第九讲

模式识别

(Pattern Recognition)

武汉大学计算机学院

Email:18986211797@189.cn

第6章 神经网络

6.1 神经网络基本知识

6.2 感知器神经网络模型

6.3 BP神经网络模型

6.4 Hopfield神经网络模型(：*不要求)

6.3 BP神经网络模型

一. BP神经网络简介

1. 多层感知器

单层感知器网络只能解决线性可分问题。 在单层感知器网络的输入层和输出层之间加入一层或多层感知器单元(层中可以有多个感知器)作为隐含层，就构成了多层感知器 (Multilayer Perceptrons)。多层感知器可以解决线性不可分的输入向量的分类问题。这种由输入层、隐含层和输出层所构成的感知器神经网络就是多层前向神经网络，BP神经网络是其中一种。

2. BP神经网络简介

BP神经网络是采用误差反向传播(Back Propagation, BP)算法的多层前向网络，其中，神经元的传输函数为S型函数，网络的输入和输出是一种非线性映射关系。

BP算法的学习过程由结果正向传播和误差反向传播组成。在正向传播过程中，输入模式从输入层经隐含层逐层处理后，传送至输出层。每一层神经元的状态只影响下一层神经元的状态。如果在输出层得不到期望输出，那么就转为反向传播，把误差信号沿原连接路径返回，并通过修改各层神经元的权值，使误差减小。

