第二章 前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

- 引言
- 采用硬限幅函数时单个神经元的分类功能
- 采用线性函数时单个神经元的最小均方分类学习算法
- 采用硬限幅函数、线性函数的前向多层神经网络
- 采用S形函数的前向多层神经网络及其BP学习算法
- 前向多层神经网络的应用
- 小结

§2.1 引言

XI'AN JIAOTONG UNIVERSITY

一、理想化静态模型

本章将要讨论人工神经网络的一个子系统，前向多层神经网络。其理想化静态模型如图2.1所示。

网络输入N维矢量：$\mathbf{X}=[x_0\ x_1\cdots x_{N-1}]$

网络输出M维矢量：$\mathbf{Y}=[y_0\ y_1\cdots y_{M-1}]$

输入一个特定的$\mathbf{X}$，就得到一个特定的$\mathbf{Y}$。前向多层神经网络的输出和输入之间是一种映射关系。

前向多层神经网络涉及神经网络的两项基本功能：联想与分类。这两种功能都可以用输入到输出的映射关系来描述。

为了便于讨论，下面分别列出这两种功能的映射关系。

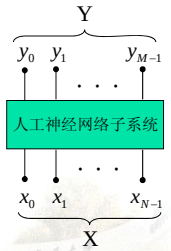


图2.1 前向多层神经网络的理想化静态模型

§2.1 引言

XI'AN JIAOTONG UNIVERSITY

二、前向多层神经网络的基本功能及其映射关系

1. 分类功能的映射关系

输入矢量（也可以称为观察矢量）$\mathbf{X}=[x_0\ x_1\cdots x_{N-1}]$ 是一个 N 维随机矢量。$\mathbf{X}\in R^N$，即其每一个分量可以取任意实数，显然，一个 $\mathbf{X}$ 相应于实 N 维空间中的一个点。

设所有$\mathbf{X}$可以分成 M 类，记为C_j，$j=0,1,2,\cdots,M-1$。与某个类别C_j相应的所有随机输入矢量$\mathbf{X}$的集合在R^N 中构成一个集合R_j。分类功能就是要求我们能够根据某个输入$\mathbf{X}$ 产生的输出$\mathbf{Y}$ 来判断$\mathbf{X}$ 属于哪一类。

一种简单的方案是，当 $\mathbf{X}$属于第C_j类时，令输出矢量 $\mathbf{Y}$的第 j个分量等于1，而其他分量等于0。数学表达如下：

若 $\mathbf{X}\in R_j$ ，则 $y_k=\begin{cases}1 & \text{当}k=j\\ 0 & \text{当}k\neq j\end{cases}$

§2.1 引言

XI'AN JIAOTONG UNIVERSITY

事实上，对一个实际的神经网络，严格要求实现此式会造成系统实现的困难，而且也没有必要。其实，只要尽量接近于1或0就可以了。算法可以改为：

若 $y_j=\max_k\{y_k\}$，则 $\mathbf{X}\in R_j$

2. 联想功能的映射关系

联想的作用是由输入信息唤起（或称检索、提取）与之有关联的其它信息，这些信息存储在神经网络中，当输入了某种特定信息后，在神经网络的输出端就应产生这些关联信息。联想功能可以由下列输入输出的映射关系来描述：

$$\mathbf{X}^{(s)}\rightarrow\mathbf{Y}^{(s)}$$

其中，$\mathbf{X}^{(s)}\in R^N$ ， $\mathbf{Y}^{(s)}\in R^M$

实际中往往要求，当$\mathbf{X}^{(s)}$具有少量变异或缺损，或者受到一定程度干扰时，这一关系仍然成立，即 $\mathbf{X}^{(s)}+\Delta\mathbf{X}^{(s)}\rightarrow\mathbf{Y}^{(s)}$。本章主要讨论异联想。

§2.1 引言

XI'AN JIAOTONG UNIVERSITY

第一章我们曾讲过，研究神经网络通常关注三个方面：神经元之间的连接形式、神经元功能函数以及学习（训练）规则。下面逐一介绍。

三、神经网络结构（神经元之间的连接形式）

本章所采用的神经网络结构是一种单向多层结构，其中每一层都包含若干神经元。同一层中的神经元之间没有相互的联系，而层间的信息传送只沿一个方向进行。第 $i-1$ 层的信号只送往第 i 层，而第 i 层的信号又只送往第 $i+1$ 层。

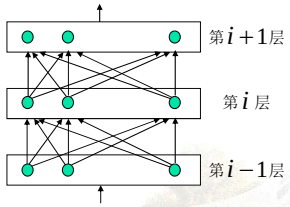


图2.4 前向多层神经网络结构示意图

§2.1 引言

XI'AN JIAOTONG UNIVERSITY

四、静态神经元模型（神经元功能函数）

将第一章讲到的PDP模型加以简化，得到一个静态的神经元模型，如图2.2所示。

输入矢量 $\mathbf{X}=[x_0\ x_1\cdots x_{N-1}]\in R^N$，各权重值也构成一个矢量 $\mathbf{W}$，$\mathbf{W}=[w_0\ w_1\cdots w_{N-1}]$。

$$\begin{cases} I=\mathbf{W}\mathbf{X}^T-\theta=\sum_{j=0}^{N-1}w_jx_j-\theta \\ y=f(I) \end{cases} \quad \cdots\cdots \text{EQ2.3}$$

其中，$[\bullet]^T$ 表示转置，$f(\bullet)$ 是一个单调非降函数，称为变换函数。本章中，变换函数将取三种形式，线性函数、硬限幅函数（或符号函数）以及S形函数（sigmoid函数），参见图2.3。

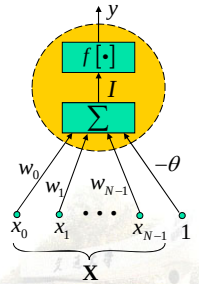


图2.2 静态神经元模型

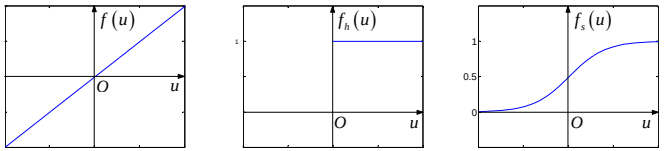
§2.1 引言

XI'AN JIAOTONG UNIVERSITY

三种变换函数的表达式为：

线性函数：$f(u)=u$ ····· EQ2.4

硬限幅函数：$f_h(u)=\begin{cases}1, & u\geq 0\\ 0, & u<0\end{cases}$ ····· EQ2.5



(a) 线性函数 (b) 硬限幅函数 (c) S形函数

图2.3 静态神经元模型的三种变换函数

§2.1 引言

XI'AN JIAOTONG UNIVERSITY

S形函数：$f(u)=\frac{1}{1+e^{-u}}$ ····· EQ2.5

如果想得到-1~1输出，对硬限幅函数可以采用 $\mathrm{sgn}[u]$，对S形函数可以采用 $\tanh[u]$。

五、对学习训练的几项要求

为了使一个前向神经网络能够完成分类或联想功能，必须将有关的信息存入其中，这就必须对其进行训练，或者说让神经网络学习。对于一个神经网络学习算法，可以提出以下几项最基本的要求。

1. **准确性** 神经网路通过学习以后，可以准确地完成希望它实现的功能（如分类或联想），所谓准确是指**按照某种准则**，使网络的实际输出与理想输出之间的误差达到最小，或达到某个最低标准。

2. **自适应性** 是指在网络学习过程中，所有的参数自动调整，不受人为干预。

§2.1 引言

XI'AN JIAOTONG UNIVERSITY

3. **收敛性与收敛速度** 经过学习，神经网络的参数应收敛到唯一的一组解上；学习的时间也不能太长，在一定的时间范围内就能达到收敛点，否则学习的开销无法接受（即学习算法的时间复杂度不能太大）。

4. **必须规定是有监督的学习还是无监督的（自组织）学习** 本章只讨论有监督学习。

5. **可推广性** 训练样本总是有限的，在工作中遇到与训练样本不同的输入时，网络应该能够可靠地完成其功能。

本章主要讨论用静态神经元模型构成的前向多层神经网络，讨论它的分类和联想功能，以及各种学习算法。

由于分类和联想功能有很多类似之处，从分析方法看也没有本质区别。所以后面的讨论和分析都以分类功能为目标。

下面先从简单的单神经元出发，进而讨论复杂的多层结构及其应用。

§2.2 采用硬限幅函数时单个神经元的分类功能

XI'AN JIAOTONG UNIVERSITY

我们先从最简单的情况出发，因此做如下假设：

假设1：输入矢量$\mathbf{X}$是二维的，即$\mathbf{X}\in R^2$，那么任何输入矢量都可以用它的两个坐标表示为$\mathbf{X}=[x_0\ x_1]$，因此，输入矢量对应于二维空间中的一个点。

假设2：被观察的客体（输入矢量）只有两类，分别记为C_1和C_2。

一、二维空间中的线性可分类

如图2.5所示，有两类输入矢量，它们分别对应于集合R_1和R_2。

显然，两个集合可以由直线 l 分开，直线 l 的方程为：

$$0.5x_0+x_1-1=0$$

R_1中的所有矢量都满足$0.5x_0+x_1-1>0$

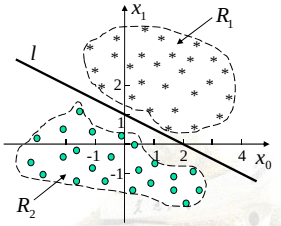


图2.5 二维空间中线性可分类的示例

§2.2 采用硬限幅函数时单个神经元的分类功能

XI'AN JIAOTONG UNIVERSITY

R_2中的所有矢量都满足： $0.5x_0+x_1-1<0$

联想到上一节的静态神经元模型（EQ2.3）

$$\begin{cases} I=\sum_{j=0}^{N-1}w_jx_j-\theta \Rightarrow w_0x_0+w_1x_1-\theta \\ y=f(I)\Rightarrow y=f_h(I) \end{cases}$$

如果令 $w_0=0.5$，$w_1=1$ 及 $\theta=1$，则

$$y=f_h\left(\sum_{j=0}^{1}w_jx_j-\theta\right)=\begin{cases}1 & \mathbf{X}\in R_1\\ 0 & \mathbf{X}\in R_2\end{cases}$$

