\ w_i(t+1)=w_i(t)+d(r(t)-y(t))x_i(t) \end{cases} \tag{3.100}$$

式中，k，d 是待选定的常数；加权重也可以由式

$$w_i(t+1)=w_i(t)+d(r(t)-y(t))u(t)x_i(t) \tag{3.101}$$

确定；状态 $x_i(t)(i=1,2,\cdots,n)$ 可选 $n=3$

$$x_1(t)=r(t),\quad x_2(t)=r(t)-y(t),\quad x_3(t)=\Delta x_2(t)$$

为确保递进式学习的收敛性，对式(3.100)进行规范化后可得

$$\begin{cases} u(t)=k\left[\sum_{i=1}^{n} w_i(t)x_i(t)\right]\Big/\left[\sum_{i=1}^{n} w_i(t)\right] \\ w_i(t+1)=w_i(t)+d(r(t)-y(t))x_i(t) \end{cases} \tag{3.102}$$

一般来说，d 作为学习速率，希望取大一些；作为比例系数 k 可取为 1 或一较小的数值，观察学习控制效果再确定 k 值。

由图 3.34 可知，自适应神经元产生的控制信号由 3 部分组成：前馈比例控制 $u_1(t)$、反馈比例控制 $u_2(t)$、反馈微分控制 $u_3(t)$，即

$$u(t)=k(u_1(t)+u_2(t)+u_3(t)) \tag{3.103}$$

式中，$u_1(t)=w_1'(t)r(t)$；$u_2(t)=w_2'(t)e(t)=w_2'(t)(r(t)-y(t))$；$u_3(t)=w_3'(t)\Delta e(t)$。

$$w_i'(t)=w_i(t)\Big/\left[\sum_{i=1}^{3} w_t(t)\right],\quad i=1,2,3 \tag{3.104}$$

这是一种多层次多模式的控制结构，集前馈与反馈、开环与闭环于一体，互为关联，互为补偿。神经元控制作用的分离如图 3.35 所示。图 3.35(a)中虚线表示 $e(t)\neq 0$ 时是关联的，并设其传递系数为 1，由图 3.35(a)所示的结构有 $y(t)=kP(1+w_1'(t))r(t)/(1+kP)$，可知前馈控制将设定信号通过控制执行元件直接作用于被控对象，不影响系统的稳定性，不会引起超调和振荡。图 3.35(b)所示的结构是反馈比例加微分控制，有

$$y(t)=kP(w_2'(t)+\Delta w_3'(t))r(t)/[1+kP(w_2'(t)+\Delta w_3'(t))] \tag{3.105}$$

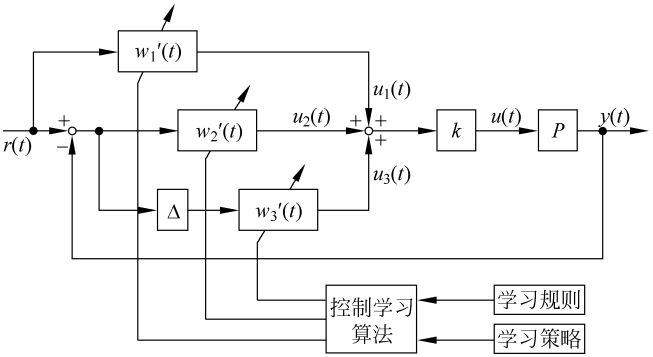
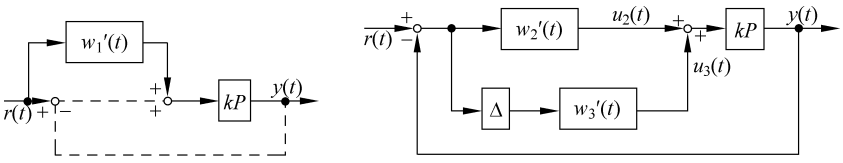


图 3.34 神经元控制系统的一种结构图



(a) 前馈控制　　(b) 反馈控制

图 3.35 神经元系统控制作用的分离

这将改变系统的闭环特性。反馈比例控制能迅速减小跟踪偏差；反馈微分控制能改善系统响应进入渐近跟踪给定信号时的动态特性，控制偏差向任何方向变化，神经元控制系统将这3种控制作用集合在一起，可使前馈控制抗扰动和适应参数变化能力增强，并可克服反馈比例控制引起的超调与振荡以及反馈微分控制对高频动态敏感等缺点。自适应神经元使用由某种学习规则和某种学习策略构成的学习算法，通过关联搜索进行递进式学习和自组织处理外部信息来确定其加权重 $w_i(t)$，即相应于这3种控制作用的比例系数 $w_i(t)(i=1,2,3)$，产生控制信号 $u(t)$，并沿着 $\varepsilon\{[r(t)-y(t)]^2\}/2$ 相应于 $w_i(t)$ 的负梯度方向使误差减小来修正 $w_i(t)$，直到偏差达到最小值0，$\varepsilon\{\cdot\}$ 是数学期望，它无须通过积分作用来消除静差。