二. BP神经网络模型

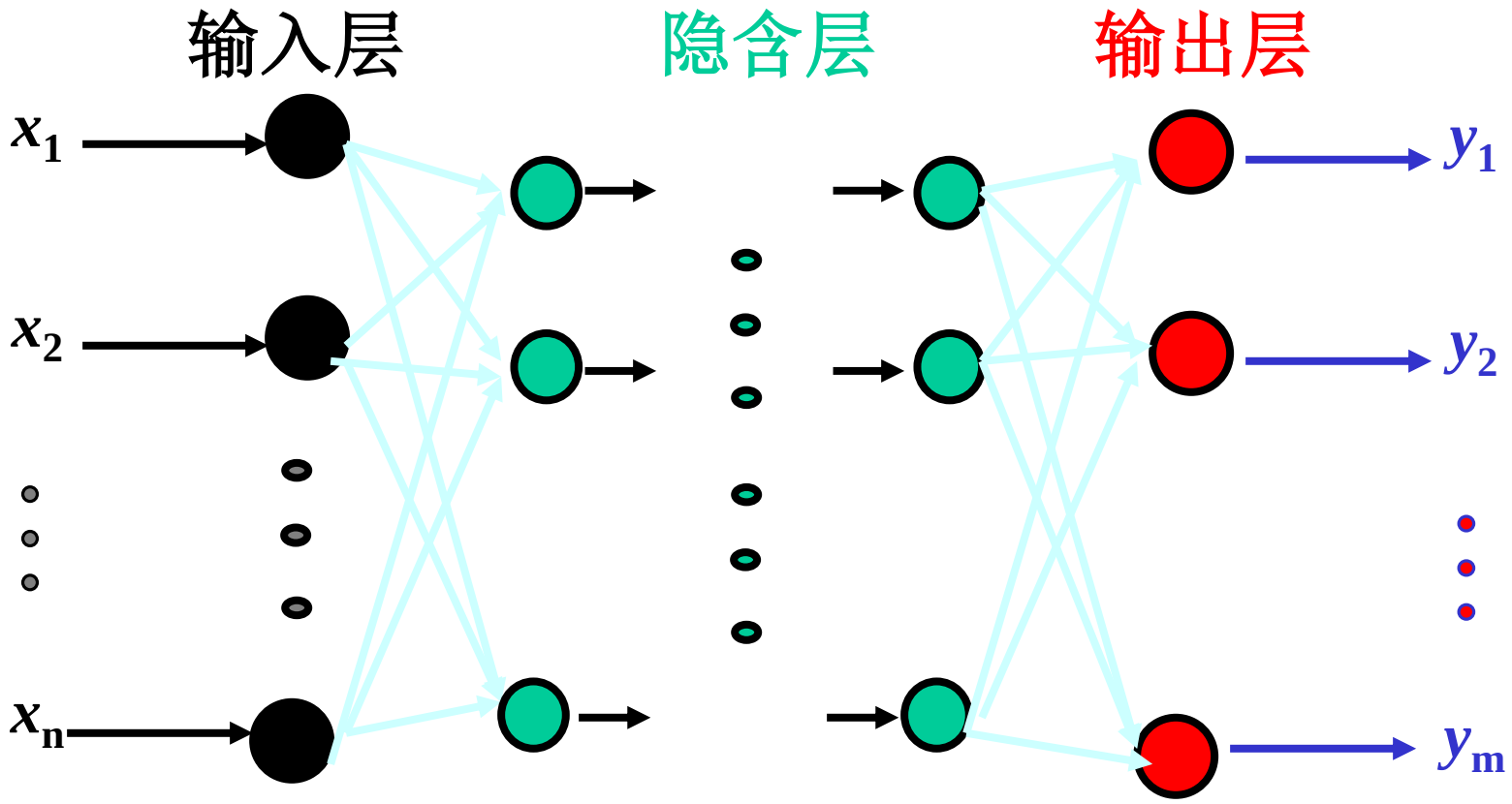


图　BP神经网络模型

BP网络为多层前向神经网络。

层与层之间多采用全互连方式，但同一层的节点之间不存在相互连接。

神经元传输函数是可微的，大多数设计者通常都采用**S型(Sigmoid)函数。**

神经元*neu*(*neuron*)的输入：

$$neu=b+x_1w_1+x_2w_2+\ldots+x_nw_n$$

神经元*neu*的输出：

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

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

$$= \frac{1+e^{-neu}-1}{(1+e^{-neu})^2}$$

$$= \frac{1}{1+e^{-neu}} - \frac{1}{(1+e^{-neu})^2}$$

$$= y - y^2 = y(1-y)$$

注意到：$\lim_{neu\to+\infty}\frac{1}{(1+e^{-neu})}=1$，$\lim_{neu\to-\infty}\frac{1}{(1+e^{-neu})}=0$

