

## 5.4 BP 神经网络的基本原理

BP (Back Propagation) 网络是 1986 年由 Rinehart 和 McClelland 为首的科学家小组提出, 是一种按误差逆传播算法训练的多层前馈网络, 是目前应用最广泛的神经网络模型之一。BP 网络能学习和存贮大量的输入-输出模式映射关系, 而无需事前揭示描述这种映射关系的数学方程。它的学习规则是使用最速下降法, 通过反向传播来不断调整网络的权值和阈值, 使网络的误差平方和最小。BP 神经网络模型拓扑结构包括输入层 (input)、隐层 (hide layer) 和输出层 (output layer) (如图 5.2 所示)。

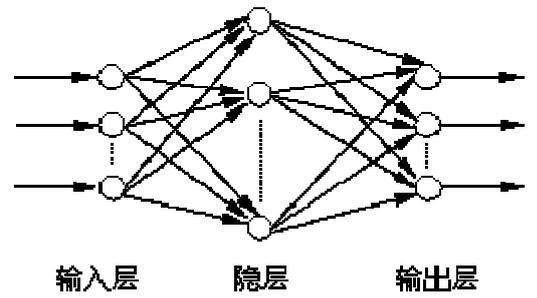


图 5.2 BP 神经网络结构示意图

### 5.4.1 BP 神经元

图 5.3 给出了第  $j$  个基本 BP 神经元 (节点), 它只模仿了生物神经元所具有的三个最基本的也是最重要的功能: 加权、求和与转移。其中  $x_1, x_2 \cdots x_i \cdots x_n$  分别代表来自神经元 1、2  $\cdots$   $i \cdots n$  的输入;  $w_{j1}, w_{j2} \cdots w_{ji} \cdots w_{jn}$  则分别表示神经元 1、2  $\cdots$   $i \cdots n$  与第  $j$  个神经元的连接强度, 即权值;  $b_j$  为阈值;  $f(\cdot)$  为传递函数;  $y_j$  为第  $j$  个神经元的输出。

第  $j$  个神经元的净输入值  $s_j$  为:

$$s_j = \sum_{i=1}^n w_{ji} \cdot x_i + b_j = W_j X + b_j \quad (5.12)$$

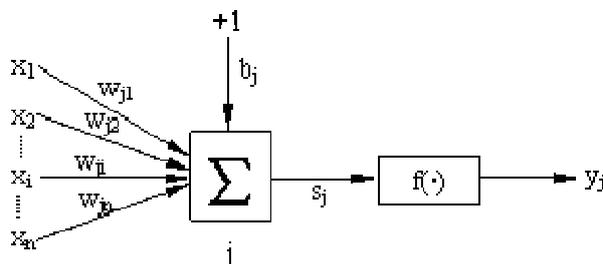


图 5.3 BP 神经元

其中:  $X = [x_1 \ x_2 \ \cdots \ x_i \ \cdots \ x_n]^T$        $W_j = [w_{j1} \ w_{j2} \ \cdots \ w_{ji} \ \cdots \ w_{jn}]$

若视  $x_0 = 1$ ,  $w_{j0} = b_j$ , 即令  $X$  及  $W_j$  包括  $x_0$  及  $w_{j0}$ , 则

$$X = [x_0 \ x_1 \ x_2 \ \cdots \ x_i \ \cdots \ x_n]^T \quad W_j = [w_{j0} \ w_{j1} \ w_{j2} \ \cdots \ w_{jn} \ \cdots \ w_{jn}]$$

于是节点  $j$  的净输入  $S_j$  可表示为:

$$S_j = \sum_{i=0}^n w_{ji} x_i = W_j X \quad (5.13)$$

净输入  $S_j$  通过传递函数 (Transfer Function)  $f(\cdot)$  后, 便得到第  $j$  个神经元的输出  $y_j$ :

$$y_j = f(S_j) = f\left(\sum_{i=0}^n w_{ji} \cdot x_i\right) = F(W_j X) \quad (5.14)$$