实际上，神经元本身建立了被控对象和过程响应的某种形式的内部模型，假定 $P=B/A$，式中 A、B 互质，均为 z^{-1} 的多项式，$P(1)\neq 0$。神经元控制系统进入稳态跟踪后，有 $w_1'(t)=1/(kP(1))$，$P(1)$ 反映了被控对象的开环放大系数。$w_1'(t)$，$w_2'(t)$，$w_3'(t)$ 反映了被控对象和过程响应的动态特性。在动态响应过程中，通过神经元的自学习决定 $w_1(t)$，$w_2(t)$，$w_3(t)$，使系统获得某种性能的动态响应特性。当系统有偏差或处于动态响应时，神经元控制器使3种控制作用关联并发挥各自特长，迅速消除偏差。进入稳态跟踪后，$u_2(t)$ 和 $u_3(t)$ 均为0，系统处于开环控制，此时，$u(t)=ku_1(t)=kw_1'(t)r(t)$，$y(t)=P(1)u(t)=r(t)$。这种将开环和闭环、前馈和反馈集于一体的控制结构的优点是不需要通过积分作用来消除静差。

神经元的可调比例系数 k 在稳态时对控制系统无影响，其作用有：对开环放大系数较大的被控对象起到衰减神经元产生的控制作用，消除学习控制中的冲击与超调；对开环放大系数较小的被控对象起到增强神经元的控制作用，保证神经元能在全局范围内搜索 $\varepsilon\{[r(t)-y(t)]^2\}/2$ 的最小值。这使得复杂系统的控制器设计变成了单参数控制器设计，而且神经元系统对 k 的取值要求不严格。对于神经元的学习速率 d，取较大的值意味着学习速度加快，取较小的值意味着收敛速度加快。实际上，规范化后 d 取值的范围相当大，仿真实验表明 d 的取值对神经元系统品质影响不大。加权重 $w_i(t)(i=1,2,3)$ 的初值 $w_i(0)$ 只要求其中一个分量非零即可。

从自适应神经元 PID 控制系统的结构看，这是一种结构简单新颖的控制方法，其特点是不需要知道被控对象的结构和参数，也无须进行系统建模。

3.6　神经自适应控制

3.6.1　模型参考神经自适应控制

将神经网络同模型参考自适应控制相结合，就构成了模型参考神经自适应控制，其系统的结构形式和传统的模型参考自适应控制系统是相同的，只是通过神经网络给出被控对象的辨识模型。根据结构的不同可分为直接模型参考神经自适应控制和间接模型参考神经自适应控制两种类型，分别如图 3.36(a)和图 3.36(b)所示。间接方式比直接方式中多采用一个神经网络辨识器，其余部分完全相同。

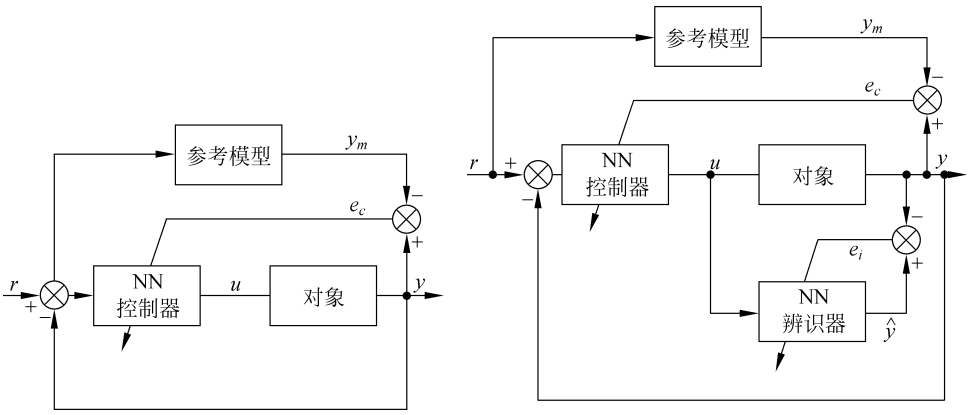


图 3.36　模型参考神经自适应控制的结构

NN 控制器的权重修正目标是使输出误差 $e_c=y_m-y\to 0$ 或二次型指标最小。对于直接方式，由于未知的非线性对象处于 e_c 和 NN 控制器的中间位置，所以给参数修正造成了

困难。为了避免这一问题,增加了 NN 辨识器,变为间接方式,此种方式其权重修正目标是使 $e_i=\hat{y}-y\to 0$。

3.6.2 神经自校正控制

自校正控制和模型参考自适应控制是自适应控制中的两种重要形式,它们有相似的特点但也有不同之处。二者的主要区别在于自校正控制没有参考模型,而依靠在线辨识来估计系统未知的参数,以此来在线校正控制参数并进行实时反馈控制。

由于神经网络的非线性映射能力强,使得它可以在自校正控制系统中充当未知系统函数逼近器。这样构成的神经网络自校正控制系统如图 3.37 所示,图中采用 BP 网络结构。

