第四章 神经网络控制

4.1 神经网络的基本原理

人工神经网络系统是指利用工程技术手段模拟人脑神经网络的结构和功能的一种技术系统，它是一种大规模并行处理的非线性动力学系统。人工神经网络（简称神经网络，Neural Netwoeks）是有众多的人工神经元连接而形成的一个网络。

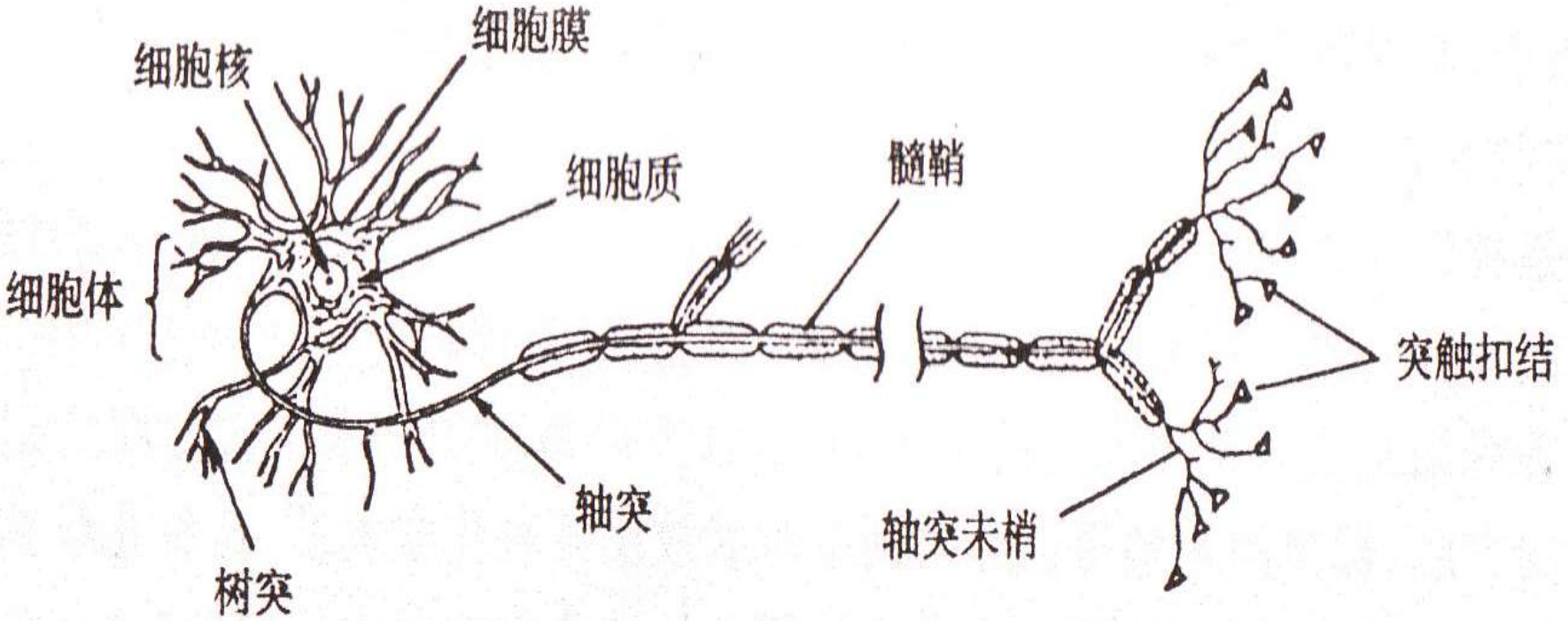
神经网络的主要特点：

- 分布式存储信息的特点；
- 对信息的处理及推理过程具有并行的特性；
- 对信息的处理具有自组织、学习的特性。

4.1.1 生物神经细胞(神经元)的结构与功能

在重约1500g的人脑中，包括着约千亿个神经细胞和胶质细胞，其中80%是胶质细胞，它们主要负责供应脑的营养，其余的神经细胞亦称为神经元，它们负责接收或产生信息，传递和处理信息。

生物神经元的基本结构：生物神经元由细胞体、树突和轴突组成，其结构如图4.1



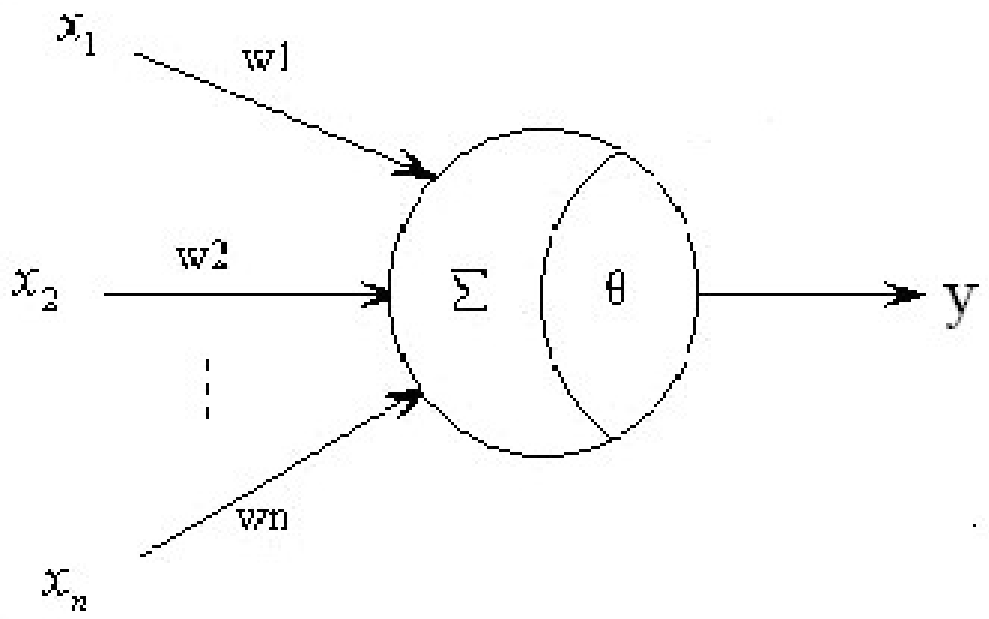
- 细胞体：由细胞核、细胞质和细胞膜组成。
- 树突：由细胞体向外伸出的许多较短的树枝状分支组成，长约1mm。树突相当于神经元的输入端，它用于接收周围其它神经元传入的神经冲动。

- 轴突：有细胞体向外伸出的最长的一条神经纤维，其长度一般是数厘米到1米左右。远离细胞体一侧的轴突端部有许多分支，称为轴突末梢，或称为神经末梢，其上有许多扣结称为突出扣结。轴突通过轴突末梢向其它神经元传出神经冲动。
- 突触：一个神经元的轴突末梢和另一个神经元的树突，通过微小间隙相联结，这样的联结处称为突触。

  一个神经元约有103～104个突触，人脑大约有1014个突触。神经细胞通过突触复杂地联结着，形成复杂的大脑神经系统。

4.1.2 人工神经元的数学模型

人工神经元是一个多输入单输出的信息处理单元，如图4.2所示。



考虑从n个神经元接收输入信号的神经元，设输入信号分别为$x_1,x_2,\ldots,x_n$，兴奋阈值为θ，输出信号为y。$w_i(i=1,2,\ldots,n)$为其它神经元到当前神经元的连接权值，f(x)为当前神经元的非线性激励函数。当前神经元所接收的输入信号的总和，可以描述为