根据*S*型传输函数可知，*y*的值域为(0,1)，从而$f'(neu)$的值域为(0,0.25)。见下图所示。

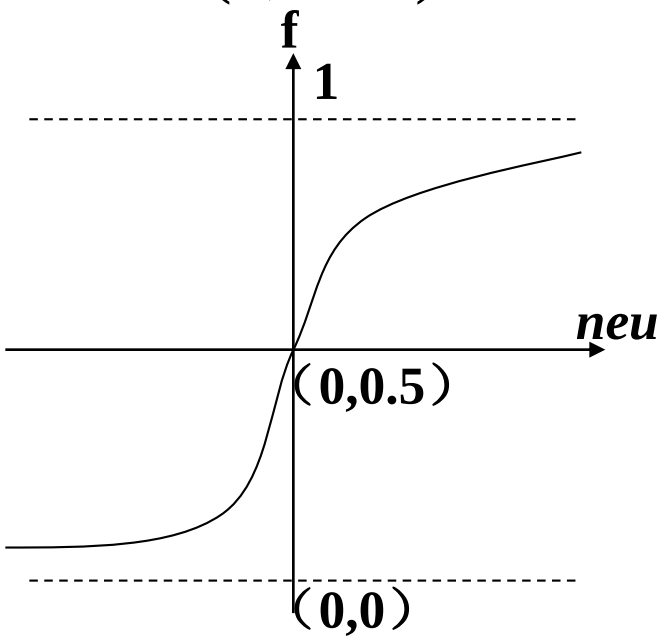


图 S型传输函数

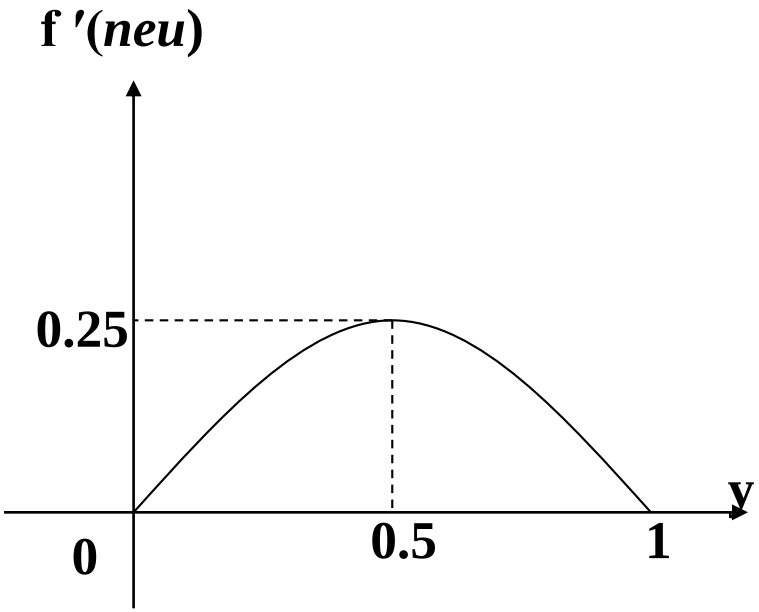


图 f′ (*neu*)的图像

由上图可知，对神经网络进行训练，应该将*neu*的值尽量控制在收敛比较快的范围内。实际上，也可以用其它函数作为BP网络神经元的传输函数，只要该函数满足处处可导的条件即可。

三. BP网络训练过程

首先，上一讲已提到，ANNs的学习过程是根据模式样本集对神经元之间的连接权值进行调整的过程，BP网络也不例外。其次，BP网络执行是有导师学习。因此，其样本集是由形如：

(输入向量，理想输出向量)

的向量对构成的。

在开始训练前，所有的权都应用一些不同的小随机数进行初始化。“小随机数”用来确保网络不会因为权值过大而进入饱和状态，从而导致训练失败；“不同”用来保证网络可以正常学习。事实上，如果用相同的数去初始化权值矩阵，则网络无学习能力。