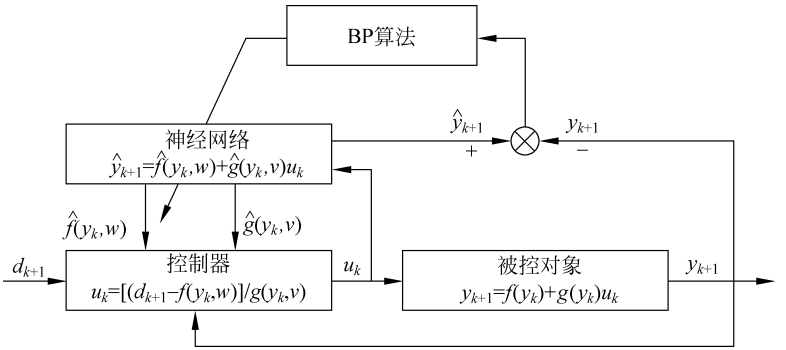


图 3.37 神经自校正控制系统

设单输入单输出线性系统为

$$y_{k+1}=f(y_k,y_{k-1},\cdots,y_{k-p},u_k,u_{k-1},\cdots,u_{k-p})+ g(y_k,y_{k-1},\cdots,y_{k-p},u_k,u_{k-1},\cdots,u_{k-p})u_k \tag{3.106}$$

其中,y_{k+1} 为对象输出,u_k 为控制器输出。在上述函数 $f(\cdot)$和 $g(\cdot)$已知情况下,可以采用下述控制

$$u_k=-\frac{f(\cdot)}{g(\cdot)}+\frac{d_{k+1}}{g(\cdot)} \tag{3.107}$$

使系统输出 y_{k+1} 精确地跟踪期望输出 d_{k+1}。

当 $f(\cdot)$和 $g(\cdot)$未知时,BP 神经网络通过学习算法可以逼近这些函数并重新自校正控制规律。为简单起见,设被控对象为一阶系统

$$y_{k+1}=f(y_k)+g(y_k)u_k \tag{3.108}$$

通过神经网络利用模型

$$\hat{y}_{k+1}=\hat{f}[y_k,\boldsymbol{W}(k)]+\hat{g}[y_k,\boldsymbol{V}(k)]u_k \tag{3.109}$$

去逼近对象模型,其中 $\boldsymbol{W}=[W_0,W_1,\cdots,W_{2p}]$,$\boldsymbol{V}=[V_0,V_1,\cdots,V_{2q}]$,且有

$$\hat{f}(0,\boldsymbol{W})=W_0,\quad \hat{g}(0,\boldsymbol{V})=V_0$$

相应的控制规律为

$$u_k=-\frac{\hat{f}[y_k,\boldsymbol{W}(k)]}{\hat{g}[y_k,\boldsymbol{V}(k)]}+\frac{d_{k+1}}{g[y_k,\boldsymbol{V}(k)]} \tag{3.110}$$

将式(3.110)代入式(3.108)可得

$$y_{k+1}=f(y_k)+g(y_k)\left\{-\frac{\hat{f}[y_k,\boldsymbol{W}(k)]}{\hat{g}[y_k,\boldsymbol{V}(k)]}+\frac{d_{k+1}}{\hat{g}[y_k,\boldsymbol{V}(k)]}\right\} \tag{3.111}$$

使得定义的输出误差

$$E_k=\frac{1}{2}(d_{k+1}-y_{k+1})^2=\frac{1}{2}e_{k+1}^2 \tag{3.112}$$

为最小,于是有

$$\begin{cases}\dfrac{\partial E_k}{\partial W^i(k)}=\dfrac{g(y_k)}{\hat{g}[y_k,\boldsymbol{V}(k)]}\cdot\left\{\dfrac{\partial\hat{f}[y_k,\boldsymbol{W}(k)]}{\partial W^i(k)}\right\}e_{k+1}\\ \dfrac{\partial E_k}{\partial V^j(k)}=\dfrac{g(y_k)}{\hat{g}[y_k,\boldsymbol{V}(k)]}\cdot\left\{\dfrac{\partial\hat{g}[y_k,\boldsymbol{W}(k)]}{\partial V^j(k)}\right\}u_k e_{k+1}\end{cases} \tag{3.113}$$

对于$\dfrac{\partial\hat{f}[y_k,\boldsymbol{W}(k)]}{\partial W^i(k)}$,$\dfrac{\partial\hat{g}[y_k,\boldsymbol{W}(k)]}{\partial V^j(k)}$可仿照推导 BP 算法过程计算,虽然 $g(y_k)$未知,但其符号已知,可用 $\mathrm{sgn}[g(y_k)]$代替 $g(y_k)$,这样就可以得到调整 $\boldsymbol{W}(k)$和 $\boldsymbol{V}(k)$的学习规则

$$\begin{cases}W^i(k+1)=W^i(k)-\eta_k\dfrac{\mathrm{sgn}[g(y_k)]}{\hat{g}[y_k,\boldsymbol{V}(k)]}\cdot\left\{\dfrac{\partial\hat{f}[y_k,\boldsymbol{W}(k)]}{\partial W^i(k)}\right\}e_{k+1}\\ V^j(k+1)=V^j(k)-\mu_k\dfrac{\mathrm{sgn}[g(y_k)]}{\hat{g}[y_k,\boldsymbol{V}(k)]}\cdot\left\{\dfrac{\partial\hat{g}[y_k,\boldsymbol{W}(k)]}{\partial V^j(k)}\right\}u_k e_{k+1}\end{cases} \tag{3.114}$$