$$x=\sum_{i=1}^{n} w_i x_i-\theta \tag{4.1}$$

当前神经元的输出为

$$y=f(\sum_{i=1}^{n} w_i x_i-\theta) \tag{4.2}$$

若将阈值也作为一个权值，并记$w_0=-\theta, x_0=1$，则(4.2)可以写

$$y=f(\sum_{i=0}^{n} w_i x_i) \tag{4.3}$$

- 阈值函数

$$f(x)=\begin{cases}1 & x>0\\ 0 & x\le 0\end{cases}$$

- 双向阈值函数

$$f(x)=\begin{cases}1 & x>0\\ -1 & x\le 0\end{cases}$$

- S函数

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

- 双曲正切函数

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

- 高斯函数

$$f(x) = e^{-(x^2/\sigma^2)}$$

4.1.3 神经网络的结构

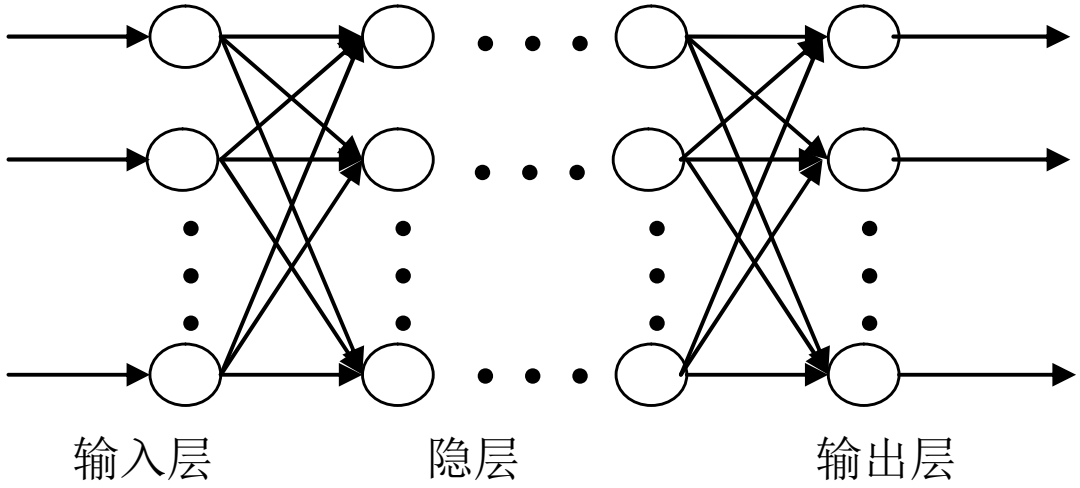
人脑中的神经细胞不是孤立的，而是通过突触形式相互连接，构成结构与功能十分复杂的神经网络系统。人工神经网络可以看成是以神经元为节点，节点间可以由有向线连接的一种图。通常所说的神经网络的结构，主要是指它的连接方式。神经网络的结构大体上可以分为层状和网状两大类。

- 层状结构：有若干层组成，每层中有一定数量的神经元，相邻层的神经元间单向连接，同层的神经元不能连接。
- 网状结构：任何两个神经元间都有可能双向连接。

下面介绍几种常见的神经网络结构：

- 前向网络（前馈网络）

  如图4.3所示，神经元分层排列，组成输入层，隐含层（可以有若干层）和输出层。这中网络的特点是只有前后相邻两层之间的神经元相互连接，层间神经元之间不连接，各个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，并只有一个输出送给下一层的各神经元。在前向网络中有计算功能的节点称为计算单元，而输入节点无计算功能。



- 反馈网络

  反馈网络从输出层到输入层有反馈，即每一个节点同时接收外来输入和来自其它节点的反馈输入，其中包括神经元输出信号引回到本身输入构成的自环反馈。这种反馈网络的每一个节点均是计算单元。

- 相互结合型网络

属于网状结构。构成网络中的每一个神经元都有可能相互双向连接，所有的神经元既作为输入，同时也作为输出。

4.1.4 神经网络的学习方法和规则

在神经网络中，修改权值的规则过程称为学习过程，而如何调整连接权值就构成了不同的学习方法。主要学习方法有以下三种：

- 监督学习（有教师学习）

为了使神经网络在实际应用中解决各种问题，必须对它进行训练，就是从应用环境中选取一些样本数据（系统的输入和输出数据对，输出数据亦称为标准答案），通过比较真值输出和实际输出调整权值，直到得到满意的输入输出关系为此。这组已知的输入输出数据对，称为教师样本。

- 非监督学习(无教师学习)

  无教师学习的训练中，只有输入而没有目标输出，训练过程神经网络自动地将各输入数据的特征提取出来，并将其分成若干类。训练好的为了能够训练识别数据集以外的新的输入类别。

- 再励学习(强化学习)

  这种学习方法介于以上两种情况之间，外部环境对系统的输出结果只给出评价（奖或罚）而不给出正确答案，通过奖罚来调整权值以达到学习目的。

4.2 感知器和前向神经网络

4.2.1 感知器

- 感知器模型是由美国学者F. Rosenblatt 于1967年建立的，主要用于模拟人的视觉，用于模式分类。
- 单层感知器可以看成是多输入单输出的单个神经元，其激励函数是阈值函数或双阈值函数。单神经元感知器如图4.2.1所示

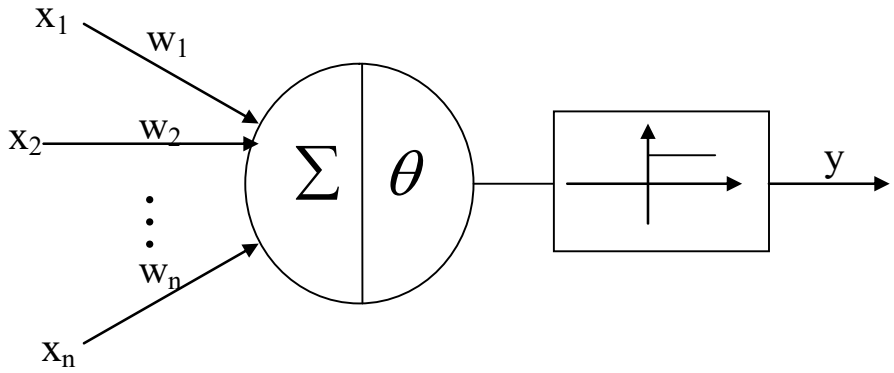


图4.2.1：单神经元感知器

单神经元感知器的输出是

$$y = f(\sum_{i=1}^{n} \omega_i x_i - \theta) \qquad (4.2.1')$$

其中$f(x)$取阈值函数

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

或双阈值函数

$$f(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

将输入和权值写成向量的形式，即记

$$\mathbf{x} = (x_1 \quad x_2 \quad \cdots \quad x_n)^T \quad \text{和} \quad \boldsymbol{\omega} = (\omega_1 \quad \omega_2 \quad \cdots \quad \omega_n)^T$$

则（4.2.1）式可以等价地写为

$$y = f(\boldsymbol{\omega}^T \mathbf{x} - \theta)$$

记 $d(x) = \boldsymbol{\omega}^T \mathbf{x} - \theta$ ，称d(x)为决策函数，而 d(x)=0所形成的分界线可以将输入分成两类。

单神经元感知器是典型的有教师学习训练，设训练样本包含有p个输入输出对 $\{\mathbf{x}^l, \bar{\mathbf{y}}^l\}$

$(l = 1,2,\cdots, p)$, 相应的学习（或称训练）算法如下：

①取算法部数k=0，各个权值置为小的随机数

$$\boldsymbol{\omega}(0) = \begin{pmatrix} \omega_1(0) & \omega_2(0) & \cdots & \omega_n(0) \end{pmatrix}^T$$

置阈值初值 $\theta(0)$

②任取样本集中的一对样本$\{\mathbf{x}^l, \bar{\mathbf{y}}^l\}$，计算

$$y^l = f(\boldsymbol{\omega}^T(0)\mathbf{x}^l - \theta(0))$$

③对连接权值和阈值进行修正：

$$\boldsymbol{\omega}(k+1) = \boldsymbol{\omega}(k) + \alpha(\bar{y}^l - y^l)\mathbf{x}^l$$

$$\theta(k+1) = \theta(k) - \alpha(\bar{y}^l - y^l)$$

④置k=k+1，重复以上2、3步骤，直到对所有的样本都成立：

$$\boldsymbol{\omega}(k+1) = \boldsymbol{\omega}(k) \text{ 和 } \theta(k+1) = \theta(k+1)$$

结论：感知器的训练算法收敛的充分必要条件是样本是线性可分的。

例**4.1**：有两类共四个样本：

$$\{x^1=(2\quad 2)^T,\ \bar{y}^1=0\}\qquad \{x^2=(1\quad -3)^T,\ \bar{y}^2=1\}$$

$$\{x^3=(-2\quad 1)^T,\ \bar{y}^3=0\}\qquad \{x^4=(-3\quad -1)^T,\ \bar{y}^4=1\}$$

采用阈值函数，找出能对这些样本正确分类的决策函数。

解：置初时权值和阈值为0，即 $\omega(0)=(0\quad 0)^T$ 和 $\theta(0)=0$，取学习率 $\alpha=1$。

1)第一次循环开始：输入第一个样本

$$y^1=f(\boldsymbol{\omega}^T(0)\mathbf{x}^1-\theta(0))=f((0\ 0)\begin{pmatrix}2\\2\end{pmatrix}-0)=f(0)=1$$

与目标值 $\bar{y}^1=1$不同，需要进行权值和阈值修正：

$$\boldsymbol{\omega}(1)=\boldsymbol{\omega}(0)+(\bar{y}^1-y^1)\mathbf{x}^1=\begin{pmatrix}0\\0\end{pmatrix}+(0-1)\begin{pmatrix}2\\2\end{pmatrix}=\begin{pmatrix}-2\\-2\end{pmatrix}$$

$$\theta(1)=\theta(0)-(\bar{y}^1-y^1)=0-(0-1)=1$$

所以，这时的决策函数是$d(x)=-2x_1-2x_2-1$,通过画图可知，它还不能对样本正确分类。输入第二个样本，

$$y^2=f(\boldsymbol{\omega}^T(1)\mathbf{x}^2-\theta(1))=f((-2\ -2)\begin{pmatrix}1\\-3\end{pmatrix}-1)=f(3)=1$$