这样，我们就可以根据神经元的输出值来判断输入矢量是属于$C_1(y=1)$还是属于$C_2(y=0)$。

以上是对二维线性可分类问题的讨论，但很容易推广到多维。

§2.2 采用硬限幅函数时单个神经元的分类功能

XI'AN JIAOTONG UNIVERSITY

若输入的是三维矢量，那么两个线性可分类之间的分界将是分割平面；而当输入矢量的维数大于3时，分界将是超平面。显然，无论输入矢量 $\mathbf{X}$ 的维数有多高，只要两个集合是线性可分的，就可以用采用硬限幅函数的单神经元按照上述方法对其进行无误的分类。

二、学习算法

对任何实际应用而言，两类之间的分割线或分割面是不可能事先确定的，也就是说，各权重系数和阈值是不能事先求得的。如果可以事先求得，就没有必要研究神经网络了。实际中，必须通过足够多的已知类别的输入矢量 $\mathbf{X}$ 的样本对神经元进行“训练”，让神经元通过学习自动地找到那些权重系数和阈值，从而确定“分界面”。这说明神经元在实现分类功能之前，必须有一个学习阶段，这是一种有监督的学习。

下面给出采用硬限幅函数时单神经元的分类学习算法，并定性证明，对线性可分类，当满足一定条件时，学习一定是收敛的。

§2.2 采用硬限幅函数时单个神经元的分类功能

XI'AN JIAOTONG UNIVERSITY

为了方便讨论，我们把EQ2.3改写为如下的形式：

$$\begin{cases} I = \mathbf{W}\mathbf{X}^T - \theta = \sum_{j=0}^{N-1} w_j x_j - \theta \Rightarrow \mathbf{W}_1\mathbf{X}_1^T \\ y = f(I) \Rightarrow y = f_h(I) \end{cases}$$

其中，$\mathbf{X}_1 = \left[\underbrace{x_0\ x_1 \cdots x_{N-1}}_{\mathbf{X}}\ 1\right]$　　$\mathbf{W}_1 = \left[\underbrace{w_0\ w_1 \cdots w_{N-1}}_{\mathbf{W}}\ -\theta\right]$

学习算法如下：（为方便起见，以后用$\mathbf{X}$和$\mathbf{W}$表示扩维后的矢量）

1. 任意设置加权矢量$\mathbf{W}$的初始值，记为$\mathbf{W}(0)$；
2. 依次送入观察矢量，记为$\mathbf{X}(k), k = 0,1,2,\cdots$，按下式修正权系数：

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha\{d(k) - y(k)\}\mathbf{X}(k), k = 0,1,2,\cdots \quad \text{EQ2.8}$$

其中，α是一个小的正实数，称为“步幅”；$d(k)$是第k个观察矢量的期望输出值，$y(k)$是实际输出值。

§2.2 采用硬限幅函数时单个神经元的分类功能

XI'AN JIAOTONG UNIVERSITY

三、学习算法收敛性的定性证明

下面我们证明，对线性可分类，只要步幅足够小且训练样本足够多时，用上面的算法调整系数，$\mathbf{W}(0)$将收敛到能给出正确分类结果的加权矢量。

为简单起见，我们先证明观察矢量为二维且分界线穿过原点的情况，在此情况下，必然有$\theta = 0$；相应地，矢量$\mathbf{X}$和$\mathbf{W}$从三维退化为二维。

按照前面的算法进行学习，到第k步时得到$\mathbf{W}(k) = [w_0(k)\ \ w_1(k)]$，由它确定的分割线为$l(k)$，此直线上的点显然满足方程$w_0(k)x_0 + w_1(k)x_1 = 0$；显然，$\mathbf{W}(k) = [w_0(k)\ \ w_1(k)]$是$l(k)$的法线。如图2.6所示，图中的直线$l$是理想分割线。$l$和$l(k)$将平面分成四个扇区，分别记为$S_i, i = 1,2,3,4$。

现在取第k个观察矢量$\mathbf{X}(k)$进行学习，以便获得$\mathbf{W}(k+1)$。显然，观察矢量$\mathbf{X}(k)$必然位于四个扇区之一，那么学习的结果将会是如下四种情况之一。

为方便起见，定义：$\varepsilon(k) = d(k) - y(k)$

§2.2 采用硬限幅函数时单个神经元的分类功能

XI'AN JIAOTONG UNIVERSITY

A. $\mathbf{X}(k) \in S_1$，有$d(k) = 1$，$y(k) = 1$。
所以$\varepsilon(k) = 0$，由EQ2.8，$\mathbf{W}(k+1) = \mathbf{W}(k)$
分割线维持不动。

B. $\mathbf{X}(k) \in S_2$，有$d(k) = 0$，$y(k) = 0$。
所以$\varepsilon(k) = 0$，由EQ2.8，$\mathbf{W}(k+1) = \mathbf{W}(k)$
分割线维持不动。

C. $\mathbf{X}(k) \in S_3$，有$d(k) = 1$，$y(k) = 0$。
所以$\varepsilon(k) = 1$，$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha\mathbf{X}(k)$

显然，在此情况下，分割线将会改变，其法线是原分割线的法线和$\alpha\mathbf{X}(k)$的矢量和。学习后的结果见下页图2.6b。

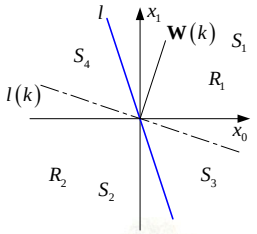


图2.6a 线性可分类学习示例

§2.2 采用硬限幅函数时单个神经元的分类功能

XI'AN JIAOTONG UNIVERSITY

D. $\mathbf{X}(k) \in S_4$，有$d(k) = 0$，$y(k) = 1$。所以$\varepsilon(k) = -1$，
由EQ2.8，$\mathbf{W}(k+1) = \mathbf{W}(k) - \alpha\mathbf{X}(k)$。
在此情况下，分割线的改变情况如图2.6c所示。

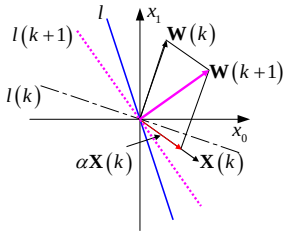


图2.6b 分割线改变示意图

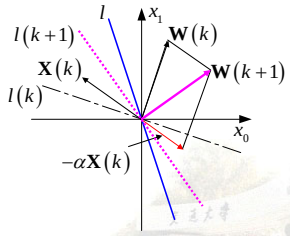


图2.6c 分割线改变示意图

§ 2.2 采用硬限幅函数时单个神经元的分类功能

XI'AN JIAOTONG UNIVERSITY

从证明过程可以看到，若 $l(k)$已经很接近最佳分界线 l，那么步幅 α 必须选得非常小，否则可能矫枉过正，使 $\mathbf{W}(k+1)$ 转过了头。因此，以上我们证明了，只要α 足够小，那么经过任何一步学习以后，$l(k+1)$或者等同于$l(k)$或者比 $l(k)$ 更趋近于 l ，不可能有其它情况。

如果两个类别之间的分界线不通过原点，即 $\theta \neq 0$ ，那么需要学习确定的参数共有三个，因此成为三维问题，这可以用在三维空间中寻找通过原点的平面来说明。这个结论可以推广到 N 为任意其它正整数的情况。

综上，我们定性证明了，如果输入矢量 $\mathbf{X}$ 分别属于两类且它们是线性可分类时，无论矢量的维数有多高，采用硬限幅函数的单个神经元按照EQ2.8的学习算法进行学习，当 α 足够小时，总能找到一个最佳的加权矢量，使神经元能够无误地完成分类任务。

证明中要求步幅 α 足够小，但如果步幅选得很小，学习速度将变得很慢，为了解决这个矛盾，可以采取变步幅的方案。

§ 2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

上一节讨论了采用硬限幅函数时单个神经元的分类功能，它可以完成任何两类的线性可分类问题。但是，在实际中，线性可分类的条件有些过于严格了。能满足这种条件的情况不是很多。图2.7给出了两种非线性可分的情况，对于这些情况，采用上一节的具有硬限幅函数的神经元及其学习算法，不能保证学习结果是收敛的；但是，我们可以采用具有线性函数的神经元来解决这种问题。

这时，我们不能要求当输入属于某个类别时，神经元的输出严格等于1或0，而只能要求输出尽可能接近于1或0。如要求实际输出值与两个理想的预期值之间的均方误差最小；我们把能实现这种目标的算法称为最小均方学习算法（LMS算法，Least Mean Square）

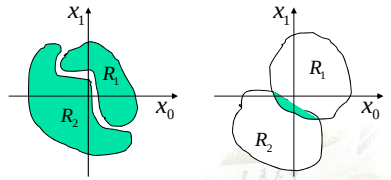


图2.7 二维空间中非线性可分类的示例

§ 2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

一、LMS算法

为方便起见，我们假设阈值为零，观察矢量用$\mathbf{X}$来表示，权矢量用$\mathbf{W}$来表示，$\mathbf{X}=[x_0\ x_1\cdots x_{N-1}]$ 、$\mathbf{W}=[w_0\ w_1\cdots w_{N-1}]$ 。在采用线性函数的条件下，神经元的输出y可以用下式表示：

$$\begin{cases} I=\sum_{j=0}^{N-1} w_j x_j = \mathbf{W}\mathbf{X}^T = \mathbf{X}\mathbf{W}^T \\ y=f(I)=I=\mathbf{W}\mathbf{X}^T=\mathbf{X}\mathbf{W}^T \end{cases}$$

设与观察矢量$\mathbf{X}$对应的期望输出为 d，那么误差$\varepsilon=d-y$。现在的任务是：寻找最佳的 $\mathbf{W}^*$ ，使与其对应的均方误差 $E[\varepsilon^2]$ 最小。其中，$E[\bullet]$ 表示取数学期望，即对所有可能出现的输入矢量的集合取统计平均。

$$E[\varepsilon^2]=E[(d-y)^2]$$
$$=E[d^2]+E[y^2]-2E[dy]\big|_{y=\mathbf{W}\mathbf{X}^T=\mathbf{X}\mathbf{W}^T}$$

§ 2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

$$=E[d^2]+E[\mathbf{W}\mathbf{X}^T\mathbf{X}\mathbf{W}^T]-2E[d\mathbf{X}\mathbf{W}^T]$$
$$=E[d^2]+\mathbf{W}E[\mathbf{X}^T\mathbf{X}]\mathbf{W}^T-2E[d\mathbf{X}]\mathbf{W}^T$$

为推导简便，做如下定义：

$E[\mathbf{X}^T\mathbf{X}]\triangleq R$，是随机矢量$\mathbf{X}$的相关矩阵，它是对称的。

$E[d\mathbf{X}]\triangleq P$，这样，我们有

$$E[\varepsilon^2]=E[d^2]+\mathbf{W}R\mathbf{W}^T-2P\mathbf{W}^T \quad \cdots\cdots\cdots\cdots \text{EQ2.15}$$

现在，就是要求得$\mathbf{W}$取何值时，上式取最小值。求极值，就是通过求导得到函数的驻点，从而得到函数的极值。而对本问题而言，是数量对矢量求导，称为求梯度。求梯度的结果是

$$\nabla_{\mathbf{W}}E[\varepsilon^2]=2\mathbf{W}R-2P \quad \cdots\cdots\cdots\cdots\cdots\cdots \text{EQ2.16}$$

从EQ2.15到EQ2.16的推导可以参考《高等数学》中对梯度的定义。

Go...