其中,η_k,μ_k 均为学习速率。

上述修正权重学习算法收敛时,所获得的控制规律即为最佳的控制规律。

3.7 基于 MATLAB 的神经控制系统设计

3.7.1 MATLAB 神经网络工具箱

MATLAB 神经网络工具箱提供了进行神经网络设计和分析的许多工具函数,只要根据自己的需要调用相关函数,就可以完成网络设计、权重初始化、网络训练等。神经网络工具箱提供的函数有通用函数和专用函数两种形式。随着 MATLAB 神经网络工具箱的版本

不断更新,它的内容会更丰富,功能会更完善。

1. 神经网络工具箱的通用函数

下面介绍神经网络工具箱的一些通用函数的名称及功能。

innit( )初始化一个神经网络,如 net=innit(NET)表示网络 NET 初始化后为 net。

initlay( )层-层结构神经网络的初始化函数,如 net=initlay(NET)。

initwb( )神经网络某层的权重和偏置初始化函数,如 net=initzero(NET,i),其中 i 为第 i 层,net 神经网络第 i 层的权重和偏置修正后的网络。

initzero( )置权重为零的初始化函数。

train( )神经网络训练函数,它只调用设定的或默认的训练函数对网络进行训练。

adapt( )神经网络自适应训练函数,它在每一个输入时间阶段更新,而在进行下一个输入仿真前完成。

sim( )神经网络仿真函数。网络的权重和偏置经训练确定后,利用 sim( )可以仿真神经网络的性能。

dotprod( )权重点积函数。调用它使网络输入向量与权重的点积可以得到加权输入。

normprod( )规范权重点积函数。

netsum( )输入求和函数,通过某层加权输入与偏置相加作为该层输入。

netprod( )网络输入的积函数。它使网络输入向量与权重的点积来得到加权输入。

concur( )结构一致函数。它使本来不一致的权重向量和偏置向量的结构一致。

2. 神经网络工具箱的专用函数

MATLAB 神经网络工具箱包括的网络模型有感知器、线性网络、BP 神经网络、径向基网络、反馈网络、自组织网络、Hopfield 网络、学习向量量化网络、控制系统网络等。对于每一种网络都有一些专用函数。以下仅简要介绍常用的 BP 神经网络 MATLAB 函数。

MATLAB 神经网络工具箱提供了大量的 BP 神经网络分析和设计的工具函数,在 MATLAB 工作空间的命令行输入“help backprop”,便可获得与 BP 神经网络相关的函数,进一步利用 help 命令可得到相关函数的详细介绍。

1) BP 神经网络参数设置函数 newff( )

函数功能:构建一个 BP 神经网络。

函数形式:

```
net = newff(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF)
```

P:输入数据矩阵。

T:输出数据矩阵。

S:隐层结点数。

TF：结点传递函数，包括硬限幅传递函数 hardlim、对称硬限幅传递函数 hardlims、线性传递函数 purelin、正切 S 型传递函数 tansig、对数 S 型传递函数 logsig。

BTF：训练函数，包括梯度下降 BP 算法训练函数 traingd、动量反传的梯度下降 BP 算法训练函数 traingdm、动态自适应学习率的梯度下降 BP 算法训练函数 traingda、动量反传和动态自适应学习率的梯度下降 BP 算法训练函数 traingdx、Levenberg_Marquardt 的 BP 算法训练函数 trainlm。

BLF：网络学习函数，包括 BP 学习规则 learngd、带动量项的 BP 学习规则 learngdm。

PF：性能分析函数，包括均值绝对误差性能分析函数 mae、均方差性能分析函数 mse。

IPF：输入处理函数。

OPF：输出处理函数。

DDF：验证数据划分函数。

一般在使用过程中设置前 6 个参数，后 4 个参数采用系统默认参数。

2）BP 神经网络训练函数 train( )

函数功能：用训练数据训练 BP 神经网络。

函数形式：

```
[net,tr] = train(NET,X,T,Pi,Ai)
```

NET：待训练网络。

X：输入数据矩阵。

T：输出数据矩阵。

Pi：初始化输入层条件。

Ai：初始化输出层条件。

net：训练好的网络。

tr：训练过程记录。

一般在使用过程中设置前 3 个参数，后两个参数采用系统默认参数。

3）BP 神经网络预测函数 sim( )