与目标值相同，不需要进行权值和阈值的修正，即取 $\boldsymbol{\omega}(2)=\boldsymbol{\omega}(1)$和 $\theta(2)=\theta(1)$ 。输入第三个样本，

$$y^3=f(\boldsymbol{\omega}^T(2)\mathbf{x}^3-\theta(2))=f((-2\ \ -2)\begin{pmatrix}-2\\1\end{pmatrix}-1)=f(1)=1$$

与目标值不同，需要进行权值和阈值修正：

$$\boldsymbol{\omega}(3)=\boldsymbol{\omega}(2)+(\bar{y}^3-y^3)\mathbf{x}^3=\begin{pmatrix}-2\\-2\end{pmatrix}-\begin{pmatrix}-2\\1\end{pmatrix}=\begin{pmatrix}0\\-3\end{pmatrix}$$

$$\theta(3)=\theta(2)-(\bar{y}^3-y^3)=1-(0-1)=2$$

这时的决策函数为d(x)=-3x$_2$-2。输入第四个样本，

$$y^4=f(\boldsymbol{\omega}^T(3)\mathbf{x}^4-\theta(3))=f((0\ \ -3)\begin{pmatrix}-3\\-1\end{pmatrix}-2)=f(1)=1$$

与样本输出一样，不需要对权值和阈值进行修正，即取 $\boldsymbol{\omega}(4)=\boldsymbol{\omega}(3)$ 和 $\theta(4)=\theta(3)$，第一个循环结束。

在这次循环中，由于对两个样本不能正确分类而对权值和阈值进行了修正，所以需要进行第二次循环。

2)第二次循环开始：输入第一个样本

$$y^1 = f(\boldsymbol{\omega}^T(4)\mathbf{x}^1 - \theta(4)) = f(\begin{pmatrix}0 & -3\end{pmatrix}\begin{pmatrix}2\\2\end{pmatrix} - 2) = f(-8) = 0$$

与目标值相同，不需要进行权值和阈值修正，即取 $\boldsymbol{\omega}(5) = \boldsymbol{\omega}(4)$ 和 $\theta(5) = \theta(4)$。输入第二个样本

$$y^2 = f(\boldsymbol{\omega}^T(5)\mathbf{x}^2 - \theta(5)) = f(\begin{pmatrix}0 & -3\end{pmatrix}\begin{pmatrix}1\\-3\end{pmatrix} - 2) = f(7) = 1$$

与目标值相同，不需要对权值和阈值进行修正，即取 $\boldsymbol{\omega}(6) = \boldsymbol{\omega}(5)$ 和 $\theta(6) = \theta(5)$。输入第三个样本

$$y^3 = f(\boldsymbol{\omega}^T(6)\mathbf{x}^3 - \theta(6)) = f(\begin{pmatrix}0 & -3\end{pmatrix}\begin{pmatrix}-2\\1\end{pmatrix} - 2) = f(-5) = 0$$

与目标值相同，不需要对权值和阈值进行修正，即取

$\boldsymbol{\omega}(7)=\boldsymbol{\omega}(6)$ 和 $\theta(7)=\theta(6)$。输入第四个样本

$$y^4=f(\boldsymbol{\omega}^T(7)\mathbf{x}^4-\theta(7))=f(\begin{pmatrix}0 & -3\end{pmatrix}\begin{pmatrix}-3\\-1\end{pmatrix}-2)=f(1)=1$$

与目标值相同，不需要对权值和阈值进行修正，即取

$\boldsymbol{\omega}(8)=\boldsymbol{\omega}(7)$ 和 $\theta(8)=\theta(7)$。第二次循环结束。在该次循环中，对任何一个样本的计算均没有对权值和阈值进行修正，所以循环结束。最终的决策函数是

$$d(x)=\boldsymbol{\omega}^T(8)\mathbf{x}-\theta(8)=-3x_2-2$$

单层感知器只能对线性可分的样本输入点进行分类，因而分类能力不强，如对简单的异或问题

输入	(0,0)	(0,1)	(1,0)	(1,1)
输出	0	1	1	0

都不能进行分类。

作业：设有一平面上的两类模式，如下表所示。采用两个输入，单输出的单层感知器对其进行分类。

输入	(-1,-0.5)	(-0.5,0.5)	(0.3,-0.5)	(-0.1,1)
输出	1	1	0	0

4.2.2 前向神经网络

- 前向网络是目前研究最多的网络形式之一，它包含输入层，隐含层和输出层，其中隐含层可以为一层和多层。
- 本节只考虑只具有一层隐含层的前向网络，其中输入层有n个节点，隐含层有m个节点，输出层有p个节点，如图4.2.1所示

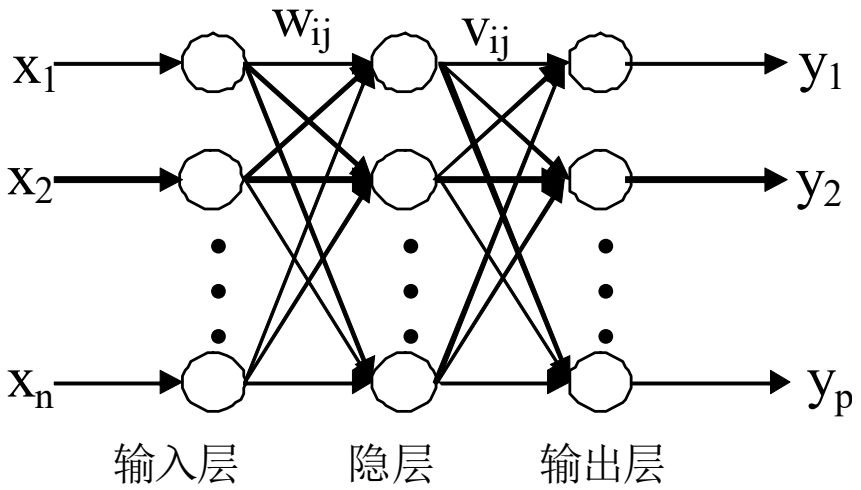


图4.2.1

前向网络的输入层不具有计算功能，其它节点均为神经元，激励函均是S函数，即

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

输入层到隐含层的连接权值为w_{ij}，它表示输入层的第j个节点到隐含层的第i个节点的权值；T_{li}表示第个i隐含层节点到第l个输出层的连接权值。前向网络是有教师学习网络，设给定的希望输出为$\bar{y}_l\ (l=1,2,\cdots,p)$

隐含层的输出：第i个隐含层节点的输出为

$$z_i=f(\sum_{j=1}^{n}w_{ij}x_j-\theta_i)=f(net_i^1)\ \ (i=1,2,\cdots,m) \qquad (4.2.1)$$

输出节点的计算：第个l输出节点的输出为

$$y_l = f(\sum_{i=1}^{m} v_{li} z_i - \xi_l) = f(net_l^2)\ (l = 1,2,\cdots p) \qquad (4.2.2)$$

其中 ξ_l 为输出神经元的阈值。输出节点的误差公式：

$$E = \frac{1}{2}\sum_{l=1}^{p}(\bar{y}_l - y_l)^2 \qquad (4.2.3)$$

将（4.2.1）代人（4.2.2）得

$$y_l = f(\sum_{i=1}^{m} v_{li} f(\sum_{j=1}^{n} w_{ij} x_i - \theta_i) - \xi_l)\ (l = 1,2,\cdots, p) \qquad (4.2.4)$$

将计算结果再代人（4.2.3）得到

$$E = \frac{1}{2}\sum_{l=1}^{p}\left[\bar{y}_l - f(\sum_{i=1}^{m} v_{li} f(\sum_{j=1}^{n} w_{ij} x_i - \theta_i) - \xi_l)\right]^2 \qquad (4.2.5)$$

由于E可以看成是$y_1, y_2, \ldots y_p$的复合函数，故有

问题转化为：求全权值v_{li}和w_{ij}及其阈值θ_i和ξ_l使得E最小，即求解如下的极值问题：

$$\min_{v_{li},w_{ij}} E = \min_{v_{li},w_{ij}} \left\{ \frac{1}{2} \sum_{l=1}^{p} \left[\overline{y}_l - f\left(\sum_{i=1}^{m} v_{li} f\left(\sum_{j=1}^{n} w_{ij} x_i - \theta_i \right) - \xi_l \right) \right]^2 \right\}$$

多元单值函数的最后化方法：迭代下降法

考虑问题：$\min_{x \in D} f(x), D \subset \mathbf{R}^n$，假设已经得到第k步的迭代最优值$x_k$，则下一步的迭代最有值由当前步(第k步)的值沿着矢量方向$S(x_k)$前进一步而得到：

$$x_{k+1} = x_k + \eta^{(k)} S(x_k)$$

其中$\eta^{(k)}$是正步长。而每次的迭代需要确定：

(1) 迭代正步长 $\eta^{(k)}$；(2) 迭代方向 $S(x_k)\in\mathbf{R}^n$ 使得

$$f\left(x_{k+1}\right)=f\left(x_k+\eta^{(k)}S(x_k)\right)<f\left(x_k\right)$$

而在以上的过程中，确定一个能够使得函数值下降的方法 $S(x_k)\in\mathbf{R}^n$，至关重要。因为，只要确定了下降的方法，对任何正步长，所有算法都是朝着由 x_k 和方向 $S(x_k)$ 所确定的直线上的使得函数下降的方向移动。