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

令EQ2.16等于零，可得

$$\mathbf{W}^* = PR^{-1} \quad \cdots\cdots\cdots\cdots\cdots\cdots \text{EQ2.17}$$

注意：当 R 不满秩时，应该求伪逆。其满秩的条件是，不相关的观察矢量的个数应该等于其维数。由于观察矢量的个数通常都很多，所以一般情况下这个条件都是满足的。

当 $\mathbf{W} = \mathbf{W}^*$ 时，$E\left[\varepsilon^2\right]$ 取得最小值。

$$E\left[\varepsilon^2\right]_{\min} = E\left[d^2\right] - 2P\mathbf{W}^{*T} + \mathbf{W}^* R \mathbf{W}^{*T} \Big|_{\mathbf{W}^* = PR^{-1}}$$
$$= E\left[d^2\right] - P\mathbf{W}^{*T}$$

至此，我们求出了最佳权矢量 $\mathbf{W}^*$，问题看似解决了，但是实际上还有些问题。(1) 求 R 和 P 需要进行大量的统计运算，这难以做到；(2) 求解 $\mathbf{W}^*$ 时需要计算 R 的逆，当输入矢量的维数很大时，需要解决高维矩阵的求逆问题。

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

为了克服这些困难，下面介绍三种递推算法。

二、Widrow的严格递推求解算法

设 $\mathbf{X}(k)$ 是第 k 步的输入矢量，对应的期望输出为 $d(k)$，

实际输出为 $y(k) = \mathbf{W}(k)\mathbf{X}^T(k) = \mathbf{X}(k)\mathbf{W}^T(k)$，误差 $\varepsilon(k) = d(k) - y(k)$

$$E\left[\varepsilon^2(k)\right] = E\left\{\left[d(k) - y(k)\right]^2\right\} = E\left\{\left[d(k) - \mathbf{X}(k)\mathbf{W}^T(k)\right]^2\right\}$$

学习算法如下：

(1) 任意设置初始加权矢量 $\mathbf{W}(0)$

(2) 对每个时序变量 k，按下式调整权矢量。

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \alpha\left\{-\nabla_{\mathbf{W}} E\left[\varepsilon^2(k)\right]\right\} \quad \cdots\cdots \text{EQ2.24}$$
$$k = 0,1,2,\cdots$$

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

公式第二项大括号中的内容指明了权矢量调整的方向，即权矢量应该向梯度的负方向调整。梯度的负方向就是误差减小的方向，权矢量调整方向的几何解释可以参见图2.8。

现在的任务就是要求出这个梯度。

$$\nabla_{\mathbf{W}} E\left[\varepsilon^2(k)\right]$$
$$= \left\{\frac{\partial E\left[\varepsilon^2(k)\right]}{\partial w_0}, \frac{\partial E\left[\varepsilon^2(k)\right]}{\partial w_1}, \cdots, \frac{\partial E\left[\varepsilon^2(k)\right]}{\partial w_{N-1}}\right\}$$
$$= E\left\{\frac{\partial \varepsilon^2(k)}{\partial w_0}, \frac{\partial \varepsilon^2(k)}{\partial w_1}, \cdots, \frac{\partial \varepsilon^2(k)}{\partial w_{N-1}}\right\}$$
$$= E\left\{2\varepsilon(k)\left[\frac{\partial \varepsilon(k)}{\partial w_0}, \frac{\partial \varepsilon(k)}{\partial w_1}, \cdots, \frac{\partial \varepsilon(k)}{\partial w_{N-1}}\right]\right\}$$

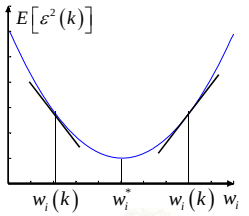


图2.8 权矢量修正方向的几何解释

More...

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

$$= E\left\{2\varepsilon(k)\left[-x_0(k), -x_1(k), \cdots, -x_{N-1}(k)\right]\right\}$$
$$= -2E\left[\varepsilon(k)\mathbf{X}(k)\right]$$

把所得结果代入EQ2.24，可得

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha E\left[\varepsilon(k)\mathbf{X}(k)\right], \quad k = 0,1,2,\cdots \quad \cdots\cdots \text{EQ2.35}$$

我们前面曾推导过这个梯度，结果是 $2\mathbf{W}R - 2P$。这两个结果其实是一致的。

$$2\mathbf{W}R - 2P = 2\mathbf{W}(k)R - 2P$$
$$= 2\left\{E\left[\mathbf{W}(k)\mathbf{X}^T(k)\mathbf{X}(k)\right] - E\left[d(k)\mathbf{X}(k)\right]\right\}$$
$$= 2E\left[\left(\mathbf{W}(k)\mathbf{X}^T(k) - d(k)\right)\mathbf{X}(k)\right]$$
$$= 2E\left[\left(y(k) - d(k)\right)\mathbf{X}(k)\right]$$
$$= -2E\left[\varepsilon(k)\mathbf{X}(k)\right]$$

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

书上的推导过程较为复杂，但包括收敛性及收敛条件的证明，证明过程我们就不讲了，结果如下：如果步幅 α 满足

$$0<\alpha<\frac{1}{\lambda_{\max}}$$

那么学习算法是收敛的，其中 $\lambda_{\max}$ 是所有观察矢量相关阵的最大特征值。

这种递推学习算法避免了矩阵求逆的困难，但在学习过程中为了求得每一步所需的梯度，必须求出该步上所有可能的输入矢量 $\mathbf{X}(k)$ 及其相应误差 $\varepsilon(k)$ 乘积的统计平均，这需要大量的运算。下面的两种算法就是为了解决这个问题的。

三、随机逼近算法

严格递推算法之所以有大量的统计运算，就是因为它求梯度（权矢量调整的负方向）的对象是每一步中所有可能的输入矢量的均方误差。那么，我们能否对一个输入矢量的误差的平方求梯度，并用它来指导权矢量的调整呢？即按照下式调整权矢量。

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

$$\mathbf{W}(k+1)=\mathbf{W}(k)+\alpha\{-\nabla_{\mathbf{W}}\varepsilon^2(k)\}\quad k=0,1,2,\cdots$$

求 $\nabla_{\mathbf{W}}\varepsilon^2(k)$，可得：

$$\nabla_{\mathbf{W}}\varepsilon^2(k)=\left[\frac{\partial\varepsilon^2(k)}{\partial w_0},\frac{\partial\varepsilon^2(k)}{\partial w_1},\cdots,\frac{\partial\varepsilon^2(k)}{\partial w_{N-1}}\right]$$

$$=-2\varepsilon(k)\mathbf{X}(k)$$

代入上式，得

$$\mathbf{W}(k+1)=\mathbf{W}(k)+2\alpha\varepsilon(k)\mathbf{X}(k)\quad k=0,1,2,\cdots$$

这种算法避免了大量的统计运算，但却使加权矢量的调整具有了随机性，因为 $\varepsilon(k)\mathbf{X}(k)$ 是随机的，而 $E[\varepsilon(k)\mathbf{X}(k)]$ 是非随机的。这种情况下，如何保证学习算法的收敛性呢？本章的文献[1]、[2]详细分析了这种算法的收敛条件，这里只给出主要结论。采用变步幅的办法修正权矢量，公式如下：

$$\mathbf{W}(k+1)=\mathbf{W}(k)+\alpha(k)\varepsilon(k)\mathbf{X}(k)\quad k=0,1,2,\cdots\quad\cdots\cdots\text{EQ2.36}$$

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

如果上式中的步幅 $\alpha(k)$ 是时序 k 的非增函数且满足

$$\sum_{k=0}^{\infty}\alpha(k)=\infty,\quad\sum_{k=0}^{\infty}\alpha^2(k)<\infty$$

那么 $\mathbf{W}(k)$ 随着 k 的增大而趋向于 $\mathbf{W}^*$ 的概率等于1。例如 $\alpha(k)=1/k$。

四、具有一定统计特性的递推算法

假设当学习进行到第 k 步时，出现的输入矢量样本有 P 个，分别用 $\mathbf{X}_p(k)$ 来表示，其中 $p=0,1,\cdots,P-1$ 表示的是第 k 步学习可能出现的不同样本的编号。对不同的样本，神经元的理想输出和实际输出分别用 $d_p(k)$ 和 $y_p(k)$ 来表示。那么误差 $\varepsilon_p(k)=d_p(k)-y_p(k)$ 定义一个称为“误差平方和”的新参量。

$$J(k)=\frac{1}{P}\sum_p\varepsilon_p^2(k)=\frac{1}{P}\sum_p[d_p(k)-y_p(k)]^2$$

我们希望权矢量朝误差平方和减小的方向调整，即按下式调整。

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

$$\mathbf{W}(k+1)=\mathbf{W}(k)+\alpha\{-\nabla_{\mathbf{W}}J(k)\}\quad k=0,1,2,\cdots$$

而

$$\nabla_{\mathbf{W}}J(k)=\left[\frac{\partial J(k)}{\partial w_0},\frac{\partial J(k)}{\partial w_1},\cdots,\frac{\partial J(k)}{\partial w_{N-1}}\right]$$

$$=-\frac{2}{P}\sum_p\varepsilon_p(k)\mathbf{X}_p(k)$$

所以，权矢量调整公式如下：

$$\mathbf{W}(k+1)=\mathbf{W}(k)+2\alpha\frac{1}{P}\sum_p\varepsilon_p(k)\mathbf{X}_p(k)\quad k=0,1,2,\cdots\quad\cdots\cdots\text{EQ2.45}$$