函数功能：用训练好的 BP 神经网络预测函数输出。

函数形式：

```
y = sim(net,x)
```

net：训练好的网络。

x：输入数据。

y：网络预测数据。

只要我们能够熟练掌握上述 3 个函数，就可以完整地编写一个 BP 神经网络预测或分类的程序。

3. 神经网络工具箱的图形用户界面

神经网络工具箱提供的有关函数是直接在 MATLAB 命令窗口执行并显示结果,为了方便,MATLAB 6.5 版本提供的神经网络工具箱增加了图形用户界面,具有简洁、友好的人机交互功能。从 MATLAB 7.4.0 开始,又增加了神经网络拟合工具的图形界面(Graphical User Interface,GUI),便于引导用户建立和训练神经网络。

1) 神经网络编辑器

神经网络编辑器(Network/Data Manager)用于建立网络、训练网络、仿真网络,将数据导出到 MATLAB 工作空间、清除数据、从 MATLAB 工作空间中导入数据、变量存盘和读取。

在 MATLAB 命令窗口中直接键入 nntool 或在 MATLAB 窗口左下角的 Start 菜单中,单击 Toolbox→ Neural Network 命令子菜单中的 NNTool 选项,即可启动神经网络编辑器窗口。也可以通过单击 Help 按钮,按其介绍步骤进行操作。

2) 神经网络拟合工具

神经网络拟合工具(Neural Network Fitting Tool)用于函数逼近和数据拟合。拟合函数的是一个两层前馈网络,输入和输出数据确定后,可以调整的只有隐层的神经元数目。训练算法采用 trainlm 算法。

在 MATLAB 命令窗口中直接输入 nftool 命令或在 MATLAB 窗口左下角的 Start 菜单中,单击 Toolbox→ Neural Network 命令子菜单中的 NFTool 选项,即可启动神经网络拟合工具。

4. 基于 Simulink 的神经网络模块

Simulink 是一个进行多种动态系统建模、仿真和综合分析的集成软件包,神经网络工具箱提供了一套可在 Simulink 中构建神经网络的模块,在 MATLAB 工作空间中也可以使用函数 gensim( )生成一个相应的 Simulink 网络模块。

1) 模块的设置

在 Simulink 库浏览窗口的 Neural Network Blockset 结点上,单击鼠标右键后,可打开 Neural Network Blockset 模块集窗口。该模块集有以下 4 个模块库:

- 传输函数模块库(Transfer Functions)。
- 网络输入模块库(Net Input Functions)。
- 权重模块库(Weight Functions)。
- 控制系统模块库(Control Systems)。

在打开的 Neural Network Blockset 模块集的窗口中,双击各个模块库的图标,便可以打开所需要的模块库。

2) 模块的生成

使用函数 gensim ( )可以在 MATLAB 工作空间中对一个网络生成 Simulink 模块化描述,进而能够在 Simulink 中对网络进行仿真。生成神经网络 Simulink 模块的指令调用方式如下:

利用函数 gensim()生成一个网络 Simulink 模块，使用调用格式 gensim(net,st)，其中第一个参数设定了要生成的模块化描述的网络；第二个参数指定了采样时间，通常为一正实数。如果网络没有与输入权重或者层中权重相关的延迟，则第二个参数设定为－1。那么函数将生成一个连续采样的网络。

5. 自定义神经网络及自定义函数

MATLAB 神经网络工具箱提供标准的创建函数创建神经网络对应一定的网络类型，对于某些特殊要求，或设计、开发新型的神经网络，或参与不同的学习训练算法，可以应用 MATLAB 神经网络工具箱自定义神经网络的方式。

自定义网络需要完成以下工作：

1）定制网络

根据定制网络要求，设计一个比较复杂的网络结构。

2）网络设计

自定义网络通过调用 network( )函数加以实现，网络对象包括很多属性，如结构属性、子对象属性、网络函数、权重和阈值属性等。通过设置相应的属性，可以获得需要的网络结构、算法及功能。

3）网络训练

网络设计完成后，首先对自定义网络初始化，然后需要给出输入和输出样本，对网络进行仿真，再设置训练参数，对网络进行训练。

MATLAB 神经网络工具箱允许创建多种类函数，可以根据需要，在初始化、仿真以及训练中应用多种方法对网络进行自调整。这些自行编制的函数称为自定义函数，主要有 4 类：仿真函数、初始化函数、学习函数和自组织映射函数。

3.7.2　基于 MATLAB 的模型参考神经自适应控制系统仿真

神经网络模型参考神经自适应控制系统如图 3.38 所示，它包括参考模型、神经网络对象模型、神经网络控制器和被控对象。首先辨识出对象模型，然后训练控制器，使系统输出能够跟随参考模型输出。