称 $\nabla f\left(x_k\right)=\left[\begin{array}{cccc}\dfrac{\partial f\left(x_k\right)}{\partial x_1} & \dfrac{\partial f\left(x_k\right)}{\partial x_2} & \cdots & \dfrac{\partial f\left(x_k\right)}{\partial x_n}\end{array}\right]^T\in\mathbf{R}^n$ 为函数 $f(x)$在x_k的梯度。已经证明：负梯度方向，是使得函数值下降的方法。

如果取 $S(x_k)=-\nabla f(x_k)$，所形成的算法，称为梯度下降法。而且已经证明：它是所有其它下降方向中，使得函数值下降最快的方向，因而该方法又称之为最速下降法。

如果步长再如下确定：$\eta^{(k)}$

$$\eta^{(k)}=\arg\min_{\eta^{(k)}>0} f\left(x_k+\eta^{(k)}S(x_k)\right)$$

则形成了真正意义上的最速下降法。

最速下降法算法过程

求解问题：$\min_{x\in D\subset \mathbf{R}^n} f(x)$

步骤1：置k=0，置迭代初值 x_0，给定迭代误差精度要求 $\varepsilon > 0$，进入步骤2；

步骤2：计算梯度 $\nabla f(x_k)$，若 $\|\nabla f(x_k)\| < \varepsilon$，则置最有解为 $x^* = x_k$，进入 步骤4；否则，置 $d_k = -\nabla f(x_k)$，并求一维最优解 $\lambda_k = \arg\min_{\lambda>0} f(x_k + \lambda d_k)$，进入步骤3；

步骤3：计算 $x_{k+1} = x_k + \lambda_k d_k$，并置 k:=k+1，返回步骤2；

步骤4：结束。

例：用最速下降法求解极值问题：

$$\min_{x \in \mathbf{R}^2} f(x) = x_1^2 + 2x_2^2 - 2x_1x_2 - 2x_2$$

我们计划用最速下降法进行求解。

复合函数的求导公式：到了复合函数：

$$f(t)=g\left(y_1(t),y_2(t),\cdots,y_n(t)\right)$$

则有：

$$\frac{df(t)}{dt}=\sum_{k=1}^{n}\frac{\partial g}{\partial y_k}\frac{\partial y_k}{\partial t}$$

其中t和y_i都是标量。

$$\frac{\partial E}{\partial v_{li}}=\sum_{k=1}^{p}\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial v_{li}}$$

由于在$y_1,y_2,\ldots,y_p$中，只有y_l含有v_{li}，所以有

$$\frac{\partial y_k}{\partial v_{li}}=0,\qquad k=1,2,\cdots,l-1,l+1,\cdots,p$$

故

$$\frac{\partial E}{\partial v_{li}}=\sum_{k=1}^{p}\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial v_{li}}=\sum_{\substack{k=1\\k\neq l}}^{p}\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial v_{li}}+\frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial v_{li}}=\frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial v_{li}}\qquad(4.2.6)$$

而由（4.2.3）知，

$$\frac{\partial E}{\partial y_l}=\frac{\partial}{\partial y_l}\left[\frac{1}{2}\sum_{k=1}^{p}(\bar{y}_k-y_k)^2\right]=\frac{\partial}{\partial y_l}[\frac{1}{2}\sum_{\substack{k=1\\k\neq l}}^{p}(\bar{y}_k-y_k)^2]+\frac{\partial}{\partial y_l}[\frac{1}{2}(\bar{y}_l-y_l)^2]$$

$$=\frac{\partial}{\partial y_l}[\frac{1}{2}(\bar{y}_l-y_l)^2]=-(\bar{y}_l-y_l)\qquad(4.2.7)$$

根据(4.2.2),

$$\frac{\partial y_l}{\partial v_{li}} = \frac{\partial y_l}{\partial net_l^2}\frac{\partial net_l^2}{\partial v_{li}} = f'(net_l^2)\cdot z_i \qquad (4.2.8)$$

将(4.2.7)和(4.2.8)代人(4.2.6)得到

$$\frac{\partial E}{\partial v_{li}} = -(\bar{y}_l - y_l)\cdot f'(net_l^2)\cdot z_i$$

记

$$\delta_l = -(\bar{y}_l - y_l)\cdot f'(net_l^2)$$

则

$$\frac{\partial E}{\partial v_{li}} = \delta_l \cdot z_i \qquad (4.2.9)$$

对隐含层节点公式的推到：E是$y_1,y_2,\ldots,y_p$的多元函数，而每个y_l是z_k(k=1,2,…,m)的函数，故

$$\frac{\partial E}{\partial w_{ij}}=\sum_{l=1}^{p}\frac{\partial E}{\partial y_l}\frac{\partial y_l}{w_{ij}}=\sum_{l=1}^{p}\frac{\partial E}{\partial y_l}\sum_{k=1}^{m}\frac{\partial y_l}{z_k}\frac{\partial z_k}{\partial w_{ij}} \qquad (4.2.10)$$

但由于k≠i时，对任何的j，z_k均不含有w_{ij}，即有

$$\frac{\partial z_k}{\partial w_{ij}}=0,\ (k=1,2,\cdots,i-1,i+1,\cdots,m)$$

这样就有

$$\sum_{k=1}^{m}\frac{\partial y_l}{z_k}\frac{\partial z_k}{\partial w_{ij}}=\sum_{\substack{k=1\\k\neq i}}^{m}\frac{\partial y_l}{z_k}\frac{\partial z_k}{\partial w_{ij}}+\frac{\partial y_l}{z_i}\frac{\partial z_i}{\partial w_{ij}}=\frac{\partial y_l}{z_i}\frac{\partial z_i}{\partial w_{ij}} \qquad (4.2.11)$$

将(4.2.11)代人(4.2.10)得到

$$\frac{\partial E}{\partial w_{ij}} = \sum_{l=1}^{p} \frac{\partial E}{\partial y_l} \frac{\partial y_l}{z_i} \frac{\partial z_i}{\partial w_{ij}} \qquad (4.2.12)$$

根据(4.2.2)有

$$\frac{\partial y_l}{\partial z_i} = \frac{\partial y_l}{\partial net_l^2} \frac{\partial net_l^2}{\partial z_i} = f'(net_l^2) \cdot v_{li} \qquad (4.2.13)$$

根据(4.2.1)有

$$\frac{\partial z_i}{\partial w_{ij}} = \frac{\partial z_i}{\partial net_i^1} \frac{\partial net_i^1}{\partial w_{ij}} = f'(net_i^1) \cdot x_j \qquad (4.2.14)$$

将(4.2.7)，(4.2.13)和(4.2.14)代人到(4.2.12)得

$$\frac{\partial E}{\partial w_{ij}} = -\sum_{l=1}^{p} (\bar{y}_l - y_l) \cdot f'(net_l^2) \cdot v_{li} \cdot f'(net_i^1) \cdot x_j \quad = f'(net_i^1) \cdot (\sum_{l=1}^{p} \delta_l \cdot T_{li}) \cdot x_j$$

再记

$$\delta_i^1 = f'(net_i^1)\cdot\sum_{l=1}^{p}\delta_l\cdot v_{li}$$

则有

$$\frac{\partial E}{\partial w_{ij}} = \delta_i^1 x_j \qquad (4.2.15)$$

按梯度下降法进行权值修正，即根据(4.2.9)和(4.2.15)取权值的修正为

$$\Delta v_{li} = -\eta\frac{\partial E}{\partial v_{li}} = -\eta\delta_l^2 z_i \qquad \delta_l^2 = -(\bar{y}_l - y_l)\cdot f(net_l^2)$$

$$\Delta w_{ij} = -\eta'\frac{\partial E}{\partial w_{ij}} = -\eta'\delta_i^1 x_j \qquad \delta_i^1 = f'(net_i^1)\cdot\sum_{l=1}^{p}\delta_l^2\cdot v_{li}$$

公式汇总：

1) 对输出节点的权值修正为

$$v_{li}(k+1)=v_{li}(k)+\Delta v_{li}=v_{li}(k)-\eta\delta_l^2 z_i \quad (l=1,2,\cdots,p;\ i=1,2,\cdots,m)$$

其中

$$\delta_l^2=-(\bar{y}_l-y_l)\cdot f(net_l^2)$$

2) 对隐含层节点的权值修正为

$$w_{ij}(k+1)=w_{ij}(k)+\Delta w_{ij}=w_{ij}(k)-\eta'\delta_i^1 x_j \quad (i=1,2,\cdots m;\ j=1,2,\cdots,n)$$