以下是关于这个算法的几点讨论：

1. 如果 P 选得非常大，使得和式足以表示全体输入矢量训练样本的统计特性时，此式与Widrow的严格递推求解算法完全一致。

2. 如果选择 $P=1$，即每步学习都只选择一个训练样本，那么此式与随机

§2.3 采用线性函数时单个神经元的最小均方分类学习算法

XI'AN JIAOTONG UNIVERSITY

逼近算法完全一致。

3. 一般情况下，我们可以把 P 取为一个不太大的有限值，使学习算法具有一定的统计意义。这时，对递推算法的每一步都要输入若干个样本以后，再按EQ2.45调整一次权矢量。

4. 这种算法概念清楚，计算简单。但我们无法确定步幅应该如何选取才能保证学习结果的收敛性。但当 P 不太大时，这一算法与随机逼近算法近似，参照它的选取条件来选取步幅是比较合理的。

第一次实验：

Adaline

地址：http://jgzhang.gr.xjtu.edu.cn/

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

前面两节讨论的是单个神经元的分类问题，它可以完成两类的线性可分类问题（采用硬限幅函数）以及两类的非线性可分类问题（采用线性函数），但是单个神经元只能区分两类问题。如果要对 N 维空间中任意多种类别的输入矢量集合进行分类，就必须采用多层神经网络。

一、采用硬限幅函数的前向多层神经网络

1. 凸集合和可分凸集合

凸集合，在一个集合中任选两个点，如果在任何情况下这两个点之间的连线上的所有点仍属于该集合，则称该集合为凸集合。

可分凸集合，假设被区分的客体有 M 个类别，记为 $C_s, s=0,1,\cdots,M-1$ 。与每个类别相应的所有输入矢量构成一个集合 R_s。

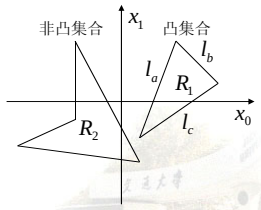


图2.9 二维空间中的凸集合和非凸集合

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

如果所有 M 个集合都是凸集合而且互不相交，我们称之为可分凸集合。

2. 采用硬限幅函数的前向二层神经网络

采用硬限幅函数的前向二层神经网络可以无误地完成对任何可分凸集合的分类。

（注意：这里所说的前向二层神经网络不包括输入信号层。）

下面，我们借助图2.9的例子来证明这个命题。

假设输入矢量分成两类，一类全在集合 R_1 以内，而另一类全在 R_1 外。我们要求，当 $\mathbf{X} \in R_1$ 时，神经网络输出 $y_0 = 1$；而当 $\mathbf{X} \notin R_1$ 时，$y_0 = 0$

利用图2.10所示的前向二层神经网络，就可以实现这种分类功能。这个神经网络共有两层，第一层包括3个神经元，而第二层只有一个神经元。其分类学习算法如下：

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

第一层： $g_i = f_h[I_i]$

$$I_i = \sum_{j=0}^{2} w_{ij} x_j \ , i = 0,1,2, (x_2 \equiv 1)$$

第二层： $y_0 = f_h[I_0]$

$$I_0 = \sum_{i=0}^{2} w_{0i}^* g_i - \theta_0^*$$

其中，x_0 和 x_1 是输入矢量 $\mathbf{X}$ 的两个分量，w_{02}、w_{12} 和 w_{22} 分别表示第一层三个神经元的阈值，θ_0^* 是第二层神经元的阈值。

在分类算法中，各个神经元的功能如下。

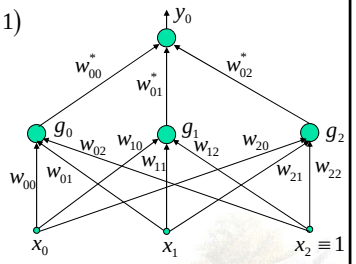


图2.10 采用硬限幅函数的前向二层神经网络示例

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

由本章第二节，我们知道采用硬限幅的单个神经元在二维平面上可以确定一条直线，这条直线把平面分成两个部分。在图2.9中，集合R_1的边界由三条直线确定，因此，我们可以用三个神经元确定这三条直线。例如：

我们让神经元 g_0 确定直线 l_a（包括两端向外的延伸线），它把平面分成了两部分，我们把包括集合 R_1 的一侧记为 R_{a1}，而把另一侧记为 R_{a2}，并规定：

$$\begin{cases} \text{当}\mathbf{X}\in R_{a1}, & g_0=1 \\ \text{当}\mathbf{X}\in R_{a2}, & g_0=0 \end{cases}$$

同理，我们可以让神经元 g_1和 g_2分别确定直线 l_b 和 l_c，且把包括 R_1的一侧分别记为 R_{b1}和 R_{c1}，而把另一侧分别记为 R_{b2}和R_{c2}，并作如下规定：

$$\begin{cases} \text{当}\mathbf{X}\in R_{b1}, & g_1=1 \\ \text{当}\mathbf{X}\in R_{b2}, & g_1=0 \end{cases} \qquad \begin{cases} \text{当}\mathbf{X}\in R_{c1}, & g_2=1 \\ \text{当}\mathbf{X}\in R_{c2}, & g_2=0 \end{cases}$$

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

显然，按照上面的规定，当$\mathbf{X}\in R_1$时，第一层三个神经元g_0、g_1和 g_2的输出必然是$g_0=g_1=g_2=1$；当 $\mathbf{X}\notin R_1$ 时，这三个神经元中至少有一个输出为零。输出层神经元就可以根据这个特点很容易地完成分类。

根据第二层分类算法，如果令 $w_{00}^*=w_{01}^*=w_{02}^*=1$以及 $\theta_0^*=2.5$，那么很容易证明下式成立。

$$\begin{cases} \text{当}\mathbf{X}\in R_1, & y_0=1 \\ \text{当}\mathbf{X}\notin R_1, & y_0=0 \end{cases}$$

两点讨论：

对任意多个非交凸集合（如M个），可以在第二层安排 M个输出神经元，如果输入矢量属于第m 类，就让第 m个神经元的输出为1，而其它神经元输出0。当然，第一层也要相应地增加神经元。

虽然上面是针对输入矢量为二维的情况进行讨论的，但可以毫无困难地推广到任意多维的情况。

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

3. 采用硬限幅函数的前向三层神经网络

采用硬限幅函数的前向三层神经网络可以无误地完成对任何非交集合的分类。

我们以图2.11中的非凸集合R为例，来证明采用硬限幅函数的前向三层神经网络可对集合的内外进行分类。设神经网络的输出为y_0，我们要求：

$$\begin{cases} \text{当}\mathbf{X}\in R, & y_0=1 \\ \text{当}\mathbf{X}\notin R, & y_0=0 \end{cases}$$

采用图2.12所示的前向三层神经网络可以完成这个分类任务。**基本思路是：把非凸集合分割为凸集合**，如图中的 R_1 和 R_2。

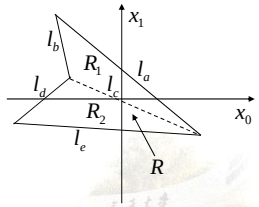


图2.11 对二维非凸集合分类的示例

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

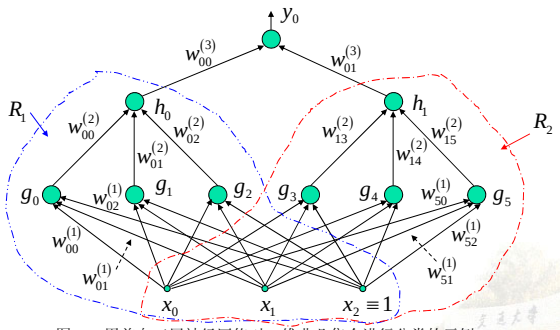


图2.12 用前向三层神经网络对二维非凸集合进行分类的示例

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

例如，我们让图中蓝色虚线框内的神经元完成对集合R_1的分类，使得：

$$\begin{cases} \text{当}\mathbf{X}\in R_1, & h_0=1 \\ \text{当}\mathbf{X}\notin R_1, & h_0=0 \end{cases}$$

而同时让红色虚线框内的神经元完成对集合R_2的分类，使得：

$$\begin{cases} \text{当}\mathbf{X}\in R_2, & h_1=1 \\ \text{当}\mathbf{X}\notin R_2, & h_1=0 \end{cases}$$

显然，只要h_0和h_1有一个输出为1，输入矢量就属于集合R。那么，我们再增加一个神经元来完成一个或运算就可以了（第一章）。

分类算法如下：

第一层：

$$g_i = f_h[I_i],\ I_i=\sum_{j=0}^{2} w_{ij}^{(1)}x_j,\ i=0,1,\cdots,5,\ (x_2\equiv 1)$$

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

其中，$w_{i2}^{(1)}$表示第一层中第i个神经元的阈值。本层中的6个神经元分别完成对直线l_a、l_b、l_c、l_d和l_e（包括其两端的延长线）两侧的集合进行分类。

第二层：

$$h_n = f_h[I_n],\ I_n=\sum_{i=0}^{5} w_{ni}^{(2)}g_i-\theta_n^{(2)},\ n=0,1$$

其中，$\theta_n^{(2)}$表示第二层中第n个神经元的阈值。

第三层：

$$y_s = f_h[I_s],\ I_s=\sum_{n=0}^{1} w_{sn}^{(3)}h_n-\theta_s^{(3)},\ s=0$$

本层只有一个神经元，$\theta_0^{(3)}$是它的阈值。如果我们令

$$w_{00}^{(3)}=w_{01}^{(3)}=1 \ \text{及}\ \theta_0^{(3)}=0.5$$

那么这个神经元就可以完成所要求的或运算。即若h_0或h_1中有一个为1，那么$y_0=1$；若两者全为0，那么$y_0=0$。这就完成了所要求的分类功能。

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

讨论：

A. 以上的结论可以推广到具有任意形状的非凸集合；也可以推广到输入矢量为任意多维的情况。

B. 如果需要对若干个互不相交的非凸集合进行分类，只需要在此神经网络的相应层次安排相应的神经元就可以了。

C. 神经元g_2和g_3可以共用一个神经元（加一个反相器），节省神经元。

4. 对采用硬限幅函数的前向多层神经网络分类功能的讨论

A. 功能强大，对任意形状的非交输入矢量集合都可以进行无误地分类。但有两个严重的缺点。

B. 它不能对相交的集合进行分类。

C. 至今尚未找到一种有效的学习算法，这个缺点是致命的，因为对实际应用而言，没有有效学习算法的神经网络是毫无价值的。

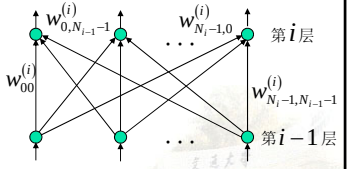
§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