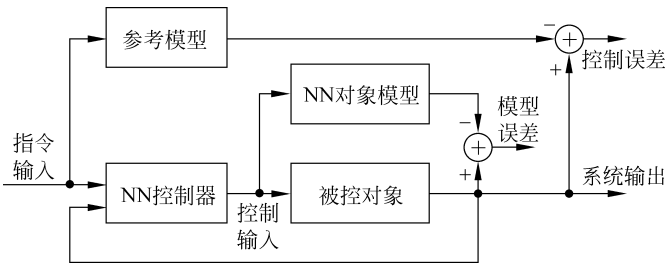


图 3.38　神经网络模型参考自适应控制系统

图3.38中神经网络控制器和神经网络对象模型,可以从神经网络工具箱中获得,图3.39给出了它们的详细情况。

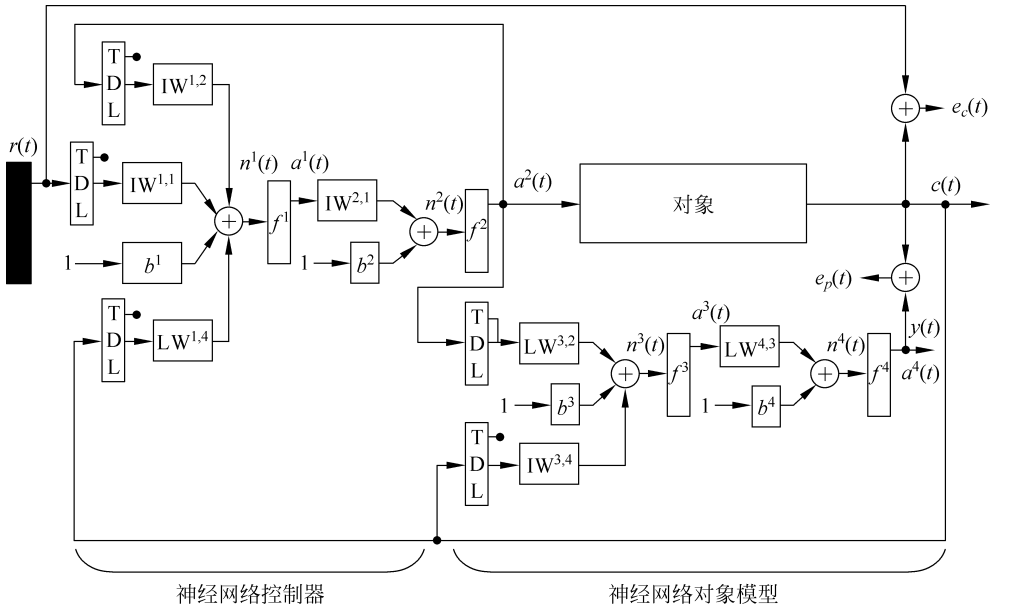


图3.39 神经网络控制器和神经网络对象模型

神经网络控制器和神经网络对象模型中使用的神经网络均选用三层前馈神经网络,隐层神经元数目自行定义。

神经网络控制器包括3个输入信号:延迟参考输入、延迟控制器输出和延迟系统输出。神经网络对象模型的输入包括两种信号:延迟控制器输出和延迟系统输出。

输入信号的延迟大小与系统阶次有关系,通常系统阶次越高,表示系统的储能元件多,延迟就会越大。

1. 单自由度机械臂控制问题的描述

一个单自由度机械臂如图3.40所示,其运动方程为

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}t^2}=-10\sin\phi-2\frac{\mathrm{d}\phi}{\mathrm{d}t}+u \tag{3.115}$$

其中,ϕ 为机械臂与参考线之间的夹角;u 为直流电机的输入转矩。

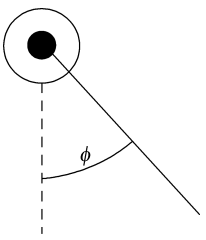


图3.40 单自由度机械臂

控制的目的是训练神经网络控制器,使得单自由度机械臂能跟踪给定的参考模型

$$\frac{\mathrm{d}^2y_r}{\mathrm{d}t^2}=-9y_r-6\frac{\mathrm{d}y_r}{\mathrm{d}t}+9r \tag{3.116}$$

其中,y_r 为参考模型的输出;r 为参考输入信号。

2. 机械臂模型和模型参考控制器模型的建立

应用 MATLAB 神经网络工具箱演示神经网络控制器采用的是前馈网络 5-13-1 结构：5 个输入端，隐层有 13 个神经元，输出为 1 个神经元。控制器的输入包括两个延迟参考输入、两个延迟系统输出和一个延迟控制器输出。采样间隔为 0.5s。