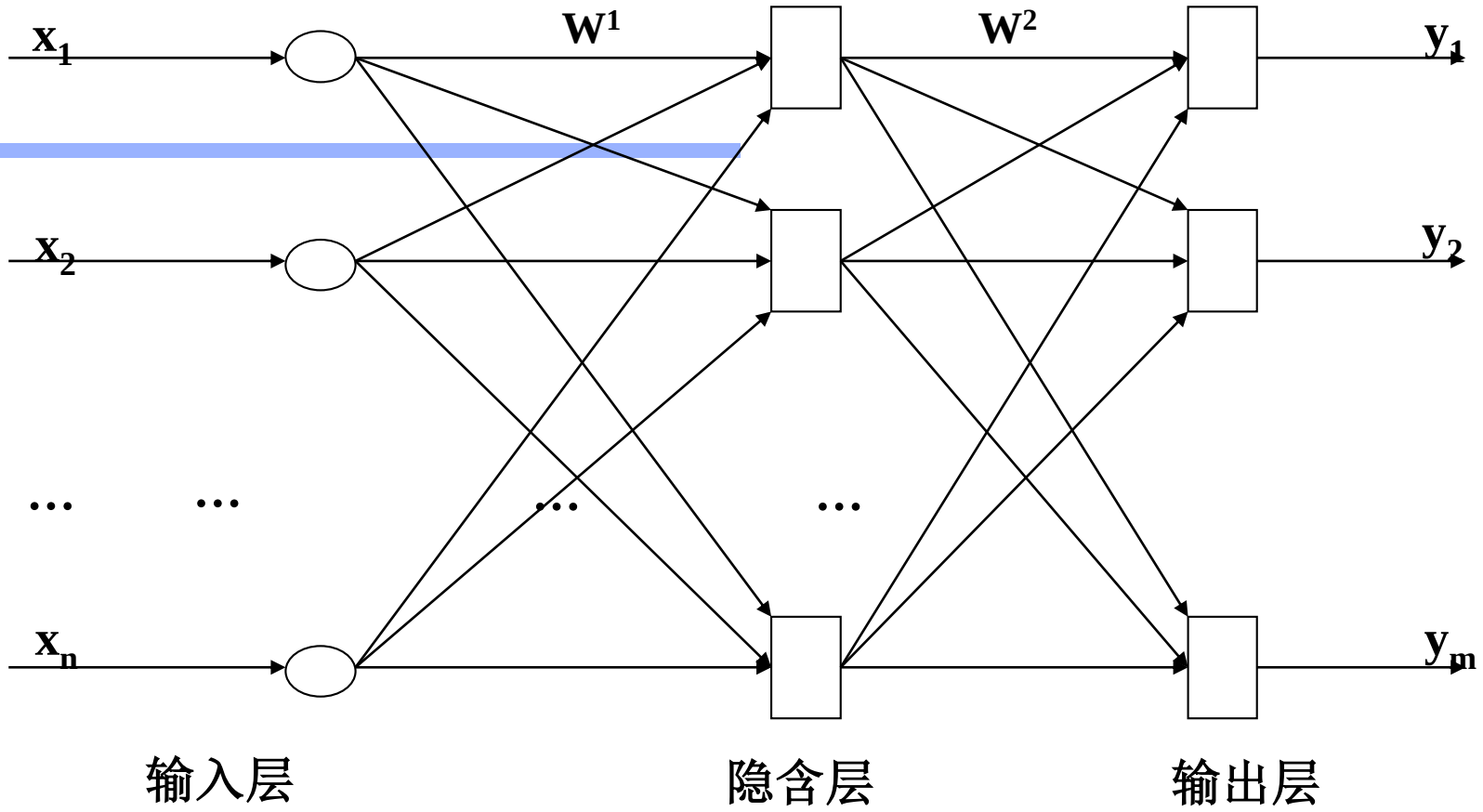


图 3层BP神经网络

BP基本算法主要包含4步，这4步被分成两个阶段：

1.前向(正向)传播输出阶段

Step 1: 给定输入样本 $\mathbf{X}_p = (x_1, x_2, \cdots, x_n)^T$ 和理想输出 $\mathbf{d}_p = (d_1, d_2, \cdots, d_m)^T$ ；将$\mathbf{X}_p$输入到网络。

Step 2: 计算相应的实际输出$\mathbf{y}_p$：

$$\mathbf{y}_p = f_L(\ldots(f_2(f_1(\mathbf{X}_p\mathbf{W}^1)\mathbf{W}^2)\ldots)\mathbf{W}^L)$$

这里，L为网络层数，$\mathbf{W}^i$为第i层的权值矩阵(i=1,2,…,L),f_i为第i层神经元的传输函数。

2. 向后(误差反向)传播(修正权值)阶段

Step 1: 计算实际输出$\mathbf{y}_p$与相应的理想输出$\mathbf{d}_p$的差;

Step 2: 按极小化误差方式调整权值矩阵。

这两个步骤的工作一般要受到精度要求(期望误差ε)的控制，这里取均方误差函数:

$$E_p = \frac{1}{2}\sum_{j=1}^{m}\left(d_j - y_j\right)^2$$

作为网络关于第p个样本的误差测度。

将整个网络样本集的误差测度定义为:

$$E = \sum_{p} E_p$$

四.误差反向传播原理分析

1.输出层权值调整

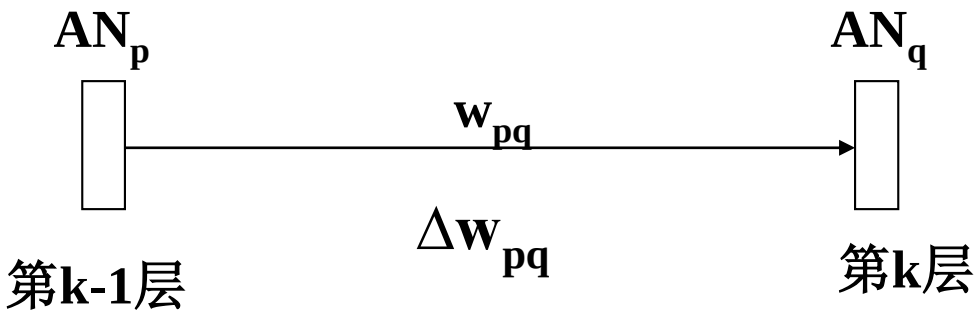


图　AN_p到AN_q的连接

$$w_{pq} = w_{pq} + \Delta w_{pq}$$

$$\Delta w_{pq} = \eta\, \delta_q y_p$$

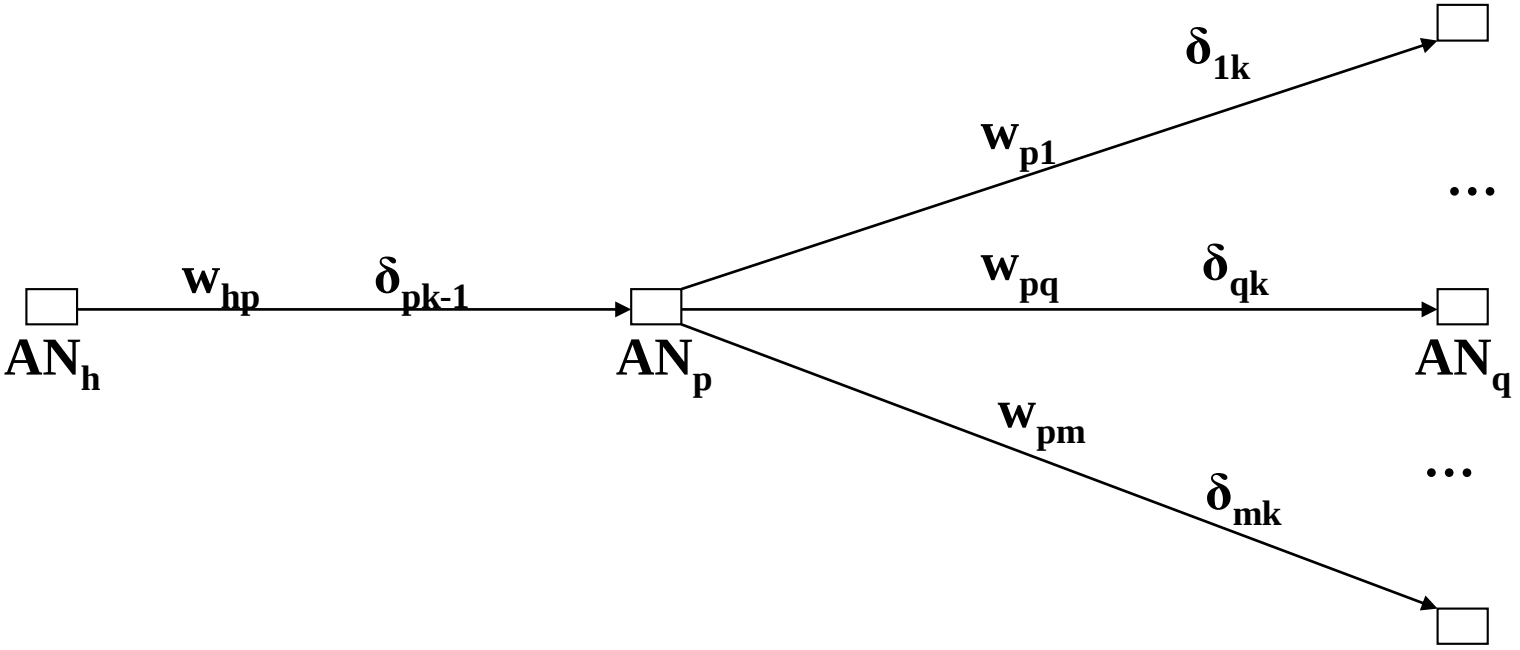
$$= \eta\, f_n'(neu_q)(d_q - y_q) y_p$$

$$= \eta\, y_q(1-y_q)(d_q - y_q) y_p$$

δ学习规则/梯度下降法

(详见上一讲基本学习算法)