式中  $f(\cdot)$  是单调上升函数, 而且必须是有界函数, 因为细胞传递的信号不可能无限增加, 必有一最大值。

#### 5.4.2 BP 网络

BP 算法由数据流的前向计算 (正向传播) 和误差信号的反向传播两个过程构成。正向传播时, 传播方向为输入层  $\rightarrow$  隐层  $\rightarrow$  输出层, 每层神经元的状态只影响下一层神经元。若在输出层得不到期望的输出, 则转向误差信号的反向传播流程。通过这两个过程的交替进行, 在权向量空间执行误差函数梯度下降策略, 动态迭代搜索一组权向量, 使网络误差函数达到最小值, 从而完成信息提取和记忆过程。

##### 5.4.2.1 正向传播

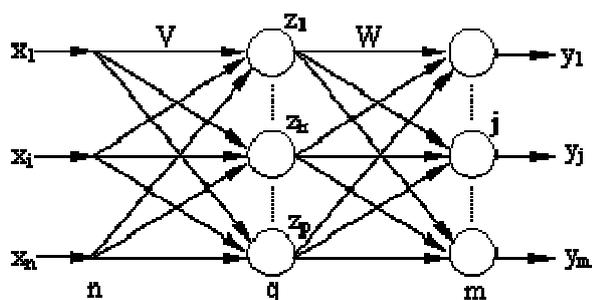


图 5.4 三层神经网络的拓扑结构

设 BP 网络的输入层有  $n$  个节点，隐层有  $q$  个节点，输出层有  $m$  个节点，输入层与隐层之间的权值为  $v_{ki}$ ，隐层与输出层之间的权值为  $w_{jk}$ ，如图 5.4 所示。隐层的传递函数为  $f_1(\cdot)$ ，输出层的传递函数为  $f_2(\cdot)$ ，则隐层节点的输出为（将阈值写入求和项中）：

$$z_k = f_1\left(\sum_{i=1}^n v_{ki} x_i\right) \quad k=1, 2, \dots, q \quad (5.15)$$

输出层节点的输出为：

$$y_j = f_2\left(\sum_{k=1}^q w_{jk} z_k\right) \quad j=1, 2, \dots, m \quad (5.16)$$

至此 B-P 网络就完成了  $n$  维空间向量对  $m$  维空间的近似映射。

#### 5.4.2.2 反向传播

##### 1) 定义误差函数

输入  $P$  个学习样本，用  $x^1, x^2, \dots, x^p, \dots, x^P$  来表示。第  $p$  个样本输入到网络后得到输出  $y_j^p$  ( $j=1, 2, \dots, m$ )。采用平方型误差函数，于是得到第  $p$  个样本的误差  $E_p$ ：

$$E_p = \frac{1}{2} \sum_{j=1}^m (t_j^p - y_j^p)^2 \quad (5.17)$$

式中： $t_j^p$  为期望输出。

对于  $P$  个样本，全局误差为：

$$E = \frac{1}{2} \sum_{p=1}^P \sum_{j=1}^m (t_j^p - y_j^p)^2 = \sum_{p=1}^P E_p \quad (5.18)$$

2) 输出层权值的变化

采用累计误差 BP 算法调整  $w_{jk}$ , 使全局误差  $E$  变小, 即

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \frac{\partial}{\partial w_{jk}} \left( \sum_{p=1}^P E_p \right) = \sum_{p=1}^P \left( -\eta \frac{\partial E_p}{\partial w_{jk}} \right) \quad (5.19)$$

式中:  $\eta$ —学习率

定义误差信号为:

$$\delta_p = -\frac{\partial E_p}{\partial s_j} = -\frac{\partial E_p}{\partial y_j} \cdot \frac{\partial y_j}{\partial s_j} \quad (5.20)$$

其中第一项:

$$\frac{\partial E_p}{\partial y_j} = \frac{\partial}{\partial y_j} \left[ \frac{1}{2} \sum_{i=1}^m (t_i^p - y_i^p)^2 \right] = -\sum_{i=1}^m (t_i^p - y_i^p) \quad (5.21)$$