在 MATLAB 命令行窗口中输入 mrefrobotarm，就会自动调用 Simulink，并且产生一个模型窗口，如图 3.41 所示。

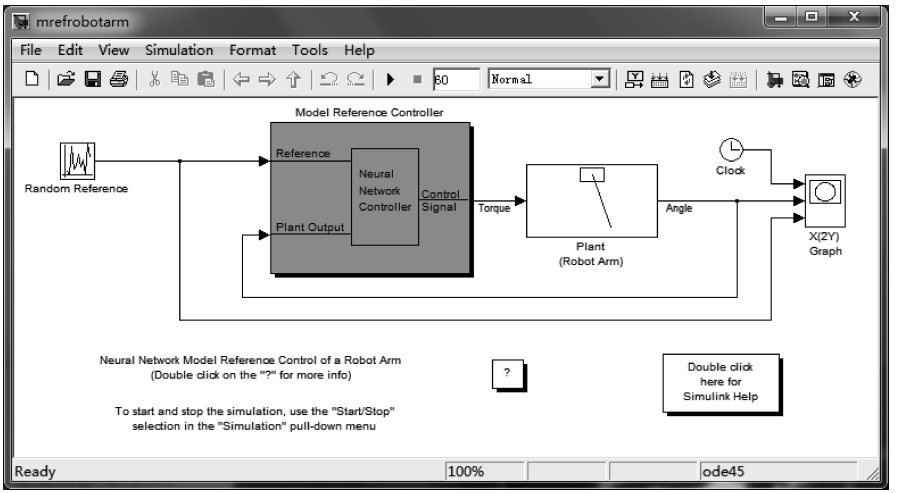


图 3.41　mrefrobotarm 模型窗口

在图 3.41 中的机械臂模块[Plant(Robot Arm)]是用机械臂运动方程编制的 Simulink 模块。双击该模块，可以看到如图 3.42 所示的详细结构。模型参考控制器模块也在如图 3.41 所示的 mrefrobotarm 模型窗口中，该模块是在神经网络工具箱中生成并复制过来的。用户可以通过 Look under mask 模型的右键快捷菜单命令查看该模块未封装的具体实现。

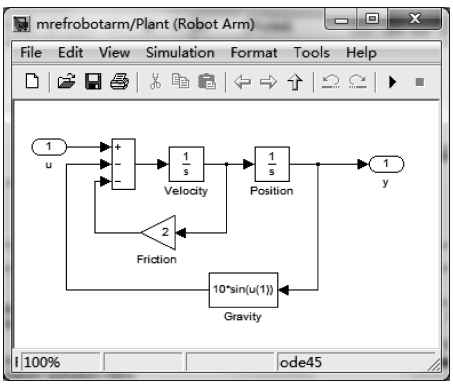


图 3.42　机械臂模块的模型窗口

3. 神经网络系统模型辨识

双击 Model Reference Control 模块,将会弹出一个如图 3.43 所示新的窗口,该窗口用于训练 NARMA-L2 模型。在如图 3.43 所示的窗口中单击 Plant Identification 按钮,将会弹出一个系统辨识窗口。

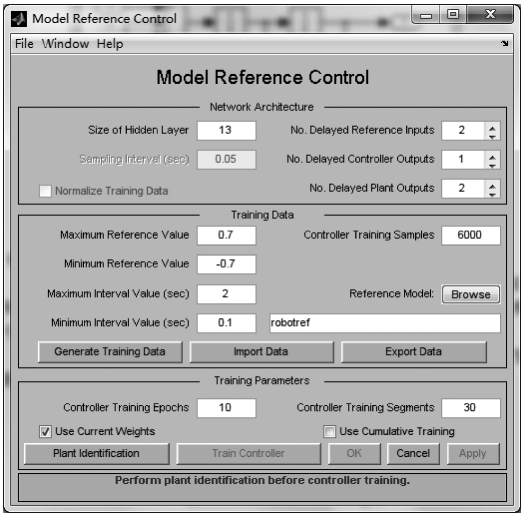


图 3.43 模型控制参数设置窗口

单击 Generate Training Data 按钮,程序开始产生控制器所需要的数据。在数据产生结束后,将会出现如图 3.44 所示的数据窗口。

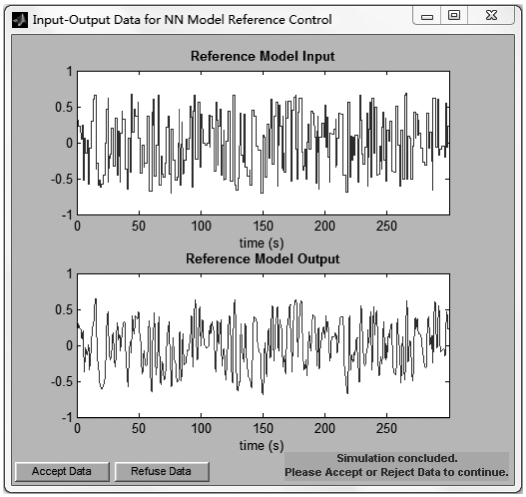


图 3.44 参考输入、输出曲线

单击 Accept Data 按钮，返回 Model Reference Control 窗口。单击 Train Controller 按钮，开始训练。程序将一段数据输入网络并进行指定次数的迭代，直到所有的训练数据都输入了网络，这个过程才结束。