二、采用线性函数的前向多层神经网络

由上一节，我们知道，采用线性函数的单个神经元可以完成对相交集合或部分非线性可分集合的分类，而且有各种有效的学习算法。那么，我们能否对前向多层神经网络使用线性函数来解决采用硬限幅函数的前向多层神经网络所存在的问题呢？结论是否定的。

设前向网络的第i层共有N_i个神经元，第$i-1$层共有N_{i-1}个神经元，如右图所示。其中每层都有一个神经元的输出恒为1，它对下一层的传输系数即为该神经元的阈值。两层的输出分别用$\mathbf{X}_{i-1}$和$\mathbf{X}_i$来表示。两者间的关系为：



$$\mathbf{X}_i=\mathbf{X}_{i-1}\mathbf{W}_i$$

§2.4 采用硬限幅函数、线性函数的前向多层神经网络

XI'AN JIAOTONG UNIVERSITY

其中，$\mathbf{X}_i=\left[x_0^{(i)}, x_1^{(i)}, \cdots, x_{N_i-1}^{(i)}\right]$

$\mathbf{X}_{i-1}=\left[x_0^{(i-1)}, x_1^{(i-1)}, \cdots, x_{N_{i-1}-1}^{(i-1)}\right]$

$\mathbf{W}_i$ 是一个 $N_{i-1}\times N_i$ 维的权系数矩阵。

如果此神经网络共包括L层，其中第一层的输入矢量为 $\mathbf{X}_0$，第 L层的输出矢量为 $\mathbf{Y}$，那么输出与输入之间的关系为：

$$\mathbf{Y}=\mathbf{X}_L=\mathbf{X}_0\mathbf{W}_1\mathbf{W}_2\cdots\mathbf{W}_L=\mathbf{X}_0\mathbf{W}$$

其中，$\mathbf{W}=\prod_{i=1}^{L}\mathbf{W}_i$

可以看到，无论 L 值取多大，即无论神经网络有多少层，其输出和输入之间的关系总可以等效为一个单层神经网络。其分类功能和 N_L个相互独立的神经元并没有区别，因此，采用线性函数的前向多层神经网络不能解决任何复杂的问题。

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

前面几节讲了采用硬限幅函数或线性函数时单个神经元和前向多层神经网络的分类功能。其中，单神经元只能解决两类的分类问题（线性可分类和部分非线性可分类）；而采用硬限幅函数的前向多层神经网络虽然功能强大，但没有有效的学习算法；采用线性函数的前向多层神经网络等效为一个单层的神经网络，因此功能很弱。因此，我们转而研究采用其它变换函数的前向多层神经网络，使网络既有强的分类功能，又有有效的学习算法。这样的变换函数有多种选择，我们下面讨论采用S形函数的前向多层神经网络。

采用硬限幅函数的前向多层神经网络之所以没有有效的学习算法，是因为硬限幅函数是非可微函数。**S形函数既具有完成分类所需的非线性特性，又有实现LMS算法所需的可微特性**。此外，S形函数的特性接近人脑的输入-输出特性。

一、采用S形函数的前向多层神经网络模型

图2.13以三层网络为例给出了采用S形函数的前向多层神经网络结构图。

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

网络输入为 N维矢量：

$\mathbf{X}=[x_0\ x_1\cdots x_{N-1}]$

网络输出为 M 维矢量：

$\mathbf{Y}=[y_0\ y_1\cdots y_{M-1}]$

网络第一层共有 J 个神经元，相应的输出记为：

$g_0, g_1, \cdots, g_{J-1}$

第二层共有 K个神经元，相应的输出记为：

$h_0, h_1, \cdots, h_{K-1}$

为了便于分析，归纳一个通用层模型，如图2.14所示。

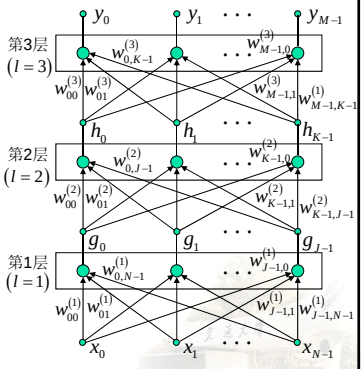


图2.13 采用S形函数的前向三层神经网络

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

例如，对第一层有：

$l=1, L=N, Q=J$

$o_{pj}^{(l-1)}=o_{pj}^{(0)}=x_j,\ \ j=0,1,\cdots,N-1$

$o_{pi}^{(l)}=g_i,\ \ i=0,1,\cdots,J-1$

其中的下标 p表示训练样本编号。

通用层的输出-输入关系如下：

$$\begin{cases} I_{pi}^{(l)}=\sum_{j=0}^{L-1}w_{ij}^{(l)}o_{pj}^{(l-1)}-\theta_i^{(l)} \\ o_{pi}^{(l)}=f_s\left[I_{pi}^{(l)}\right] \end{cases}$$

$$i=0,1,\cdots,Q-1$$

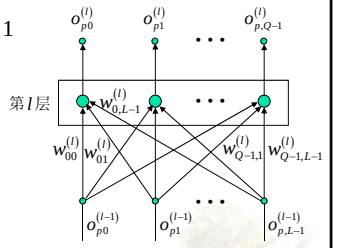


图2.14 前向多层神经网络的通用层模型

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

如果设 $o_{pL}^{(l-1)}=1, -\theta_i^{(l)}=w_{iL}^{(l)}$，那么净输入 $I_{pi}^{(l)}$ 可以写为：

$$I_{pi}^{(l)}=\sum_{j=0}^{L} w_{ij}^{(l)} o_{pj}^{(l-1)}$$

二、采用S形函数的前向多层神经网络的LMS算法（BP算法）

设对于训练样本 p，网络总输出矢量 $\mathbf{Y}$ 的各分量的理想值为 d_{pi}，而实际值为 y_{pi}，两者之间的误差为 $\varepsilon_{pi}=\left(d_{pi}-y_{pi}\right)$，$i=0,1,\cdots,M-1$。

输出矢量各分量误差的平方和为：$E_p=\sum_{i=0}^{M-1}\varepsilon_{pi}^2$

我们希望通过递推算法改变网络中的各个加权系数 $w_{ij}^{(l)}$，使 E_p 尽可能减小。

$$w_{ij}^{(l)}(k+1)=w_{ij}^{(l)}(k)+\Delta_p w_{ij}^{(l)}$$

$$\Delta_p w_{ij}^{(l)}=-\alpha\frac{\partial E_p}{\partial w_{ij}^{(l)}};\quad \begin{matrix} i=0,1,\cdots,Q-1 \\ j=0,1,\cdots,L \end{matrix} \quad l=1,2,3 \qquad \cdots\cdots \text{EQ2.54}$$

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

其中，α 为步长。注意：在此计算调整量时我们只使用了一个样本，**这样得到的算法是一种随机梯度算法**。

现在的目的就是求上式右侧的偏微分，我们先将它表示为下列的复合微分：

$$\frac{\partial E_p}{\partial w_{ij}^{(l)}}=\frac{\partial E_p}{\partial I_{pi}^{(l)}}\cdot\frac{\partial I_{pi}^{(l)}}{\partial w_{ij}^{(l)}} \qquad \cdots\cdots\cdots\cdots \text{EQ2.55}$$

其中第二项很简单，由EQ2.53，显然有

$$\frac{\partial I_{pi}^{(l)}}{\partial w_{ij}^{(l)}}=\frac{\partial\left\{\sum_{j=0}^{L} w_{ij}^{(l)} o_{pj}^{(l-1)}\right\}}{\partial w_{ij}^{(l)}}=o_{pj}^{(l-1)} \qquad \cdots\cdots\cdots\cdots \text{EQ2.57}$$

把第一项定义为一个新变量：

$$-\delta_{pi}^{(l)}=\frac{\partial E_p}{\partial I_{pi}^{(l)}} \qquad \cdots\cdots\cdots\cdots\cdots\cdots \text{EQ2.56}$$

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

综合EQ2.54~EQ2.57，有：

$$\Delta_p w_{ij}^{(l)}=\alpha\delta_{pi}^{(l)} o_{pj}^{(l-1)};\quad \begin{matrix} i=0,1,\cdots,Q-1 \\ j=0,1,\cdots,L \end{matrix} \quad l=1,2,3 \qquad \cdots\cdots \text{EQ2.58}$$

其中，$o_{pj}^{(l-1)}$ 可以从神经网络中得到；因此，只要求出 $\delta_{pi}^{(l)}$ 就可以了。再次使用复合微分可得。

$$\delta_{pi}^{(l)}=-\frac{\partial E_p}{\partial I_{pi}^{(l)}}=-\frac{\partial E_p}{\partial o_{pi}^{(l)}}\cdot\frac{\partial o_{pi}^{(l)}}{\partial I_{pi}^{(l)}} \qquad \cdots\cdots\cdots\cdots\cdots\cdots \text{EQ2.59}$$

上式右侧的第二项相当于对S形函数进行求导，结果如下：

$$\frac{\partial o_{pi}^{(l)}}{\partial I_{pi}^{(l)}}=f_s'\left[I_{pi}^{(l)}\right]=o_{pi}^{(l)}\left(1-o_{pi}^{(l)}\right) \qquad \cdots\cdots\cdots\cdots\cdots\cdots \text{EQ2.60}$$

$$f_s'(u)=\left(\frac{1}{1+e^{-u}}\right)'=\frac{e^{-u}}{\left(1+e^{-u}\right)^2}=f_s(u)\left[1-f_s(u)\right]$$

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

对于EQ2.59式右侧的第一项 $\frac{\partial E_p}{\partial o_{pi}^{(l)}}$，分为以下两种情况进行讨论。

1. 对输出层，

$$o_{pi}^{(l)}=y_{pi},\qquad E_p=\sum_{i=0}^{M-1}\left(d_{pi}-y_{pi}\right)^2$$

所以

$$\frac{\partial E_p}{\partial o_{pi}^{(l)}}=-2\left(d_{pi}-o_{pi}^{(l)}\right),\ i=0,1,\cdots,M-1 \qquad \cdots\cdots\cdots\cdots \text{EQ2.62}$$

d_{pi} 是伴随训练样本给出的，而 $o_{pi}^{(l)}$ 即为网络的输出 y_{pi}，所以此式可以立即计算，综合EQ2.59，EQ2.60和EQ2.62，有：

$$\delta_{pi}^{(l)}=2\left(d_{pi}-o_{pi}^{(l)}\right)o_{pi}^{(l)}\left(1-o_{pi}^{(l)}\right),\ i=0,1,\cdots,M-1 \qquad \cdots\cdots\cdots \text{EQ2.63}$$