第二项:

$$\frac{\partial y_j}{\partial s_j} = f_2'(s_j) \quad (5.22)$$

是输出层传递函数的偏微分。

于是:

$$\delta_p = \sum_{i=1}^m (t_i^p - y_i^p) f_2'(s_j) \quad (5.23)$$

由链定理得:

$$\frac{\partial E_p}{\partial w_{jk}} = \frac{\partial E_p}{\partial S_j} \cdot \frac{\partial S_j}{\partial w_{jk}} = -\delta_p z_k = -\sum_{j=1}^m (t_j^p - y_j^p) f_2'(S_j) \cdot z_k \quad (5.24)$$

于是输出层各神经元的权值调整公式为：

$$\Delta w_{jk} = \sum_{p=1}^P \sum_{j=1}^m \eta (t_j^p - y_j^p) f_2'(S_j) z_k \quad (5.25)$$

3) 隐层权值的变化

$$\Delta v_k = -\eta \frac{\partial E}{\partial v_k} = -\eta \frac{\partial}{\partial v_k} \left( \sum_{p=1}^P E_p \right) = \sum_{p=1}^P \left( -\eta \frac{\partial E_p}{\partial v_k} \right) \quad (5.26)$$

定义误差信号为：

$$\delta_{pk} = -\frac{\partial E_p}{\partial S_k} = -\frac{\partial E_p}{\partial z_k} \cdot \frac{\partial z_k}{\partial S_k} \quad (5.27)$$

其中第一项：

$$\frac{\partial E_p}{\partial z_k} = \frac{\partial}{\partial z_k} \left[ \frac{1}{2} \sum_{j=1}^m (t_j^p - y_j^p)^2 \right] = -\sum_{j=1}^m (t_j^p - y_j^p) \frac{\partial y_j^p}{\partial z_k} \quad (5.28)$$

依链定理有：

$$\frac{\partial y_j^p}{\partial z_k} = \frac{\partial y_j^p}{\partial S_j} \cdot \frac{\partial S_j}{\partial z_k} = f_2'(S_j) w_{jk} \quad (5.29)$$

第二项：

$$\frac{\partial z_k}{\partial S_k} = f_1'(S_k) \quad (5.30)$$

是隐层传递函数的偏微分。

于是：

$$\delta_{zk} = \sum_{j=1}^m (t_j^p - y_j^p) f_2'(S_j) w_{jk} f_1'(S_k) \quad (5.31)$$

由链定理得：

$$\frac{\partial E_p}{\partial v_{ki}} = \frac{\partial E_p}{\partial S_k} \cdot \frac{\partial S_k}{\partial v_{ki}} = -\delta_{zk} x_i = -\sum_{j=1}^m (t_j^p - y_j^p) f_2'(S_j) w_{jk} f_1'(S_k) \cdot x_i \quad (5.32)$$

从而得到隐层各神经元的权值调整公式为：

$$\Delta v_{ki} = \sum_{p=1}^P \sum_{j=1}^m \eta (t_j^p - y_j^p) f_2'(S_j) w_{jk} f_1'(S_k) x_i \quad (5.33)$$

### 5.4.3 BP 算法的改进

BP 算法理论具有依据可靠、推导过程严谨、精度较高、通用性较好等优点，但**标准 BP 算法存在以下缺点：收敛速度缓慢；容易陷入局部极小值；难以确定隐层数和隐层节点个数。**在实际应用中，BP 算法很难胜任，因此出现了很多改进算法。

#### 1) 利用动量法改进 BP 算法

标准 BP 算法实质上是一种简单的最速下降静态寻优方法，**在修正  $W(K)$  时，只按照第  $K$  步的负梯度方向进行修正**，而没有考虑到以前积累的经验，即以前时刻的梯度方向，从而常常使学习过程发生振荡，收敛缓慢。动量法权值调整算法的具体做法是：将上一次权值调整量的一部分迭加到按本次误差计算所得的权值调整量上，作为本次的实际权值调整量，即：

$$\Delta W(n) = -\eta \nabla E(n) + \alpha \Delta W(n-1) \quad (5.34)$$

其中： $\alpha$  为动量系数，通常  $0 < \alpha < 0.9$ ； $\eta$ —学习率，范围在  $0.001 \sim 10$  之间。这种方法所加的动量因子实际上相当于阻尼项，它减小了学习过程中的振荡趋势，从而改善了收敛性。动量法降低了网络对于误差曲面局部细节的敏感性，有效的抑制了网络陷入局部极小。