因为控制器必须使用动态反向传播算法进行训练，因此控制器训练时间要比系统模型训练时间长得多。控制器训练结束后，会显示出如图 3.45 所示的系统闭环响应曲线。

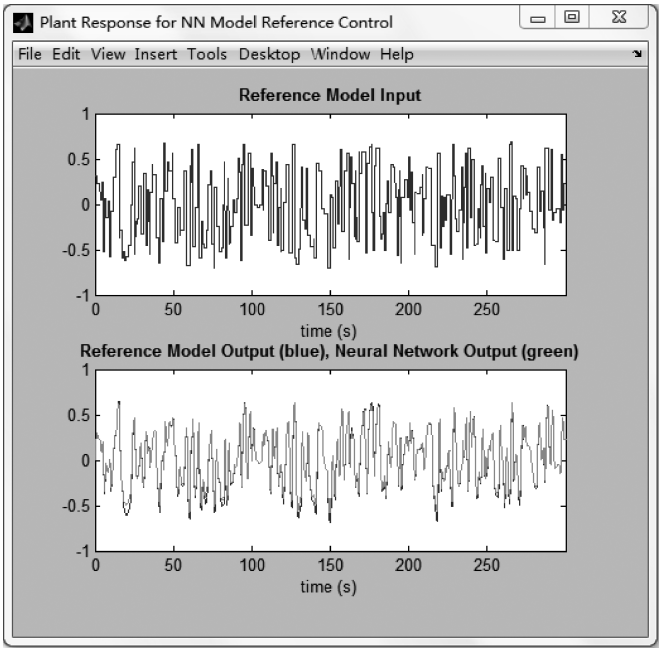


图 3.45　系统闭环响应曲线

图 3.45 中上半部分曲线是用于训练的随机参考信号，下半部分两条曲线分别是参考模型的响应信号和闭环系统的响应信号。如果系统的响应信号跟踪参考模型的信号不准确，表明控制器的性能还不够好，为提高其控制性能，需要返回到 Model Reference Control 窗口，可以再次单击 Train Controller 按钮，这样会继续使用同样的数据进行训练。

如果需要使用新的数据对控制器继续进行训练，可单击 Train Controller 按钮之前单击 Generate Training Data 按钮，或者在确认 Use Current Weights 被选中的情况下单击 Input Data 按钮。应该指出，如果系统模型不够准确，也会影响控制器的训练。

4. 机械臂控制系统的 MATLAB 仿真

在前面已完成机械臂模型和模型参考控制器 Simulink 模型的建立，以及神经网络系统模型辨识的基础上，可以进行机械臂模型参考神经自适应控制系统的 MATLAB 仿真。

在系统辨识窗口中单击 OK 按钮，将训练好的神经网络控制器权重导入 Simulink 模型

中。返回到 mrefrobotarm 模型窗口,从 Simulink 菜单中选择 Start 命令开始仿真。

当系统仿真结束时,将会显示控制系统的输出信号和参考信号,如图 3.46 所示。其中,方波形曲线为参考模型信号,而另一曲线为控制系统输出响应信号。

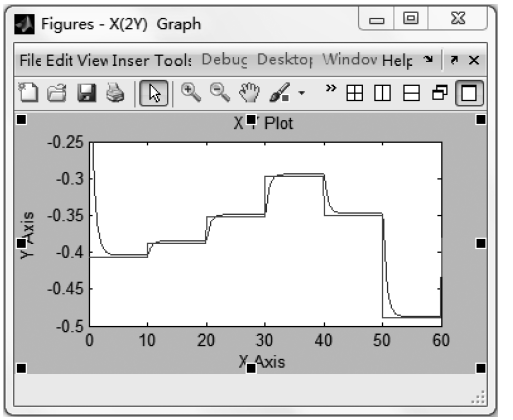


图 3.46 控制系统 MATLAB 仿真结果

启迪思考题

3.1 人工神经元有哪 3 个主要组成部分?从信息处理的角度它们各起什么作用?

3.2 神经细胞的细胞膜和突触各有什么作用?

3.3 画出一个神经元形式化结构模型,指出各部分的名称和作用,说明“形式化”和“结构”各是什么意思。

3.4 人工神经元模型的输出函数有哪几种形式?这些函数都具有突变性和饱和性,这两个特点是模拟神经细胞的什么特性?

3.5 指出人工神经元模型在信息处理过程中都有哪些部分具有非线性特性。

3.6 人工神经网络具有哪些特点?这些特点使它可以应用到哪些方面?

3.7 什么是神经网络的学习?什么是对神经网络的训练?学习和训练有何区别和联系?

3.8 神经网络的学习规则有哪几种形式?

3.9 如何理解神经元、神经网络和学习算法是神经网络系统的三要素?

3.10 BP 神经网络的误差反向传播学习算法为什么要求误差反向传播?正向传播可以实现学习吗?

3.11 在 BP 网络学习算法中加动量因子有何作用?

3.12 RBF 神经网络与 BP 神经网络在结构和功能上有何区别?

3.13 改进神经网络的设计应该从哪几方面考虑？

3.14 神经网络控制的原理是什么？其本质是应用了神经网络的什么特性？

3.15 为什么神经网络可以应用于智能控制、系统辨识、参数优化、模式识别、故障诊断等领域？

3.16 神经网络控制与模糊控制在控制原理上有何异同？

3.17 说明神经网络和模糊系统融合的具体形式。

3.18 神经网络模糊控制与模糊神经网络控制二者有何区别？

3.19 神经元 PID 控制和神经网络优化 PID 控制有何异同？

3.20 试比较前馈网络、反馈网络和深度信念网络之间在结构上和训练上有何不同。

3.21 卷积神经网络在结构上有什么特点？它在用于图像识别中局部连接和权重共享时是基于什么原理？

3.22 说明循环神经网络和递归神经网络的区别和联系。