2.隐含层权值调整



第k-2层　　第k-1层　　第k层

δ_{pk-1}的值和δ_{1k}，δ_{2k}，…，δ_{mk}有关

不妨认为δ_{pk-1}

通过权值w_{p1}对δ_{1k}做出贡献，

通过权值w_{p2}对δ_{2k}做出贡献，

……

通过权值w_{pm}对δ_{mk}做出贡献。

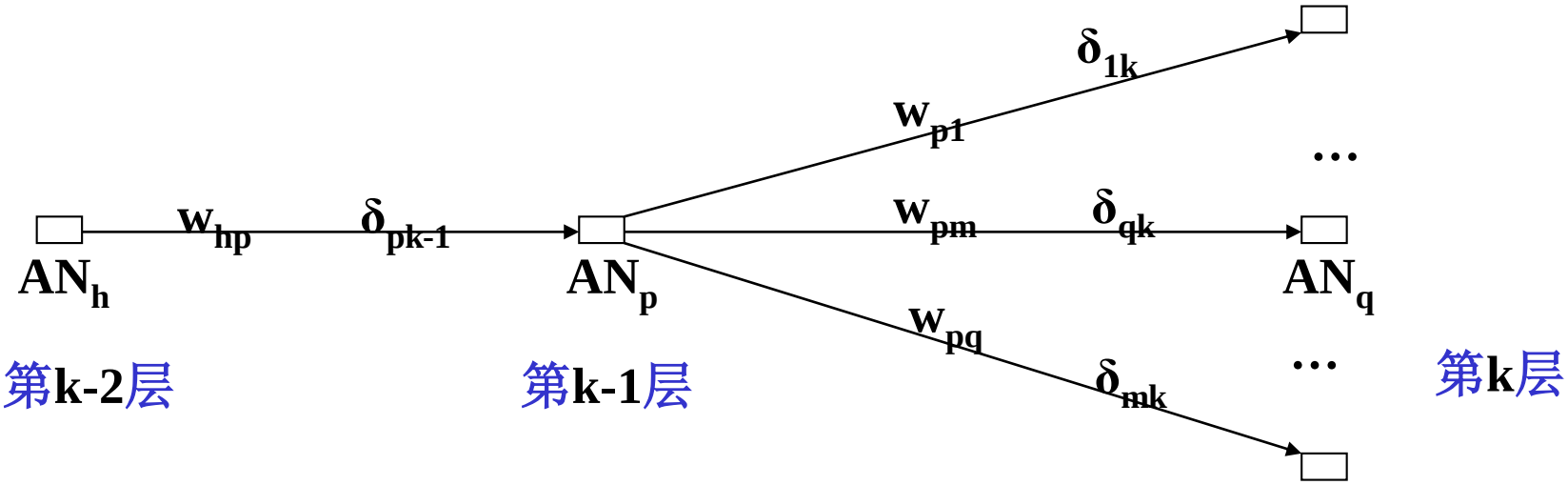
$$\delta_{pk-1} = f_{k-1}'(neu_p)\,(w_{p1}\delta_{1k} + w_{p2}\delta_{2k} + \ldots + w_{pm}\delta_{m\,k})$$

$$w_{hp}=w_{hp}+\Delta w_{hp}$$

$$\Delta w_{hp}=\eta\,\delta_{pk-1}y_{hk-2}$$

$$=\eta f_{k-1}'(neu_p)(w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\ldots+w_{pm}\delta_{mk})y_{hk-2}$$

$$=\eta\,y_{pk-1}(1-y_{pk-1})(w_{p1}\delta_{1k}+w_{p2}\delta_{2k}+\ldots+w_{pm}\delta_{mk})y_{hk-2}$$



五.基本的BP算法思想

模式样本集：$S=\{(X_1,d_1),(X_2,d_2),\ldots,(X_s,d_s)\}$

基本的BP算法思想：

(1)逐一地根据样本集中的样本(X_k,d_k)计算出实际输出y_k和误差测度E_p，对W^1，W^2，…，W^L各做一次调整，重复这个循环，直到$\sum E_p<\varepsilon$。

(2)用输出层的误差调整输出层权值矩阵，并用此误差估计输出层的直接前导层的误差，再用输出层前导层误差估计更前一层的误差。如此获得所有其它各层的误差估计，并用这些估计实现对权值矩阵的修改。形成将输出端表现出的误差沿着与输入信号相反的方向逐级向输入端传递的过程。

基本的BP神经网络学习算法使用δ学习规则/梯度下降法 (MATLAB使用traingd作为梯度下降学习算法标识符)，学习过程是通过调整权值和阈值，使输出期望值和神经网络的实际输出值的均方误差趋于最小实现的，但是它只用到均方误差函数对权值和阈值的一阶导数(梯度)信息，使得算法收敛速度较慢，易陷入局部极小等缺陷。