#### 2) 自适应调整学习速率

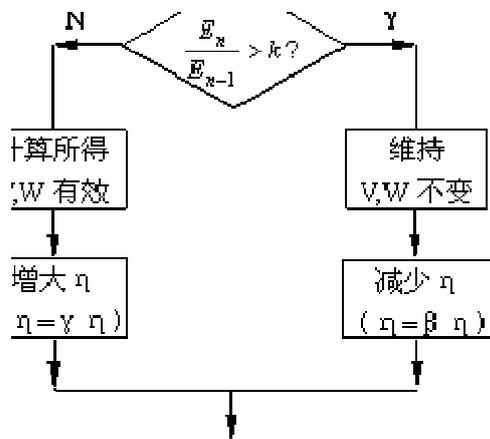


图 5.5 自适应学习

标准 BP 算法收敛速度缓慢的一个重要原因是学习率选择不当，学习率选得太小，收敛太慢；学习率选得太大，则有可能修正过头，导致振荡甚至发散。可采用图 5.5 所示的自适应方法调整学习率。

调整的基本指导思想是：在学习收敛的情况下，增大  $\eta$ ，以缩短学习时间；当  $\eta$  偏大致使不能收敛时，要及时减小  $\eta$ ，直到收敛为止。

### 3) 动量-自适应学习速率调整算法

采用动量法时，BP 算法可以找到更优的解；采用自适应学习速率法时，BP 算法可以缩短训练时间。将以上两种方法结合起来，就得到动量-自适应学习速率调整算法。

### 4) L-M 学习规则

L-M (Levenberg-Marquardt) 算法比前述几种使用梯度下降法的 BP 算法要快得多，但对于复杂问题，这种方法需要相当大的存储空间。L-M (Levenberg-Marquardt) 优化方法的权值调整率选为：

$$\Delta w = (J^T J + \mu I)^{-1} \cdot J^T e \quad (5.35)$$

其中： $e$ —误差向量； $J$ —网络误差对权值导数的雅可比 (Jacobian) 矩阵； $\mu$ —标量，当  $\mu$  很大时上式接近于梯度法，当  $\mu$  很小时上式变成了 Gauss-Newton 法，在这种方法中， $\mu$  也是自适应调整的。

综合考虑，拟采用 L-M 学习规则和动量法分别作为神经网络的训练函数和学习函数。

## 5.5 BP 神经网络的训练策略及结果

本文借助于 MATLAB 神经网络工具箱来实现多层前馈 BP 网络 (Multi-layer feed-forward backpropagation network) 的颜色空间转换，免去了许多编写计算机程序的烦恼。神经网络的实际输出值与输入值以及各权值和阈值有关，为了使实际输出值与网络期望输出值相吻合，可用含有一定数量学习样本的样本集和相应期望输出值的集合来训练网络。训练时仍然使用本章 5.2 节中所述的实测样本数据。

另外，目前尚未找到较好的网络构造方法。确定神经网络的结构和权系数来描述给定的映射或逼近一个未知的映射，只能通过学习方式得到满足要求的网络模型。神经网络的学习可以理解为：对确定的网络结构，寻找一组满足要求的权系数，使给定的误差函数最小。设计多层前馈网络时，主要侧重试验、探讨多种模型方案，在实验中改进，直到选取一个满意方案为止，可按下列步骤进行：对任何实际问题先都只选用一个隐层；使用很少的隐层节点数；不断增加隐层节点数，直到获得满意性能为止；否则再采用两个隐层重复上述过程。

训练过程实际上是根据目标值与网络输出值之间误差的大小反复调整权值和阈值，直到此误差达到预定值为止。

### 5.5.1 确定 BP 网络的结构

确定了网络层数、每层节点数、传递函数、初始权系数、学习算法等也就确定了 BP 网络。确定这些选项时有一定的指导原则，但更多的是靠经验和试凑。

#### 1) 隐层数的确定：

1998 年 Robert Hecht-Nielson 证明了对任何在闭区间内的连续函数，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  $m$  维的映照。因此我们从含有一个隐层的网络开始进行训练。