代入EQ2.58，得：

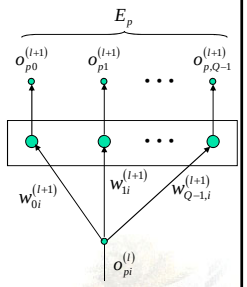
$$\Delta_p w_{ij}^{(l)}=\alpha\cdot\underbrace{2\left(d_{pi}-o_{pi}^{(l)}\right)o_{pi}^{(l)}\left(1-o_{pi}^{(l)}\right)}_{\delta_{pi}^{(l)}}\cdot o_{pj}^{(l-1)}$$

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

2. 对隐含层

由于 $o_{pi}^{(l)}$是第 $l+1$ 层所有神经元的输入，因此它的误差会引起 $l+1$ 层净输入的误差，从而使神经网络的输出存在误差。参见右图。



对复合全微分的说明

因此，对隐含层，我们可以用复合全微分来计算EQ2.59右侧的第一项。

$$\frac{\partial E_p}{\partial o_{pi}^{(l)}}=\sum_{k=0}^{Q-1}\left(\frac{\partial E_p}{\partial I_{pk}^{(l+1)}}\cdot\frac{\partial I_{pk}^{(l+1)}}{\partial o_{pi}^{(l)}}\right),\quad \begin{array}{l}\text{当}l=1,Q=K\\ \text{当}l=2,Q=M\end{array}$$

由EQ2.53，可得：

$$I_{pk}^{(l+1)}=\sum_{i=0}^{L}w_{ki}^{(l+1)}o_{pi}^{(l)},\quad \begin{array}{l}\text{当}l=1,\ L=J,\ k=0,1,\cdots K-1\\ \text{当}l=2,\ L=K,\ k=0,1,\cdots M-1\end{array}$$

More...

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

所以，复合全微分等式中的第二项为：

$$\frac{\partial I_{pk}^{(l+1)}}{\partial o_{pi}^{(l)}}=w_{ki}^{(l+1)}\qquad\cdots\cdots\cdots\cdots\cdots\cdots\text{EQ2.65}$$

根据EQ2.59，其中的第一项可以表示为：

$$\frac{\partial E_p}{\partial I_{pk}^{(l+1)}}=-\delta_{pk}^{(l+1)}\qquad\cdots\cdots\cdots\cdots\cdots\cdots\text{EQ2.66}$$

将上两式代入复合全微分等式，有：

$$\frac{\partial E_p}{\partial o_{pi}^{(l)}}=-\sum_{k=0}^{Q-1}\left(\delta_{pk}^{(l+1)}\cdot w_{ki}^{(l+1)}\right),\quad \begin{array}{l}\text{当}l=1,Q=K,i=0,1,\cdots J-1\\ \text{当}l=2,Q=M,i=0,1,\cdots K-1\end{array}\qquad\cdots\cdots\text{EQ2.67}$$

因为$\delta_{pk}^{(l+1)}$在求 $l+1$层的调整量时已经求出了，而 $w_{ki}^{(l+1)}$是已知的权重系数。

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

同样，当我们求出$\delta_{pk}^{(l)}$ 以后，就可以求出 $\delta_{pk}^{(l-1)}$，所以对隐含层我们有：

$$\delta_{pi}^{(l)}=\left\{\sum_{k=0}^{Q-1}\delta_{pk}^{(l+1)}w_{ki}^{(l+1)}\right\}o_{pi}^{(l)}\left(1-o_{pi}^{(l)}\right),\quad \begin{array}{l}\text{当}l=1,Q=K,i=0,1,\cdots,J-1\\ \text{当}l=2,Q=M,i=0,1,\cdots,K-1\end{array}$$

所以隐含层的权调整公式为：

$$\Delta_p w_{ij}^{(l)}=\alpha\cdot\underbrace{\left\{\sum_{k=0}^{Q-1}\delta_{pk}^{(l+1)}w_{ki}^{(l+1)}\right\}o_{pi}^{(l)}\left(1-o_{pi}^{(l)}\right)}_{\delta_{pi}^{(l)}}\cdot o_{pj}^{(l-1)}$$

我们把推导出的算法，总结如下：

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

权系数调整的递推公式如下：

$$w_{ij}^{(l)}(k+1)=w_{ij}^{(l)}(k)+\Delta_p w_{ij}^{(l)}$$

对输出层：

$$\Delta_p w_{ij}^{(l)}=\alpha\cdot\underbrace{2\left(d_{pi}-o_{pi}^{(l)}\right)o_{pi}^{(l)}\left(1-o_{pi}^{(l)}\right)}_{\delta_{pi}^{(l)}}\cdot o_{pj}^{(l-1)}$$

对隐含层：

$$\Delta_p w_{ij}^{(l)}=\alpha\cdot\underbrace{\left\{\sum_{k=0}^{Q-1}\delta_{pk}^{(l+1)}w_{ki}^{(l+1)}\right\}o_{pi}^{(l)}\left(1-o_{pi}^{(l)}\right)}_{\delta_{pi}^{(l)}}\cdot o_{pj}^{(l-1)}$$

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

对图2.13所示的前向三层神经网络，我们有：

$l=3$，输出层

$$\Delta_p w_{ij}^{(3)} = \alpha \cdot \underbrace{2\left(d_{pi}-o_{pi}^{(3)}\right)o_{pi}^{(3)}\left(1-o_{pi}^{(3)}\right)}_{\delta_{pi}^{(3)}} \cdot o_{pj}^{(2)}$$

$$i=0,1,\cdots,M-1; \quad j=0,1,\cdots,K$$

$l=2$：

$$\Delta_p w_{ij}^{(2)} = \alpha \cdot \underbrace{\left\{\sum_{k=0}^{M-1}\delta_{pk}^{(3)} w_{ki}^{(3)}\right\} o_{pi}^{(2)}\left(1-o_{pi}^{(2)}\right)}_{\delta_{pi}^{(2)}} \cdot o_{pj}^{(1)}$$

$$i=0,1,\cdots,K-1; \quad j=0,1,\cdots,J$$

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

$l=1$：

$$\Delta_p w_{ij}^{(1)} = \alpha \cdot \underbrace{\left\{\sum_{k=0}^{K-1}\delta_{pk}^{(2)} w_{ki}^{(2)}\right\} o_{pi}^{(1)}\left(1-o_{pi}^{(1)}\right)}_{\delta_{pi}^{(1)}} \cdot o_{pj}^{(0)}$$

$$i=0,1,\cdots,J-1; \quad j=0,1,\cdots,N$$

注意：

$$o_{pi}^{(3)} = y_{pi}, \ i=0,1,\cdots,M-1$$

$$o_{pj}^{(0)} = x_{pj}, \ j=0,1,\cdots,N$$

我们可以看到，这一算法的计算过程是先计算输出层各项的“误差分量”，然后再利用它求出倒数第二层的“等效误差分量”，以此类推，直到计算出第一层的“等效误差分量”。计算过程中，误差从输出逆推到输入，所以称为“逆推”学习算法，或称为BP（Back Propagation）算法。

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

以上是对采用S形函数的前向多层神经网络进行讨论的，但可以很容易地推广到采用其它可微非线性函数的情况。

三、应该注意的几个问题

实际应用中，在如何利用BP算法以及如何设置各层神经元的数量等方面还有许多问题需要讨论，下面是要注意的几个问题。

1. 由于采用的是S形函数，因而输出层各神经元的理想输出值不能达到1或0，而只能接近于1或0，所以，训练时理想输出值通常设为0.9或0.1。

2. 步幅α的选择是至关重要的。一种方法是采用变步幅的方案；另一种方法是采用所谓有“惯性效应”的调整算法，即在计算t_k步的调整量时要考虑到第t_{k-1}步的调整量。此算法可用下式描述：

$$\Delta_p w_{ij}^{(l)}(t_k) = \alpha\delta_{pi}^{(l)}(t_k)o_{pj}^{(l-1)}(t_k) + \eta \cdot \Delta_p w_{ij}^{(l)}(t_{k-1})$$

其中，η称为惯性系数，它的值越大系数调整的惯性越大。

§2.5 采用S形函数的前向多层神经网络及其反向学习算法

XI'AN JIAOTONG UNIVERSITY

3. 权重系数$w_{ij}^{(l)}$初值的选择。初值可以随机设置，但不能相等。如果初始权值相等，那么隐含层权重系数的调整量始终相同。

4. 局部极小点问题。对前面讨论的采用线性函数的单神经元LMS学习算法，其均方误差只有一个最小点，各种学习算法都可收敛到这个点。而本节讨论的前向多层神经网络，其误差相对于各权重系数的变化具有非常复杂的“曲面”，具有很多极小点，其中只有一个是“全局最小点”，其它都是“局部最小点”。显然，只有全局最小点才是网络的最佳解。如果初值设置不当就有可能使权重系数在学习结束时收敛到某个局部极小值上。

5. 以上的讨论都是针对分类问题的，但也可以用于解决联想问题。

6. 以上给出的是一种随机梯度算法。与LMS递推算法类似，可以在一次调整中考虑多个样本。

7. 可以改变陡度因子。

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

一、奇偶校验网络

如图2.15所示，输入矢量

$$\mathbf{X}=[x_0\ x_1\cdots x_{N-1}]$$

其中 N 为偶数，$x_j\in\{0,1\}$ 。要求

$$y_0=\begin{cases}0 & \text{若}\mathbf{X}\text{中包括偶数个1}\\ 1 & \text{若}\mathbf{X}\text{中包括奇数个1}\end{cases}$$

不难验证，当网络具有如图所示的参数时确能完成奇偶校验的功能。图中圆圈内的数值是该神经元的阈值，实线表示加权系数为1，虚线表示加权系数为-1。

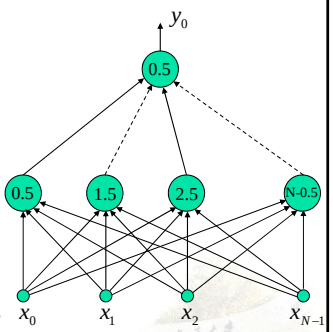


图2.15 用于奇偶校验的前向二层神经网络

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

当输入中有 n 个1时，第一层的前 n 个神经元输出接近1，其它的神经元输出接近0；当 n 为偶数时，第二层神经元的净输入接近-0.5，网络输出接近0；当 n 为奇数时，第二层神经元的净输入接近0.5，网络输出接近1。