其中

$$\delta_i^1=f'(net_i^1)\cdot\sum_{l=1}^{p}\delta_l^2\cdot v_{li}$$

阈值的修正：阈值 θ 也是一个变化的值，在修正权值的同时也修正它，原理同修正权值一样。

3) 对输出节点阈值修正的推导

$$\frac{\partial E}{\partial \xi_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial \xi_l} \tag{4.2.16}$$

而

$$\frac{\partial y_l}{\partial \xi_l} = \frac{\partial y_l}{\partial net_l^2}\frac{\partial net_l^2}{\partial \xi_l} = f'(net_l^2)\cdot(-1) = -f'(net_l^2) \tag{4.2.17}$$

将(4.2.7)和(4.2.17)代人(4.2.16)得到

$$\frac{\partial E}{\partial \xi_l} = (\bar{y}_l - y_l) f'(net_l^2) = -\delta_l^2$$

输出节点阈值的修正也取梯度下降方向，即取为

$$\Delta\xi_l = -\eta\frac{\partial E}{\partial \xi_l} = \eta\delta_l^2$$

于是得到输出节点阈值的修正公式为

$$\xi_l(k+1) = \xi_l(k) + \eta\delta_l^2 \qquad (l = 1,2,\cdots,p)$$

4) 对隐含层节点阈值修正的公式推导：

$$\frac{\partial E}{\partial \theta_i} = \sum_{l=1}^{p}\frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial \theta_i} = \sum_{l=1}^{p}\frac{\partial E}{\partial y_l}\sum_{k=1}^{m}\frac{\partial y_l}{\partial z_k}\frac{\partial z_k}{\partial \theta_i} \qquad (4.2.18)$$

同样，当k≠i时，由(4.2.1)看出z_k不含 θ_i，这样就有

$$\sum_{k=1}^{m}\frac{\partial y_l}{\partial z_k}\frac{\partial z_k}{\partial \theta_i} = \sum_{\substack{k=1\\k\neq i}}^{m}\frac{\partial y_l}{\partial z_k}\frac{\partial z_k}{\partial \theta_i} + \frac{\partial y_l}{\partial z_i}\frac{\partial z_i}{\partial \theta_i} = \frac{\partial y_l}{\partial z_i}\frac{\partial z_i}{\partial \theta_i}$$

将上式代人(4.2.18)得到

$$\frac{\partial E}{\partial \theta_i} = \sum_{l=1}^{p} \frac{\partial E}{\partial y_l} \frac{\partial y_l}{\partial z_i} \frac{\partial z_i}{\partial \theta_i} \tag{4.2.19}$$

根据(4.2.1)有

$$\frac{\partial z_i}{\partial \theta_i} = \frac{\partial z_i}{\partial net_i^1} \frac{\partial net_i^1}{\partial \theta_i} = f'(net_i^1) \cdot (-1) = -f'(net_i^1)$$

将上式及(4.27)和(4.2.13)代人(4.2.19)得到

$$\frac{\partial E}{\partial \theta_i} = \sum_{l=1}^{p} (\bar{y}_l - y_l) \cdot f'(net_l^2) \cdot v_{li} \cdot f'(net_i^1) = -f'(net_i^1) \sum_{l=1}^{p} \delta_l^2 T_{li} = -\delta_i^1$$

于是取修正项为

$$\Delta \theta_i = -\eta' \frac{\partial E}{\partial \theta_i} = \eta' \delta_i^1$$

修正公式为

$$\theta_i(k+1)=\theta_i(k)+\eta'\delta_i^1$$

激励函数的导数公式：

由，$f(x)=\dfrac{1}{1+e^{-x}}$ 知

$$f'(x)=\frac{e^{-x}}{(1+e^{-x})^2}=\frac{1+e^{-x}-1}{(1+e^{-x})^2}$$

$$=\frac{1}{(1+e^{-x})}+\frac{-1}{(1+e^{-x})^2}=\frac{1}{(1+e^{-x})}[1-\frac{1}{(1+e^{-x})}]$$

即有

$$f'(x)=f(x)[1-f(x)]$$

所以，对输出节点，由(4.2.2)，知

$$f'(net_l^2) = f(net_l^2)[1 - f(net_l^2)] = y_l(1 - y_l)$$

对隐含层节点，由(4.2.2)，知

$$f'(net_i^1) = f(net_i^1)[1 - f(net_i^1)] = z_i(1 - z_i)$$

前向神经网络权值阈值修正算法:

(1) 置各权值和阈值的初始值$v_{li}(0)$，$w_{ij}(0)$，隐含层神经元的阈值$\theta_i(0)$和输出层节点的阈值$\xi_l(0)$;设定修正项调节参数η和η'及期望误差$\varepsilon > 0$。

(2)输入样本训练集 $\{\mathbf{x}_q, \bar{\mathbf{y}}_q\}$,$q = 1,2,\cdots,s$ (即设有s对样本)，对每个样本进行如下的(3)~(4).

(3)计算网络在k步迭代的各项数据，为此，按如下先后顺序计算

$$net_i^1(k) = \sum_{j=1}^{n} w_{ij}(k)x_j - \theta_i(k) \qquad z_i(k) = f(net_i^1(k))$$

$$net_l^2(k) = \sum_{i=1}^{m} v_{li}(k)z_i(k) - \xi_l(k) \qquad y_l(k) = f(net_l^2(k))$$

$$f'(net_l^2(k)) = y_l(k)(1 - y_l(k)) \qquad \delta_l^2(k) = -(\bar{y}_l - y_l(k)) \cdot f'(net_l^2(k))$$

$$f'(net_i^1(k)) = z_i(k)(1 - z_i(k)) \qquad \delta_i^1(k) = f'(net_i^1(k)) \cdot \sum_{l=1}^{p} \delta_l^2(k) \cdot v_{li}(k)$$

于是，对输出节点和隐含层节点的权值和阈值按如下方式修正

$$v_{li}(k+1)=v_{li}(k)-\eta\delta_l^2(k)z_i(k) \qquad w_{ij}(k+1)=w_{ij}(k)-\eta'\delta_i^1(k)x_j$$

$$\xi_l(k+1)=\xi_l(k)+\eta\delta_l^2(k) \qquad \theta_i(k+1)=\theta_i(k)+\eta'\delta_i^1(k)$$

同时，计算出当前样本(设为第s个)的误差函数

$$E_s(k+1)=\frac{1}{2}\sum_{l=1}^{p}(\bar{y}_l-y_l(k+1))^2$$

(4)当样本结中的所有样本都经历了(3)～(4)步，即完成了一个训练周期。计算

$$E(k+1)=\sum_{s=1}^{q}E_s(k+1)$$

如果 $E(k+1)\le\varepsilon$ ，则当前 $k+1$ 步所得到的权值和阈值 $T_{li}(k+1),\ w_{ij}(k+1),\ \xi_l(k+1),\ \theta_i(k+1)$ 作为网络训练成功的权值和阈值，计算结束退出。否则返回第2步。

注：以上计算均要求对 $j=1,2,\cdots,n;\ i=1,2,\cdots,m$ 和 $l=1,2,\cdots,p$ 进行。

其它神经网络简介和基于神经网络的控制

1. 递归神经网络

- 人工神经网络具有自适应和自学习的特点，被广泛的应用于各个领域；
- 一个神经网络如前向神经网络所建立的是输入/输出之间的一种静态关系；
- 能完成系统动态特性的神经网络称为递归神经网络或者反馈神经网络。