#### 2) BP 网络常用传递函数：

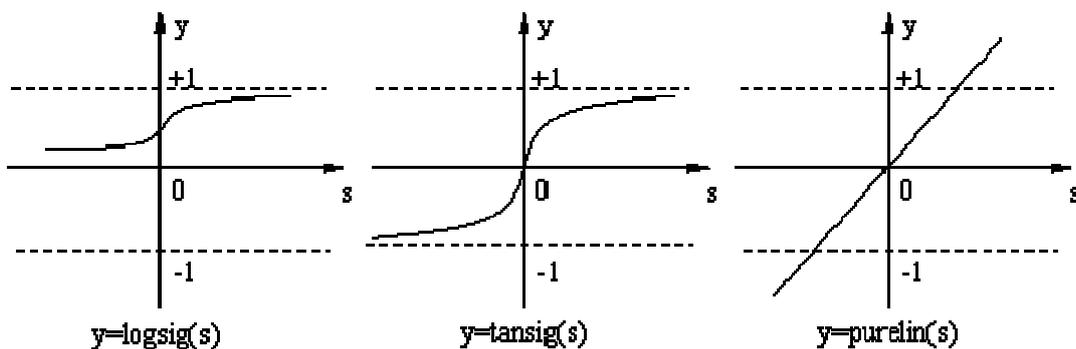


图 5.6 BP 网络常用的传递函数

BP 网络的传递函数有多种。Log-sigmoid 型函数的输入值可取任意值，输出值在 0 和 1 之间；tan-sigmoid 型传递函数  $\text{tansig}$  的输入值可取任意值，输出值在 -1 到 +1 之间；线性传递函数  $\text{purelin}$  的输入与输出值可取任意值。BP 网络通常有一个或多个隐层，该层中的神

经元均采用 sigmoid 型传递函数，输出层的神经元则采用线性传递函数，整个网络的输出可以取任意值。各种传递函数如图 5.6 所示。

只改变传递函数而其余参数均固定，用本章 5.2 节所述的样本集训练 BP 网络时发现，传递函数使用 tansig 函数时要比 logsig 函数的误差小。于是在以后的训练中隐层传递函数改用 tansig 函数，输出层传递函数仍选用 purelin 函数。

### 3) 每层节点数的确定：

使用神经网络的目的是实现摄像机输出 RGB 颜色空间与 CIE-XYZ 色空间转换，因此 BP 网络的输入层和输出层的节点个数分别为 3。下面主要介绍隐层节点数量的确定。

对于多层前馈网络来说，隐层节点数的确定是成败的关键。若数量太少，则网络所能获取的用以解决问题的信息太少；若数量太多，不仅增加训练时间，更重要的是隐层节点过多还可能出现所谓“过渡吻合”（Overfitting）问题，即测试误差增大导致泛化能力下降，因此合理选择隐层节点数非常重要。关于隐层数及其节点数的选择比较复杂，一般原则是：在能正确反映输入输出关系的基础上，应选用较少的隐层节点数，以使网络结构尽量简单。本文中采用网络结构增长型方法，即先设置较少的节点数，对网络进行训练，并测试学习误差，然后逐渐增加节点数，直到学习误差不再有明显减少为止。

## 5.5.2 误差的选取

在神经网络训练过程中选择均方误差 MSE 较为合理，原因如下：

① 标准 BP 算法中，误差定义为：

$$E_p = \frac{1}{2} \sum_{j=1}^m (t_j^p - y_j^p)^2 \quad (5.36)$$

每个样本作用时，都对权矩阵进行了一次修改。由于每次权矩阵的修改都没有考虑权值修改后其它样本作用的输出误差是否也减小，因此将导致迭代次数增加。

② 累计误差 BP 算法的全局误差定义为：

$$E = \frac{1}{2} \sum_{p=1}^P \sum_{j=1}^m (t_j^p - y_j^p)^2 = \sum_{p=1}^P E_p \quad (5.37)$$