下面的实验说明，这些参数可以通过BP学习算法自动地得到。

以 $N=4$ 为例，输入 $\mathbf{X}$ 可能出现16种模式，将其及相应的理想输出随机送入神经网络；实验采用带惯性效应的BP学习算法，学习步幅选为 $\alpha=0.5$ ，惯性系数选为 $\eta=0.9$ 。经过2825次学习后，得到图2.15中的各参数。

改变初值以及 $\mathbf{X}$ 的送入顺序，可以得到异于如图所示的解，但也可以完成奇偶校验的功能。

当$N=2$时，奇偶校验电路退化为异或电路。

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

二、T-C辨识问题

输入信号为两类图形，T图形和C图形，每类图形各有4种旋转角度，它们均由5个相邻的取值为1的分量构成。实验中使用的是一个9×11的输入信号阵列，这些图形可以出现在阵列的任何位置上。我们要求当输入T图形时，神经网络输出 $y_0=1$，而当输入C图形时，输出 $y_0=0$ 。我们可以用一个前向两层神经网络（图2.19）完成此辨识任务。

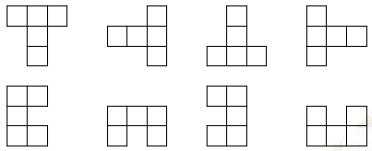


图2.18 T-C辨识问题中的八个图形

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

为了保持识别的位移不变性，规定每个小方阵（3×3）到相应隐含层神经元的各对应权系数要始终保持一致。这要求它们的初始值相同，并且对它们的调整也要一致。对第一层（ $l=1$ ），每个小方阵有9个权系数，因此，共有9种权值；对第二层（ $l=2$ ），所有隐含层的神经元都具有同等的地位，它们到输出层的权系数也应该相等。因此，整个网络共有9+1种权系数。

为了保证调整后仍然保持9+1种值，需要对对应的权系数进行相同的调整，因此我们要根据计算出的调整量计算调整总量，学习步骤如下：

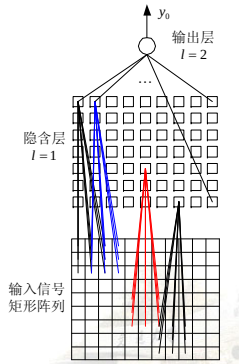


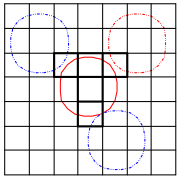
图2.19 解决T-C辨识问题的前向网络

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

① 用BP算法算出所有权系数的调整量；② 算出隐含层权系数的9个调整总量和输出层的一个调整总量；③ 按以上9+1种调整总量对相应的权系数进行相同的调整。

训练时，随机抽取8种图形并将它随机置于接收场的任何一个位置，依次送入神经网络进行学习。大约经过5000~10000次学习就可以得到正确的学习结果。



在讨论学习结果之前，要注意的是神经网络并非只是用图形所在的那个小方阵来进行辨识的；因为，当输入一个图形时，输入场中有多个小方阵的输入会发生变化。

下面讨论四种不同的学习结果。

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

A

−1	−1	−1
−1	+2	−1
−1	−1	−1

B

1	+1	1
−1	+1	−1
−1	+1	−1

图2.20 T-C辨识问题的学习结果

隐含层神经元的阈值 $\theta = 0.5$，隐含层到输出层神经元的权系数均为1，输出神经元的阈值 $\theta = 0.5$。

当输入T时，隐含层总有一个神经元的输出为1；而如果输入C，则没有一个隐含层神经元的输出会大于0。

对于B的情况：

隐含层神经元的阈值 $\theta = 1.5$，隐含层到输出层神经元的权系数均为1，输出神经元的阈值 $\theta = 0.5$。

当输入T时，隐含层总有一个神经元的输出为1（总输入等于2）；而如果输入C，则没有一个隐含层神经元的总输入会大于1。

这两种情况都属于“T图形检测器”

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

隐含层神经元的阈值 $\theta = 0.5$，隐含层到输出层神经元的权系数都为1，输出层神经元的阈值 $\theta = 4$。

如果输入T，隐含层会有5个神经元的输出为1；而如果输入C，隐含层只有3个神经元的输出等于1。

对于D的情况：

隐含层神经元的阈值 $\theta = -1$，因此，当没有图形输入时，隐含层神经元的输出都为1。当输入T时，隐含层中有21个神经元输出为0，其它输出1；当输入C时，隐含层中有20个神经元输出为0，其它输出1。

C

−1	−1	+1			−1	−1	+1	
−1	+2	−1			−1	+2	−1	
+1	−1	−1			+1	−1	−1	

图2.20 T-C辨识问题的学习结果

D

−2	−2	−2
−2	−2	−2
−2	−2	−2

More…

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

三、手写体字母及数字的识别

我们通过对工整规范的手写体进行预处理编码得到便于识别的输入矢量。一种预处理方案是用光电器件（LED）构成“影码符号读出器”，如图2.21所示。

对手写体字母或数字进行规格化处理，然后把输入字符上的每一点向其邻近的垂直、水平和倾斜的三根LED杆上投影，每根LED杆上的投影面积决定了该杆的输出值。如图2.21c中，S投影后得：

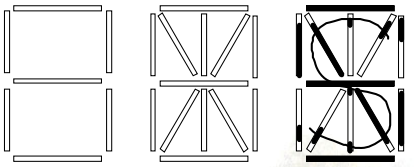


图2.21 (a) 由7段光电器件构成的影码读出器；(b) 由13段光电器件构成的影码读出器；(c) 工作原理。

§ 2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

$$\mathbf{X} = [50\ 46\ 42\ 4\ 14\ 14\ 50\ 24\ 14\ 6\ 43\ 42\ 50]$$

采用前向二层神经网络可以完成这个任务，网络结构如下。

输出层（第二层），共有26个神经元，每个神经元对应一个字母，当输入该字母时，此神经元输出为1，其它神经元输出为0。

隐含层（第一层），**神经元的个数可变**。

在实验中，让用户在一周内将每个字母写8遍（扩大样本差异），共得208个样本，其中的一半（104）作为训练样本，另一半作为测试样本。

在学习时，每一遍都将104个训练样本随机地依次送入神经网络，学习速度按照遍数来计算。

学习算法采用有惯性系数的BP算法，$\eta = 0.9$，步幅 α 在实验中是可变的。

实验中分别研究了学习速度和正确识别率与神经元个数及步幅 α 的关系。

§ 2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

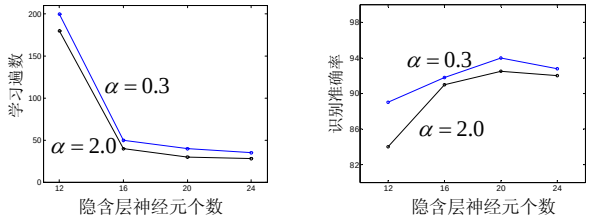


图2.22 字母识别的学习速度及识别准确率与隐含层神经元个数及步幅的关系

左图给出了学习遍数与隐含层神经元个数及步幅的关系，可以看到步幅的影响不大。右图给出了识别准确率与隐含层神经元个数及步幅的关系，可以看到，步幅较小时，识别率较高，当隐含层神经元个数为20时，识别率最高，约为94%。如果加上上下文构词法的约束，字母的识别准确率可以达到99.7%。

§ 2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

进行手写数字识别时，要求用户将每个数字写10遍，共得到100个样本，其中一半用于学习，一半用于测试。

神经网络的结构及学习算法与字母识别相同，输出层共10个神经元，隐含层神经元个数可变。实验中选择 $\alpha = 2.0,\ \eta = 0.9$ 。

学习速度、识别准确率与隐含层神经元的个数之间的关系如图2.23所示。可以看到，隐含层有6个神经元时，识别准确率最高，可达98%左右。

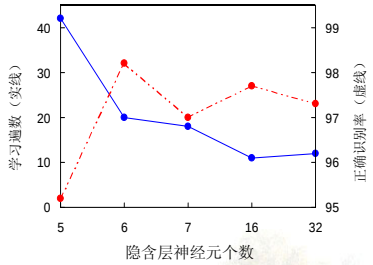


图2.23 数字识别的学习速度及正确识别率与隐含层神经元个数的关系

§ 2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

四、语音识别

用神经网络识别三个最难以区分的浊塞辅音：[b]、[d]、[g]。

将语音送往神经网络前需要进行预处理。预处理过程如下：首先用12kHz的采样率对150ms的信号进行采样，共得1800点的数据，然后进行分帧，每60点（5ms的数据）分为一帧。第二步是对每帧数据加汉明窗后进行256点的FFT，将得到的功率谱按临界带非均匀地划分、归并形成16维矢量。最后将两个相邻的16维临界带功率谱矢量取平均，得到间隔为10ms的16维平均临界带功率谱，将其送入神经网络。

语音以孤立音节读出，如[da]、[bi]、[gu]等。对任何一个输入的语音音节，首先要找到它的浊音段的开始时刻，然后以此为中心在其前后各选出7个16维的临界带平均功率谱矢量，再加上一个中心的16维矢量，从而形成一个 16×15 维的输入信号阵列，送入神经网络。

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

实验使用的是如图2-24所示的前向三层神经网络。

输入层共有240个输入神经元，第一层共安排了104个（8×13）神经元，输入信号阵列中的第一、二、三个矢量与第一层阵列中最左侧的8个神经元相互连接，依次类推，第一层共有 $48\times8\times13=4992$ 个权系数。第二层共有27个神经元，第一层的1到5列共40个神经元与第二层最左侧的3个神经元连接并依次类推，第2层共有$120\times9=1080$个权系数。第三层只有三个神经元分别和三个辅音相对应，第二层的所有神经元和第三层的每个神经元都要连接，共有81个权系数。

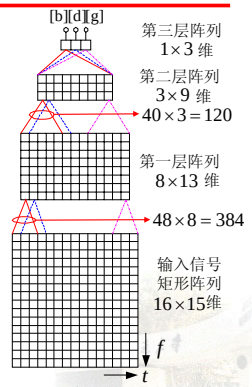


图2-24 用于语音识别的前向三层神经网络

§2.6 前向多层神经网络的应用

XI'AN JIAOTONG UNIVERSITY

模拟实验采用的是带有惯性系数的BP算法。其第一层和第二层的权系数调整和T-C辨识中类似，都要计算调整总量以保证对应系数的调整步调一致。

实验中共用了800个训练样本，学习参数为 $\alpha=0.002,\ \eta=0.1$。学习遍数（系数每调整一次算一遍）在20000～50000之间。在具有4个处理器的Alliant超级计算机上运行了4天才完成此学习任务，这个代价在当时是相当惊人的。当然学习后工作时的识别速度是极快的。