在介绍几类递推神经网络的结果之前，做为与前向神经网络的对比和过度，首先介绍一下有前向网络构成的非递归单步延时动态神经网络的结构。

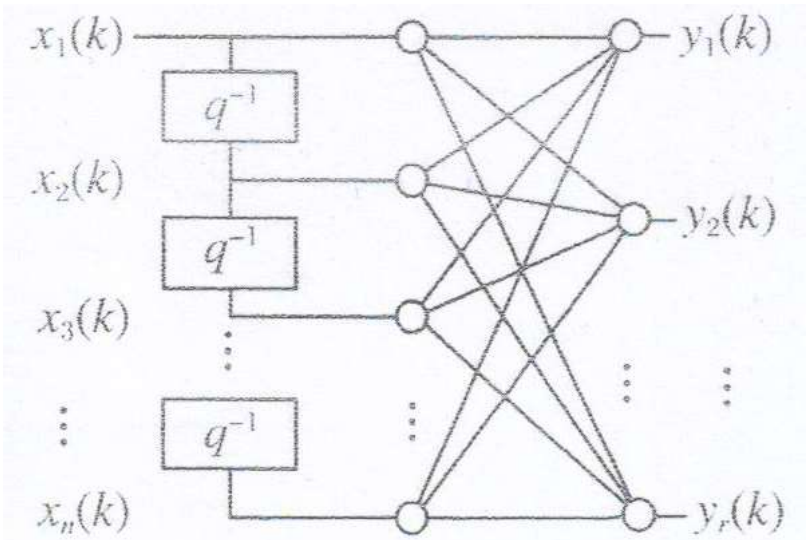


图 4.1　单步延时线性动态网络

它的作用是将输入信号经过一步或者多步的时间延时后，作为网络的输入信号，从而使得网络具有了动态特性。

根据图**4.1**，我们有

$$x_1(k)=x(k),x_2(k)=x(k-1),\cdots,x_n(k)=x(k-n+1)$$

于是网络的输入*/*输出关系为

$$y_i(k)=\sum_{j=1}^{n}w_{ij}x(k-j-1)+\theta_i,i=1,2,\cdots,r$$

单输入单输出线性动态系统，可以有如下的单步延时线性动态网络训练得到：

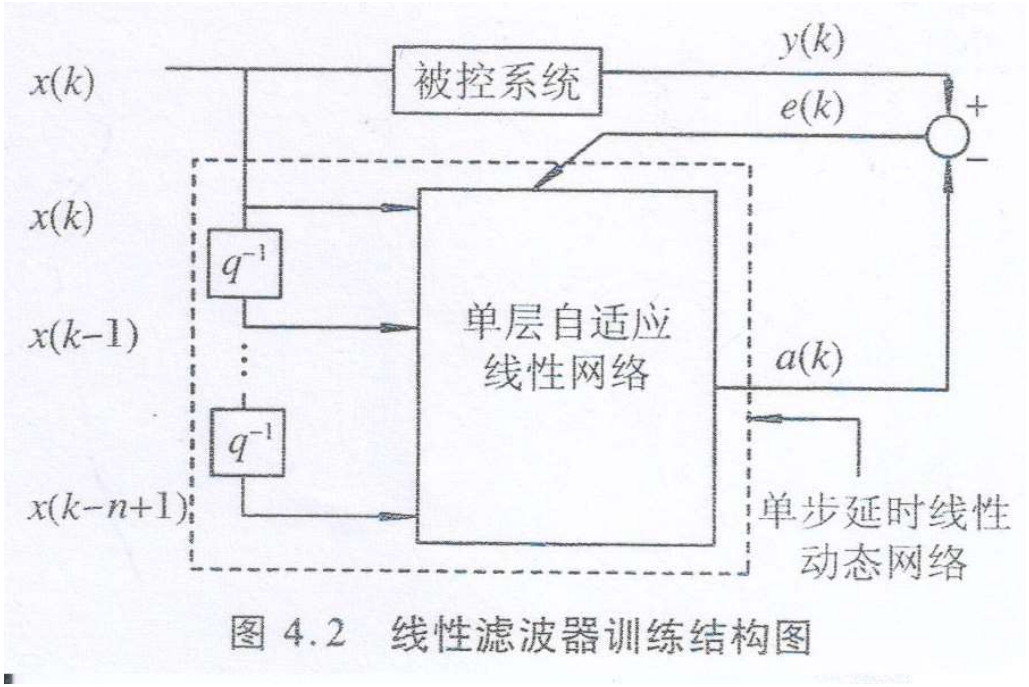


图 4.2　线性滤波器训练结构图

即使得网络的输入输出关系是

$$a(k)=c_1x(k)+c_2x(k-1)+\cdots+c_{n-1}x(k-n+1)$$

然后，按某种标注，确定系数 $c_i(i=1,\cdots,n-1)$ 使得 $e(k)=y(k)-a(k)$ 趋于零。

2. 全局反馈型递推神经网络

1. ARX网络

对于多层前向网络，对网络的输出反馈到输入端，并加上单步延时因子q^{-1}，就得到自反馈单步延时动态网络，该类网络也称之为ARX(Auto Regressive with exogenous variable)网络。网络结构如下：

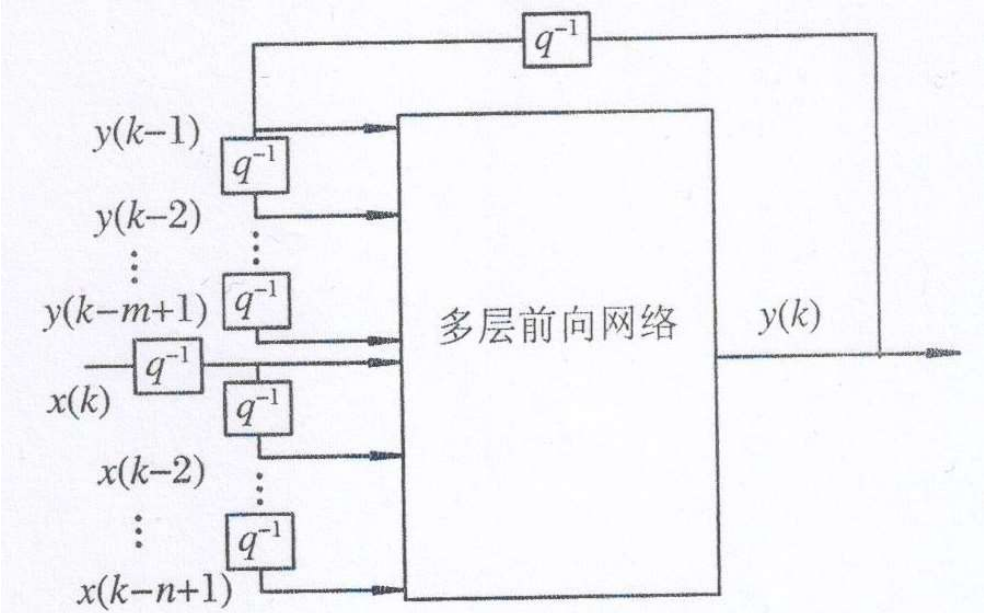


图 4.3　ARX的网络结构

选择激活函数函数是线性函数，得到网络的输入输出关系表达式为

$$y(k)=a_1y(k-1)+a_2x(k-2)+\cdots+a_{m-1}x(k-m+1)$$
$$+b_1x(k-1)+b_2x(k-2)+\cdots+b_{n-1}x(k-n+1)$$