这种算法是为了减小整个训练集的全局误差，而不针对某一特定样本，因此如果作某种修改能使全局误差减小，并不等于说每一个特定样本的误差也都能同时减小。它不能用来比较 P 和 m 不同的网络性能。因为对于同一网络来说，P 越大，E 也越大；P 值相同，m 越大 E 也越大。

③ 均方误差 MSE:

$$MSE = \frac{1}{PP} \sum_{p=1}^P \sum_{j=1}^m (\hat{y}_{pj} - y_{pj})^2 \quad (5.38)$$

其中： $m$ —输出节点的个数， $P$ —训练样本数目， $\hat{y}_{pj}$ —网络期望输出值， $y_{pj}$ —网络实际输出值。均方误差克服了上述两种算法的缺点，所以选用均方误差算法较合理。

### 5.5.3 训练结果

训练一个单隐层的三层 BP 网络，根据如下经验公式选择隐层节点数<sup>[125]</sup>：

$$n_1 = \sqrt{n + m} + a \quad (5.39)$$

式中： $n$  为输入节点个数， $m$  为输出节点个数， $a$  为 1 到 10 之间的常数。针对本论文  $n_1$  取值范围为 3~13。训练结果如表 5.1 所示。

表 5.1 隐层节点数与误差的关系

| 隐层神经元个数 | 训练误差     | 测试误差   |
|---------|----------|--------|
| 3       | 1.25661  | 1.1275 |
| 4       | 0.797746 | 0.8232 |
| 5       | 0.631849 | 0.7278 |
| 6       | 0.570214 | 0.6707 |
| 7       | 0.552873 | 0.6895 |
| 8       | 0.445118 | 0.6575 |
| 9       | 0.385578 | 0.6497 |
| 10      | 0.259624 | 0.4555 |
| 11      | 0.185749 | 0.6644 |
| 12      | 0.183878 | 0.48   |

|    |          |        |
|----|----------|--------|
| 13 | 0.168587 | 0.6671 |
|----|----------|--------|

由上表可以看出：

- ① 增加隐层节点数可以减少训练误差，但超过 10 以后测试误差产生波动，即泛化能力发生变化。综合比较隐层节点数为 10 与 12 的训练误差和测试误差，决定隐层节点数选用 12。
- ② 训练误差和测试误差都很大，而且收敛速度极慢（训练过程如图 5.7 所示），这个问题可以通过对输出量进行归一化来解决。

根据 Sigmoid 型传递函数输入和输出的范围，对输入变量不进行归一化处理，只对输出变量进行归一化，这是因为在输出数据要求归一化的同时，对输入数据也进行归一化的话，权值的可解释性就更差了。目标值按下式进行变化：

$$d' = \frac{d - d_{\min}}{d_{\max} - d_{\min}} \times 0.9 + 0.05 \quad (5.40)$$

使目标值落在 0.05~0.95 之间，这样靠近数据变化区间端点的网络输出值就有一波动范围，网络的性能较好。用新生成的训练样本与测试样本对隐层节点数为 12 的网络进行训练，得到的训练误差为  $9.89028 \times 10^{-5}$ ，测试误差为  $1.9899 \times 10^{-4}$ ，达到了预定的目标（训练过程如图 5.8 所示）。