实验的结果是令人满意的。对3个实验者的统计表明，[b]、[d]、[g]三个音素的正确识别率分别达到了98.8%，99.1%和97.5%。在完全相同的条件下，用隐含马尔可夫模型（HMM）算法进行识别时，这三个音素的正确识别率为92.9%，97.2%和90.9%。前者较后者平均高出4.8个百分点。

§2.7 本章小结

XI'AN JIAOTONG UNIVERSITY

BP算法的出现使前向多层神经网络的研究取得了重大突破，有力地促进了人工神经网络研究高潮的兴起。下面简单作一小结。

1. 前向多层神经网络及其算法的研究与自适应信号处理之间有着极为密切的渊源关系。自适应信号处理的研究成果可以用到神经网络的研究中，而前向神经网络的研究也促进了自适应信号处理的发展。

2. 采用硬限幅函数和线性函数的单个神经元具有一定的分类功能，且有学习算法。

3. 采用硬线幅函数的前向多层神经网络具有很强的分类功能，但没有学习算法；采用线性函数的前向多层神经网络等效为一个单层的神经网络。

4. 对前向多层神经网络及BP算法有一些批评：

① 没有证据表明，人脑神经系统是按BP算法进行学习的。

② BP算法不能保证学习结果达到均方误差的全局最小点。

§2.7 本章小结

XI'AN JIAOTONG UNIVERSITY

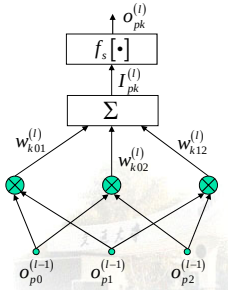
③ 学习样本要充分多，否则很难“推广”。

④ BP算法运算量太大。

5. 对前向多层神经网络可以采用一些新的结构。如采用 $\Sigma-\Pi$ 模型构成神经元。

如右图是这种模型的一个简化示意图。输出层神经元的输入输出关系为：

$$\begin{cases} o_{pk}^{(l)} = f_s\left[I_{pk}^{(l)}\right] \\ I_{pk}^{(l)} = \sum_{i,j} w_{kij}^{l} o_{pi}^{(l-1)} o_{pj}^{(l-1)} \end{cases}$$



其BP算法如下：

$$\Delta_p w_{kij}^{(l)} = \delta_{pk}^{(l)} o_{pi}^{(l-1)} o_{pj}^{(l-1)}$$

$$\delta_{pk}^{(l)} = 2\left(d_{pk} - o_{pk}^{(l)}\right) f_s'\left[I_{pk}^{(l)}\right], l=3$$

$$\delta_{pk}^{(l)} = \left\{\sum_{j\neq i}\sum_{k} \delta_{pk}^{(l+1)} w_{kij}^{(l+1)} o_{pj}^{(l)}\right\} f'\left[I_{pi}^{(l)}\right], l=1,2$$

附录A. EQ2.15到EQ2.16的推导（1）

XI'AN JIAOTONG UNIVERSITY

梯度定义：设 $\mathbf{x}=(x_1,x_2,\ldots,x_N)$，$y=f(\mathbf{x})$ 则

$$\nabla_{\mathbf{x}} y=\nabla_{\mathbf{x}} f(\mathbf{x})=\left(\frac{\partial f(\mathbf{x})}{\partial x_1},\frac{\partial f(\mathbf{x})}{\partial x_2},\ldots,\frac{\partial f(\mathbf{x})}{\partial x_N}\right)$$

由此，对第三项，有 $\mathbf{P}\mathbf{w}^T=\sum_{i=0}^{N-1}p_i w_i$

$$\nabla_{\mathbf{w}}(\mathbf{P}\mathbf{w}^T)=\left(\frac{\partial\sum_{i=0}^{N-1}p_i w_i}{\partial w_0},\frac{\partial\sum_{i=0}^{N-1}p_i w_i}{\partial w_1},\ldots,\frac{\partial\sum_{i=0}^{N-1}p_i w_i}{\partial w_{N-1}}\right)$$
$$=(p_0,p_1,\ldots,p_{N-1})=\mathbf{P}$$

附录A. EQ2.15到EQ2.16的推导（2）

XI'AN JIAOTONG UNIVERSITY

对第二项，有 $\mathbf{w}\mathbf{R}\mathbf{w}^T=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}w_i r_{ij} w_j$，以对 w_0 的求导为例：

$$\begin{cases} a.\ \ i=0,j\neq 0\text{时}\ \ \sum_{j=1}^{N-1}r_{0j}w_j \\ b.\ \ i\neq 0,j=0\text{时}\ \ \sum_{i=1}^{N-1}r_{i0}w_i \\ c.\ \ i=0,j=0\text{时}\ \ 2w_0 r_{00} \\ d.\ \ i\neq 0,j\neq 0\text{时}\ \ 0 \end{cases}$$

注意到 $\mathbf{R}$ 的对称性，若 $i=j$，则 $r_{i0}=r_{0j}$。所以，

$$a+b+c+d=2\sum_{i=0}^{N-1}r_{i0}w_i=2\mathbf{w}\mathbf{R}(:,0)$$

$$\therefore\ \nabla_{\mathbf{w}}\mathbf{w}\mathbf{R}\mathbf{w}^T=[2\mathbf{w}\mathbf{R}(:,0),2\mathbf{w}\mathbf{R}(:,1),\ldots,2\mathbf{w}\mathbf{R}(:,N-1)]=2\mathbf{w}\mathbf{R}。$$

Return

附录B. 误差曲线是抛物线的说明

XI'AN JIAOTONG UNIVERSITY

已知 $y=\sum_{i=0}^{N-1}w_i x_i$，$E[\varepsilon^2]=E[(d-y)^2]$

考察 w_I 的改变对 $E[\varepsilon^2]$ 的影响：

$$y=\sum_{i=0}^{N-1}w_i x_i=w_I x_I+\sum_{\substack{i=0\\ i\neq I}}^{N-1}w_i x_i=w_I x_I+C$$

所以 $E[\varepsilon^2]=E[(d-y)^2]=E[(d-w_I x_I-C)^2]$

$$=E[x_I^2]w_I^2-E[2(d-C)x_I]w_I+E[(d-C)^2]$$

即 $E[\varepsilon^2]$ 是关于 w_I 的一元二次方程。

Return

附录C. 对复合全微分的说明

XI'AN JIAOTONG UNIVERSITY

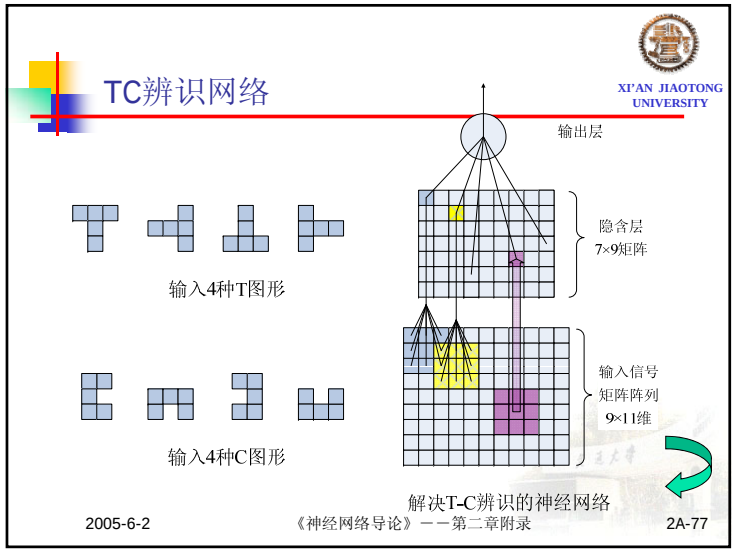
设 z 是 $y_i, i=1,\cdots,N$ 的函数，而 y_i 又是 x 的函数，即

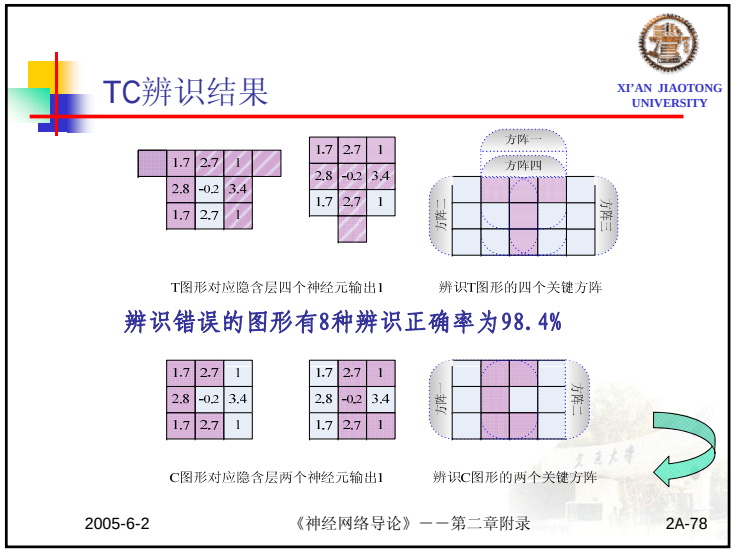
$$z=f_0(y_1,y_2,\ldots,y_N),\qquad y_i=f_i(x),i=1,2,\ldots N$$

则，z 对 x 的导数为

$$\frac{dz}{dx}=\frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x}+\frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}+\cdots+\frac{\partial z}{\partial y_N}\frac{\partial y_N}{\partial x}$$
$$=\sum_{i=1}^{N}\frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x}$$

Return


TC辨识网络
XI'AN JIAOTONG UNIVERSITY
输出层
隐含层
7×9矩阵
输入4种T图形
输入信号
矩阵阵列
9×11维
输入4种C图形
解决T-C辨识的神经网络
2005-6-2
《神经网络导论》——第二章附录
2A-77


TC辨识结果
XI'AN JIAOTONG UNIVERSITY
1.7 2.7 1
2.8 -0.2 3.4
1.7 2.7 1
方阵一
方阵四
方阵二
方阵三
T图形对应隐含层四个神经元输出1
辨识T图形的四个关键方阵
辨识错误的图形有8种辨识正确率为98.4%
C图形对应隐含层两个神经元输出1
辨识C图形的两个关键方阵
2005-6-2
《神经网络导论》——第二章附录
2A-78