2. Hopfield 网络

Hopfield 网络是最简单的全反馈网络，其网络结构图如图4.4所示

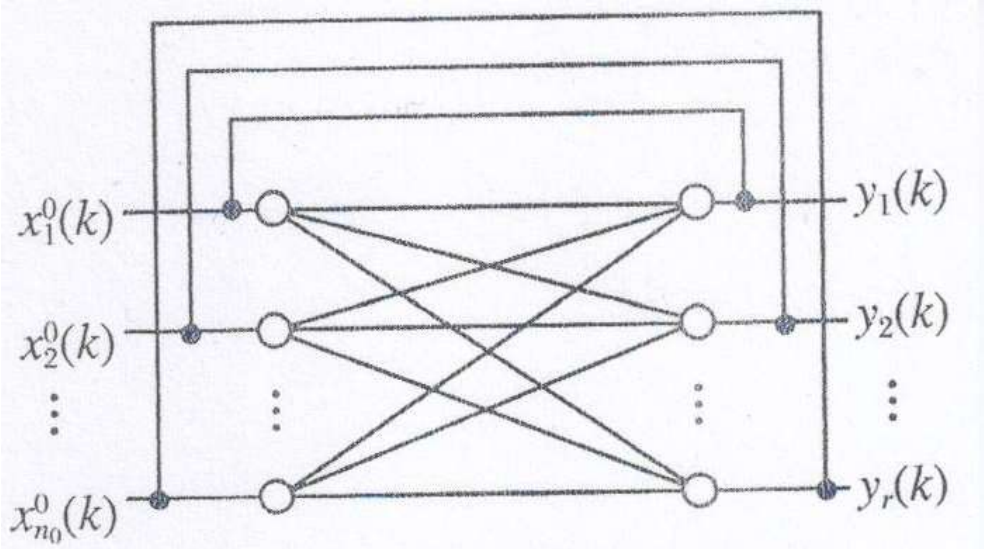


图 4.4　Hopfield 网络结构图

Hopfield网络已经被广泛的应用于联系记忆和优化计算中。

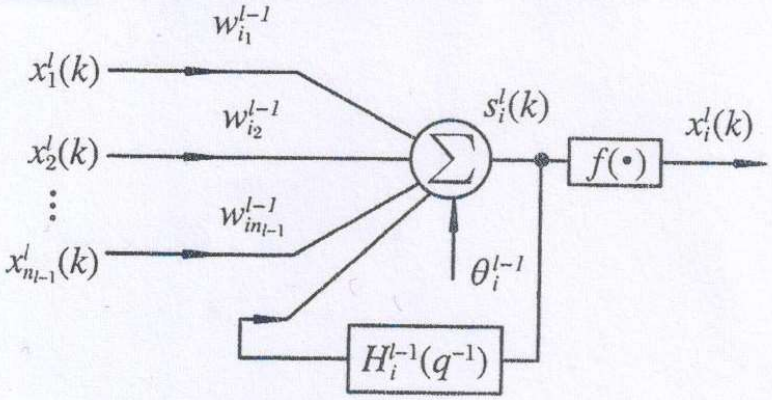
3. 前向递归神经网络

1. 局部连接递归神经网络

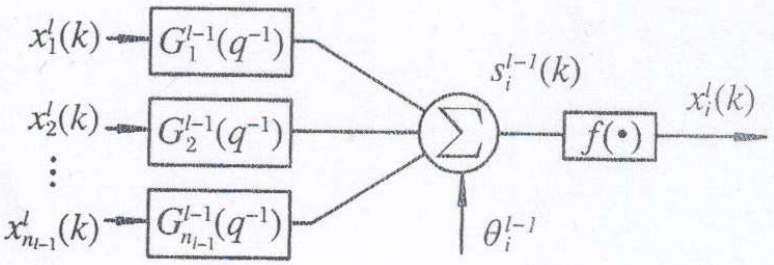
这类网络的特点是：神经元的反馈仅作用在自身上，不会作用到其它神经元上。

因而，这类神经网络，主要体现在神经元的结构上，或者说主要体现在神经元反馈的位置上。

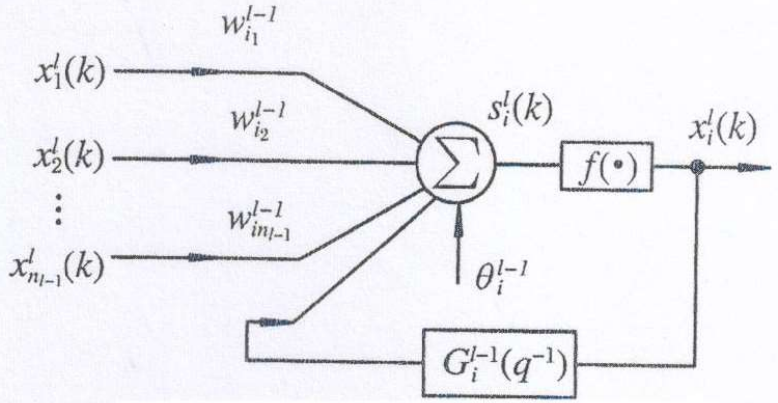
常简的有四类反馈：激活反馈、突触反馈、输出反馈和混合型反馈。



(a) 激活反馈型



(b) 突触反馈型



(c) 输出反馈型

图 4.7　典型局部连接递归网络

从图4.7(a)可以看出，计划反馈是由激活函数的输入端反馈加权到输入端。它的输入输出关系是：

$$x_i^l(k) = f\left(s_i^l(k)\right)$$

$$s_i^l(k) = \sum_{j=1}^{n^{l-1}} w_{ij}^{l-1}(k)\, x_j^{l-1}(k) + H_i^{l-1}\left(q^{-1}\right) s_i^l(k) - \theta_i^{l-1}$$

$$H_i^{l-1}\left(q^{-1}\right) = \sum_{n_i=0}^{n_{Z_i}^{l-1}} b_{n_i}^{l-1} q^{-n_i}$$

$$x_i^l(k) = f\left(s_i^l(k)\right)$$

$$s_i^l(k) = \sum_{j=1}^{n^{l-1}} G_{ij}^{l-1}\left(q^{-1}\right) x_j^{l-1}(k) - \theta_i^{l-1}$$

针对突出反馈，有

$$G_{ij}^{l-1}\left(q^{-1}\right) = \frac{\sum_{n_{ij}=0}^{n_{Z_{ij}}^{l-1}} b_{n_{ij}}^{l-1} q^{-n_{ij}}}{\sum_{m_{ij}=0}^{m_{P_{ij}}^{l-1}} a_{m_{ij}}^{l-1} q^{-m_{ij}}}$$

输出反馈的特点：从神经网络的输出端反将反馈引入到自身非线性激励函数的输入端：

$$x_i^l(k) = f\left(s_i^l(k)\right)$$

$$s_i^l(k) = \sum_{j=1}^{n^{l-1}} w_{ij}^{l-1}(k)\, x_j^{l-1}(k) + G_i^{l-1}\left(q^{-1}\right) x_i^l(k) - \theta_i^{l-1}$$

$$G_i^{l-1}\left(q^{-1}\right) = H_i^{l-1}\left(q^{-1}\right) = \sum_{n_i=0}^{n_{Z_i}^{l-1}} b_{n_i}^{l-1} q^{-n_i}$$

如果 $G_i^{l-1}\left(q^{-1}\right) = q^{-1}$，就变成了单位延时输出反馈。

基于神经网络的PID非线性自适应控制(NLPIDC)

NLPIDC结构

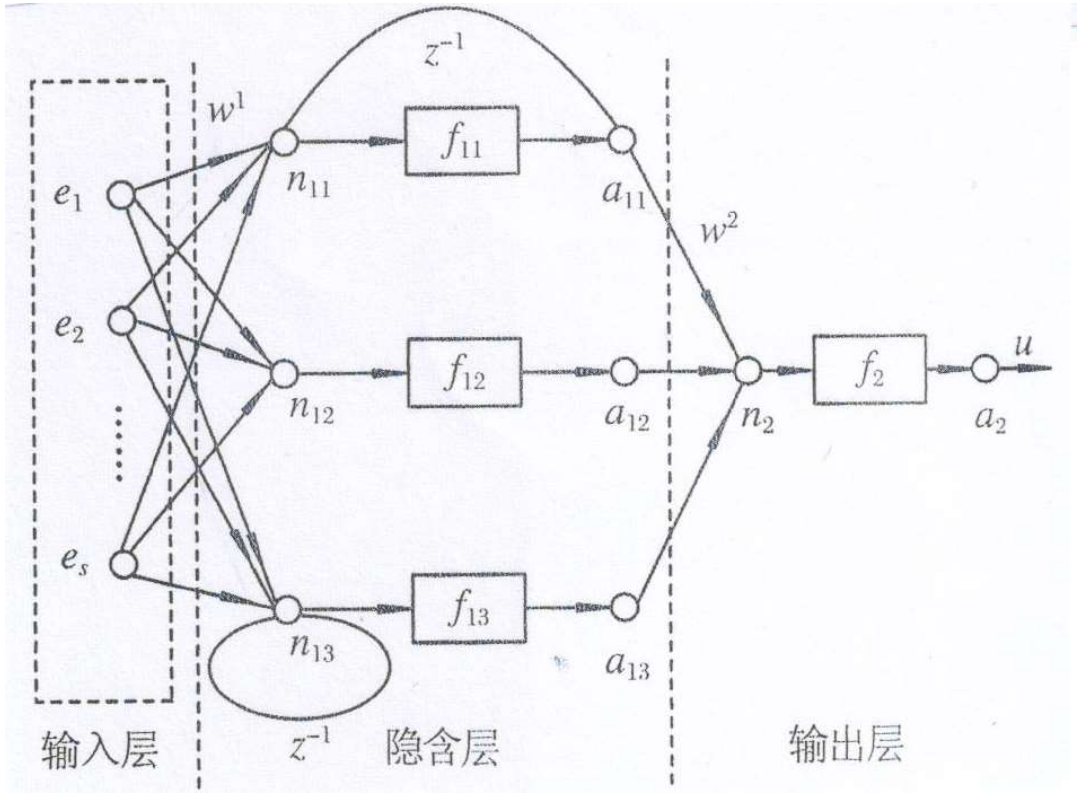


图4.8 NLPIDC结构

NLPIDC由输入层、隐含层和输出层构成，其中输入成是控制系统的s个输出误差；隐含层有3个节点，其中两个节点分别带有输出延时反馈和激活延时反馈；输出节点只有一个，它输出控制系统的控制量。