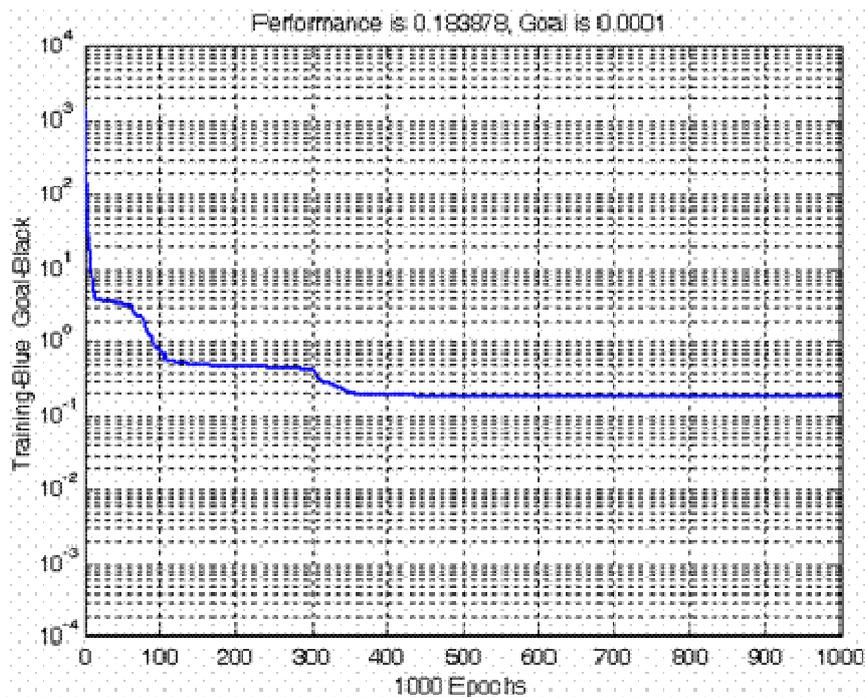


图 5.7 隐层节点数为 12 的神经网络训练过程

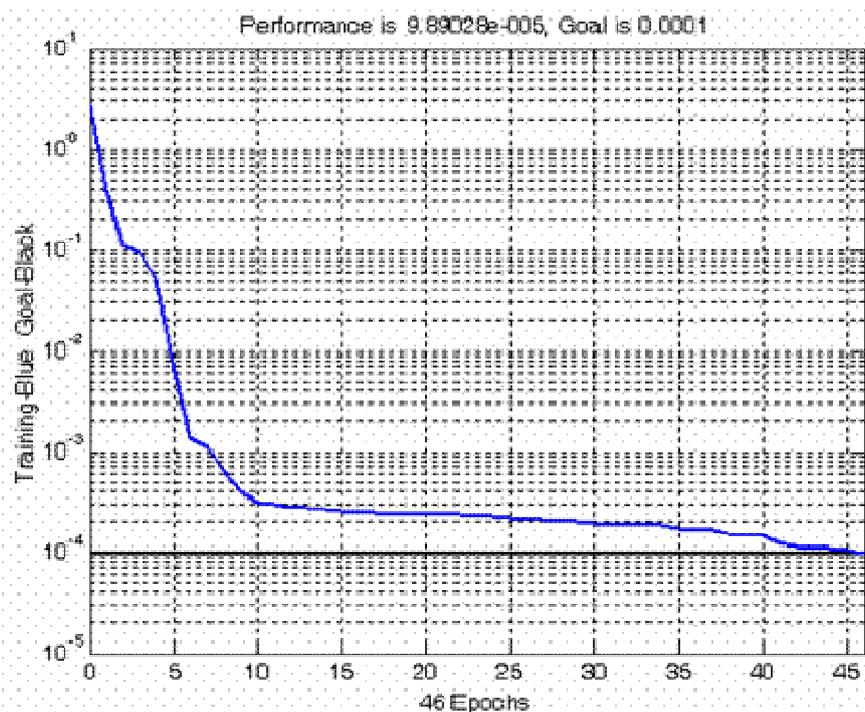
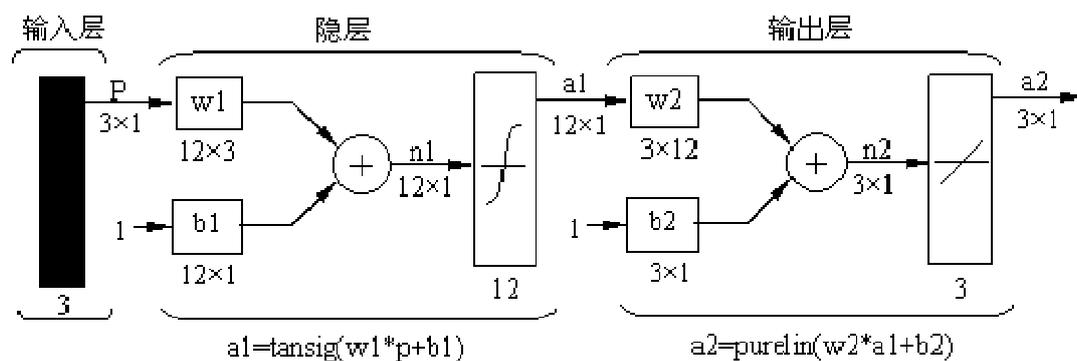