六.BP神经网络分类器设计

在设计BP神经网络分类器时，需从网络的层数、每层包含的神经元数、初始权值以及学习速率等几方面考虑。

1.网络层数

已经证明：三层BP神经网络可以实现多维单位立方体R^n到R^m的映射，即能够逼近任意有理函数。这实际上给出了设计BP神经网络的基本原则。增加层数可以更进一步地降低误差、提高精度，但同时也使网络复杂化，从而增加网络权值的训练时间。而误差精度的提高实际上也可以通过增加隐含层中的神经元数目来获得，其训练结果也比增加层数更容易观察和调整。因此，一般情况下，应优先考虑增加隐含层的神经元数目。

2.隐含层的神经元数目

网络训练精度的提高，可以通过采用一个隐含层增加神经元数目获得，这在结构实现上要比增加更多的隐含层简单得多。

隐含层神经元数目的确定是目前没有解决的一个难题，一般根据经验和实验确定。

3.初始权值的的选取

由于系统是非线性的，初始权值的选取对于学习是否达到局部最小、是否能能够收敛以及训练时间的长短有很大的关系。

初始权值过大、过小都会影响学习速度，初始仅值一般可取较小的随机数。

4.学习速率

过大的学习速率可能导致系统不稳定，过低的学习速率导致较长的训练时间，可能收敛很慢。在一般情况下，倾向于选取较小的学习速率以保证系统的稳定性。

学习速率 η 一般在**0.01~0. 8**之间选取。

七.BP神经网络的MATLAB编程

在MATLAB的命令行窗口中输入“**help backprop**”可得到BP神经网络相关函数，进一步利用“**help**”命令即可得到相关函数的详细描述。

newff-创建前馈BP网络

语法：net=newff(PR,[S1 S2...SN],
{TF1 TF2...TFN},BTF,BLF,PF)

功能：Create a feed-forward backpropagation network.

参数： 输入

PR-R×2矩阵，即网络输入向量各分量的取值范围；

[S1 S2...SN]-隐含层和输出层神经元数目；

{TF1 TF2...TFN}-隐含层和输出层传输函数，缺省为’**tansig**’(双曲正切传输函数)；

BTF-训练函数，缺省为’**trainlm**’；

BLF-学习函数，缺省为’**learngdm**’(有动量的梯度下降法:**the gradient descent with momentum**)；

输出 **net**-所创建的BP网络(**net**以结构体形式存储)

常用的BTF标识符：

(1)traingd:梯度下降法(基本的BP算法)；

(2)traingdm:有动量的梯度下降法(改进的BP算法)；

(3)traingda:有自适应lr的梯度下降法(改进的BP算法)；

(4)Trainscg(scaled conjugate gradient):量化共轭梯度法(改进的BP算法)；

……

net常用调整参数(梯度下降法)：

net.trainParam.epochs-最大训练次数(缺省为10)

net.trainParam.goal-训练的目标误差(缺省为0)

net.trainParam.lr-学习速率(缺省为0.01)

net.trainParam.show-显示迭代过程(NaN表示不显示，缺省为25)

net.trainParam.time-最大训练时间(缺省为inf:infinity，即∞)

……..

net常用调整参数(有动量梯度下降法:*不要求):

net.trainParam.epochs-最大训练次数(缺省为10)

net.trainParam.goal-训练的目标误差(缺省为0)

net.trainParam.lr-学习速率(缺省为0.01)

net.trainParam.show-显示迭代过程(NaN表示不显示，缺省为25)

net.trainParam.time-最大训练时间(缺省为inf)

net.trainParam.mc-动量因子(缺省为0.9)

……..

net常用调整参数(有自适应lr的梯度下降法:*不要求)

net.trainParam.epochs-最大训练次数(缺省为10)

……

net.trainParam.lr_inc-学习速率lr增长比(缺省为1.05)

net.trainParam.lr_dec-学习速率lr下降比(缺省为0.7)

net.trainParam.max_perf_inc-性能函数增加最大化(缺省为1.04)

net.trainParam.mc-动量因子(缺省为0.9)

……

net常用调整参数(量化共轭梯度法:*不要求):

net.trainParam.epochs-最大训练次数(缺省为10)

net.trainParam.goal-训练的目标误差(缺省为0)

net.trainParam.lr-学习速率(缺省为0.01)

net.trainParam.show-显示迭代过程(NaN表示不显示，缺省为25)

net.trainParam.time-最大训练时间(缺省为inf)

net.trainParam.sigma-因二次求导对权值的影响参数(缺省为5.0e-5)

net.trainParam.sigma-Hessian矩阵不确定性调节参数(缺省为5.0e-7)

……..