NLPIDC的输入输出关系

$$\begin{cases} n_{11}(k)=\sum_{j=1}^{s} w_{1j}^{1}(k)e_j(k)+a_{11}(k-1) \\ n_{12}(k)=\sum_{j=1}^{s} w_{2j}^{1}(k)e_j(k) \\ n_{13}(k)=\sum_{j=1}^{s} w_{3j}^{1}(k)e_j(k)-\sum_{j=1}^{s} w_{3j}^{1}(k-1)e_j(k-1) \end{cases}$$

n_{11}具有积分作用，而n_{13}具有微分作用

$$n_2(k) = f_{11}\left(n_{11}(k)\right) + f_{12}\left(n_{12}(k)\right) + f_{13}\left(n_{13}(k)\right)$$

$$u(k) = f_2\left(n_2(k)\right)$$

NLPIDC权值调整算法

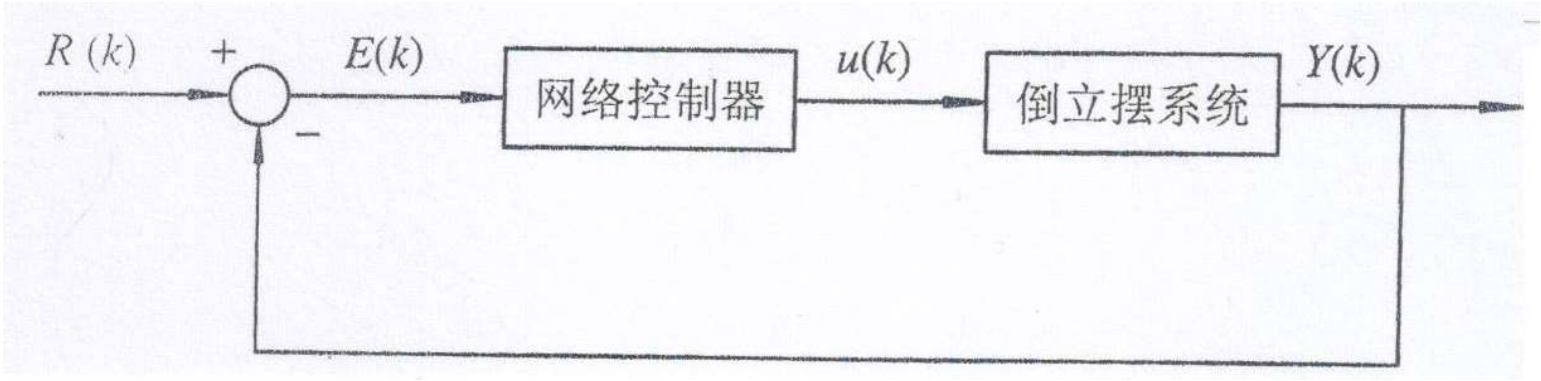


图4.9 NLPIDC闭环控制

设控制系统的输出变量为 $y_j(k)(j=1,\cdots,s)$，参考输入 $r_j(k)(j=1,\cdots,s)$，构造

$$E(k)=\frac{1}{2}\left\|R(k)-Y(k)\right\|^2=\frac{1}{2}\sum_{j=1}^{s}\left(r_j(k)-y_j(k)\right)^2$$

问题转化为求解

$$\min_{w_{ij}^1(k)} E(k)=\frac{1}{2}\sum_{j=1}^{s}\left(r_j(k)-y_j(k)\right)^2=\frac{1}{2}\sum_{j=1}^{s}e_j^2(k)$$

基于BP网络误差反馈的基本原理，进行如下求解：

首先

$$\frac{\partial E(k)}{\partial w_{ij}^1(k)}=\frac{\partial E(k)}{\partial n_2(k)}\frac{\partial n_2(k)}{\partial w_{ij}^1(k)} \qquad (4.2.20)$$

其次

$$\frac{\partial E(k)}{\partial n_2(k)}=-\left(\sum_{j=1}^{s}e_j(k)\frac{\partial y_j(k)}{\partial u(k)}\frac{\partial u(k)}{\partial n_2(k)}\right)$$

在上式中，考虑到$y_j(k)$与$u(k)$间的复杂关系，近似处理 $\frac{\partial y_j(k)}{\partial u(k)}$ 如下：

$$\frac{\partial y_j(k)}{\partial u(k)} \approx \frac{y_j(k)-y_j(k-1)}{u(k)-u(k-1)} \qquad \frac{\partial u(k)}{\partial n_2(k)} = \frac{\partial f\left(n_2(k)\right)}{\partial n_2(k)} = \dot{f}_2[k]$$

故

$$\frac{\partial E(k)}{\partial n_2(k)} = -\left(\sum_{j=1}^{s} e_j(k)\frac{\partial y_j(k)}{\partial u(k)}\dot{f}_2[k]\right) \qquad (4.2.21)$$

由于 $n_2(k) = f_{13}\left(n_{13}(k)\right) + f_{12}\left(n_{12}(k)\right) + f_{11}\left(n_{11}(k)\right)$

$$\frac{\partial n_2(k)}{\partial w_{ij}^1(k)} = \dot{f}_{13}[k]\frac{\partial n_{13}(k)}{w_{ij}^1(k)} + \dot{f}_{12}[k]\frac{\partial n_{12}(k)}{w_{ij}^1(k)} + \dot{f}_{11}[k]\frac{\partial n_{11}(k)}{w_{ij}^1(k)}$$

$$\frac{\partial n_2(k)}{\partial w_{1j}^1(k)} = \dot{f}_{13}[k]\frac{\partial n_{13}(k)}{\partial w_{1j}^1(k)} + \dot{f}_{12}[k]\frac{\partial n_{12}(k)}{\partial w_{1j}^1(k)} + \dot{f}_{11}[k]\frac{\partial n_{11}(k)}{\partial w_{1j}^1(k)}$$

由于

$$\frac{\partial n_{13}(k)}{\partial w_{1j}^1(k)} = 0 \qquad \frac{\partial n_{12}(k)}{\partial w_{1j}^1(k)} = 0$$

$$\frac{\partial n_{11}(k)}{\partial w_{1j}^1(k)} = \frac{\partial\left(\sum_{j=1}^{s} w_{1j}^1(k)e_j(k) + a_{11}(k-1)\right)}{\partial w_{1j}^1(k)} = e_j(k)$$

所以

$$\frac{\partial n_2(k)}{\partial w_{1j}^1(k)} = \dot{f}_{11}[k]e_j(k)$$

$$\frac{\partial n_2(k)}{\partial w_{2j}^1(k)} = \dot{f}_{13}[k]\frac{\partial n_{13}(k)}{\partial w_{2j}^1(k)} + \dot{f}_{12}[k]\frac{\partial n_{12}(k)}{\partial w_{2j}^1(k)} + \dot{f}_{11}[k]\frac{\partial n_{11}(k)}{\partial w_{2j}^1(k)}$$

由于

$$\frac{\partial n_{13}(k)}{\partial w_{2j}^1(k)} = 0,\ \frac{\partial n_{11}(k)}{\partial w_{2j}^1(k)} = 0,\ \frac{\partial n_{12}(k)}{\partial w_{2j}^1(k)} = \frac{\partial\left(\sum_{j=1}^{s} w_{2j}^1(k)e_j(k)\right)}{\partial w_{2j}^1(k)} = e_j(k)$$

所以

$$\frac{\partial n_2(k)}{\partial w_{2j}^1(k)} = \dot{f}_{12}[k]e_j(k)$$

$$\frac{\partial n_2(k)}{\partial w_{3j}^1(k)} = \dot{f}_{13}[k]\frac{\partial n_{13}(k)}{\partial w_{3j}^1(k)} + \dot{f}_{12}[k]\frac{\partial n_{12}(k)}{\partial w_{3j}^1(k)} + \dot{f}_{11}[k]\frac{\partial n_{11}(k)}{\partial w_{3j}^1(k)}$$

由于

$$\frac{\partial n_{11}(k)}{\partial w_{3j}^1(k)} = 0,\ \frac{\partial n_{12}(k)}{\partial w_{3j}^1(k)} = 0, \frac{\partial n_{13}(k)}{\partial w_{2j}^1(k)} = e_j(k)$$

所以

$$\frac{\partial n_2(k)}{\partial w_{3j}^1(k)} = \dot{f}_{13}[k]e_j(k)$$

综合得到

$$\frac{\partial n_2(k)}{\partial w_{ij}^1(k)} = \dot{f}_{1i}[k]e_j(k), (i=1,2,3;\ j=1,\cdots,s) \qquad (4.2.22)$$

(4.2.21)和(4.2.22)代入到(4.2.20)，得到

$$\frac{\partial E(k)}{\partial w_{ij}^{1}(k)}=-\left(\sum_{j=1}^{s}e_j(k)\frac{\partial y_j(k)}{\partial u(k)}\dot{f}_2[k]\right)\dot{f}_{1i}[k]e_j(k) \qquad (4.2.23)$$

其中，记

$$\dot{f}_2[k]=\frac{df_2\left(n_2(k)\right)}{dn_2(k)},\ \dot{f}_{1i}[k]=\frac{df_{1i}\left(n_{1i}(k)\right)}{dn_{1i}(k)}$$

于是，可以得到权值修改公式如下：

$$w_{ij}^{1}(k+1)=w_{ij}^{1}(k)-\eta_{ij}\frac{\partial E(k)}{\partial w_{ij}^{1}(k)} \qquad (4.2.24)$$

其中 $\frac{\partial E(k)}{\partial w_{ij}^{1}(k)}$ 由(4.2.23)计算。

NLPIDC实时在线控制策略步骤

(1) 将网络控制器与被控对象串联，并连接成如图4.9所示的闭环系统；

(2) 对$w_{ij}^{1}(0), w_{ij}^{1}(-1), e_j(-1)(i=1,2,3; j=1,\cdots,s)$和$u(0)$赋初值，在$u(0)$的激励下，实时测量出被控对象的响应输出$y_j(0)$，形成$e_j(0)=r_j(0)-y_j(0)(j=1,\cdots,s)$，进入步骤(3);

(3) 按()计算出 $n_{11}(0), n_{12}(0), n_{13}(0)$

$$n_2(0), u(0)$$