图 5.8 归一化训练样本后隐层节点数为 12 的神经网络训练过程

## 5.6 最终训练后的神经网络结构

采用三层 BP 网络实现摄像机输出 RGB 颜色空间与 CIEXYZ 色空间转换，其中隐层含有 12 个节点，传递函数采用 tansig 函数；输出层传递函数选用 purelin 函数。经过测试后结果满意，可以认为该神经网络可以用来实现这个关系映射。网络的结构如图 5.9 所示：



其中：R—输入层节点数 S1—隐层节点数 S2—输出层节点数

图 5.9 三层 BP 网络结构

得到的 BP 神经网络的权值和阈值为：

$$W_1 = \begin{bmatrix} -0.0029 & 0.0237 & -0.0098 \\ -0.0088 & 0.0095 & -0.0122 \\ -0.0098 & 0.0019 & 0.0004 \\ 0.0792 & -0.1199 & 0.0324 \\ 0.0067 & 0.0341 & -0.0129 \\ 0.0030 & 0.0104 & -0.0129 \\ -0.0043 & -0.0142 & -0.0127 \\ -0.0011 & -0.0016 & 0.0085 \\ 0.1392 & -0.1281 & -0.0141 \\ 0.0094 & 0.0879 & -0.0807 \\ -0.0031 & 0.0244 & -0.0208 \\ -0.0214 & -0.0529 & 0.0058 \end{bmatrix} \quad b_1 = \begin{bmatrix} -2.4894 \\ 3.4834 \\ 1.9623 \\ 1.9752 \\ -5.2871 \\ -2.2328 \\ 3.9847 \\ -1.2996 \\ 1.9118 \\ -5.0905 \\ -0.5144 \\ 4.8162 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.2393 & 0.4086 & 0.1712 \\ 1.1705 & 0.7430 & -0.0577 \\ -0.4251 & -0.3157 & -0.1085 \\ 0.0684 & 0.0553 & 0.0018 \\ 0.0064 & -0.0457 & -0.1109 \\ -0.0941 & -0.2796 & 0.0034 \\ 0.0283 & 0.0191 & 0.0731 \\ 0.8756 & 0.7501 & 1.4704 \\ 0.2924 & 0.1938 & 0.3203 \\ -0.0023 & -0.0106 & -0.0116 \\ 0.1497 & 0.1433 & 0.0429 \\ -0.0162 & -0.0207 & 0.0164 \end{bmatrix}^T \quad b_2 = \begin{bmatrix} -0.1068 \\ 0.1532 \\ 1.1378 \end{bmatrix}$$

## 5.7 本章小结

- 1) 定量地分析了用线性关系转换摄像机 RGB 空间到 CIE-XYZ 空间数据后产生的均方误差，表明 CCD 摄像机与标准观察者之间有明显的差别，也就是说 RGB 与 CIE-XYZ 间的转换是非线性的。
- 2) 采用 MATLAB 中神经网络工具箱实现多层前馈 BP 网络的 RGB 到 CIEXYZ 颜色空间转换，用经过归一化的训练样本与测试样本对隐层节点数为 12 的三层网络进行训练，得到的训练误差为  $9.89028 \times 10^{-5}$ ，测试误差为  $1.9899 \times 10^{-4}$ ，结果表明经过训练的多层前馈 BP 网络可以满足 RGB 空间向 CIEXYZ 颜色空间转换要求，达到了预定目标。
- 3) 确定了用于 RGB 和 XYZ 颜色空间转换的 BP 网络结构，并求出了该神经网络的权值和阈值。使用该网络可以定量表达食品颜色，定量比较高压加工食品颜色的变化，可以使食品颜色测定和控制实现定量化，而不再是主观性很强的模糊描述。