BP网络举例1：利用BP神经网络做预测。

已知某商品的销售情况表(表1)，现创建一个三层BP神经网络对商品销售情况做预测：输入层结点数为3，隐含层结点数为5(传输函数使用tansig)，输出层结点数为1(传输函数使用logsig)，试用此网络编写一个商品销售量预测程序。

表1 商品销售情况表

月份	1	2	3	4	5	6
销量	2056	2395	2600	2298	1634	1600
月份	7	8	9	10	11	12
销量	1873	1478	1900	1500	2046	1556

解：将表1按下面方法进行归一化处理，得到表2。

线性归一化处理方法：$x_{new} = \dfrac{x - x_{\min}}{x_{\max} - x_{\min}}, (x_{new} \in [0,1])$

$x_{\max}$、$x_{\min}$分别为样本的最大值和最大值，

x、x_{new}分别为转换前后的值。

表2 商品销售归一化表

月份	1	2	3	4	5	6
销量	0.5152	0.8173	1.0000	0.7308	0.1390	0.1087
月份	7	8	9	10	11	12

关于归一化处理的几点说明：

(1)归一化的作用是归纳统一样本的统计分布。归一化在0~1之间的是统计的概率分布；归一化在-1~1之间的是统计的坐标分布。归一化具有同一、统一和合一的意思。无论是建模还是计算，首先是基本度量单位的量纲要统一。

(2)在进行神经网络训练前，必须对输入样本和输出样本进行归一化处理。目的是使归一化的输入值和输出值较均匀地落在[0,1]区间。

(3)特定的样本必须采用特定的归一化处理方法，而不是仅采用线性归一化处理这一种方法。

训练时，需要对样本进行归一化处理。但如果是仿真，需要对仿真后的结果进行反归一化处理。

BP网络销售量预测MATLAB程序清单：

```
%Filename:ex9_1.m
clear all;      %清除所有变量
%以每三个月的商品销售量经归一化处理后作为输入
Xp=[0.5152  0.8173  1.0000;   %1-3月数据作模式输入
  0.8173   1.0000  0.7308;    %2-4月数据作模式输入
  1.0000   0.7308  0.1390;    %3-5月数据作模式输入
  0.7308   0.1390  0.1087;    %4-6月数据作模式输入
  0.1390   0.1087  0.3520;    %5-7月数据作模式输入
  0.1087   0.3520  0.0000]';  %6-8月数据作模式输入
%以第四个月的销售量归一化处理后作为目标向量(4-9月数
%据作目标向量)
T=[0.7308 0.1390 0.1087 0.3520 0.0000 0.3761];
```

```
%创建BP神经网络，输入向量的各特征取值范围为[0 ,1]，
%隐含层有5个神经元，输出层有1个神经元，隐含层传输
%函数为tansig，输出层传输函数为logsig，
%训练函数为梯度下降函数traingd，即采用基本的BP
%学习算法
net=newff([0 1;0 1;0 1],[5,1],{'tansig','logsig'},'traingd');
net.trainParam.epochs=20000;
net.trainParam.goal=1e-3;
%设置学习速率为0.1
net.trainParam.lr=0.1;
```

```
net=train(net,Xp,T);
Yp=sim(net,Xp);
m=length(Yp);
for i=1:m
    Yp(i)=round(Yp(i)*(2600-1478)+1478)
%反归一化、四舍五入取整，得到预测结果
end
```

```
x=[1873 1478 1900]‘ ;
%给定某个连续三个月的商品销售量模式向量
m=size(x);
for i=1:m
    x(i)=(x(i)-1478)/(2600-1478);
%模式向量元素归一化
end
y=sim(net,x);
y=round(y*(2600-1478)+1478)
%反归一化、四舍五入取整，得某月商品销售量预测结果
```

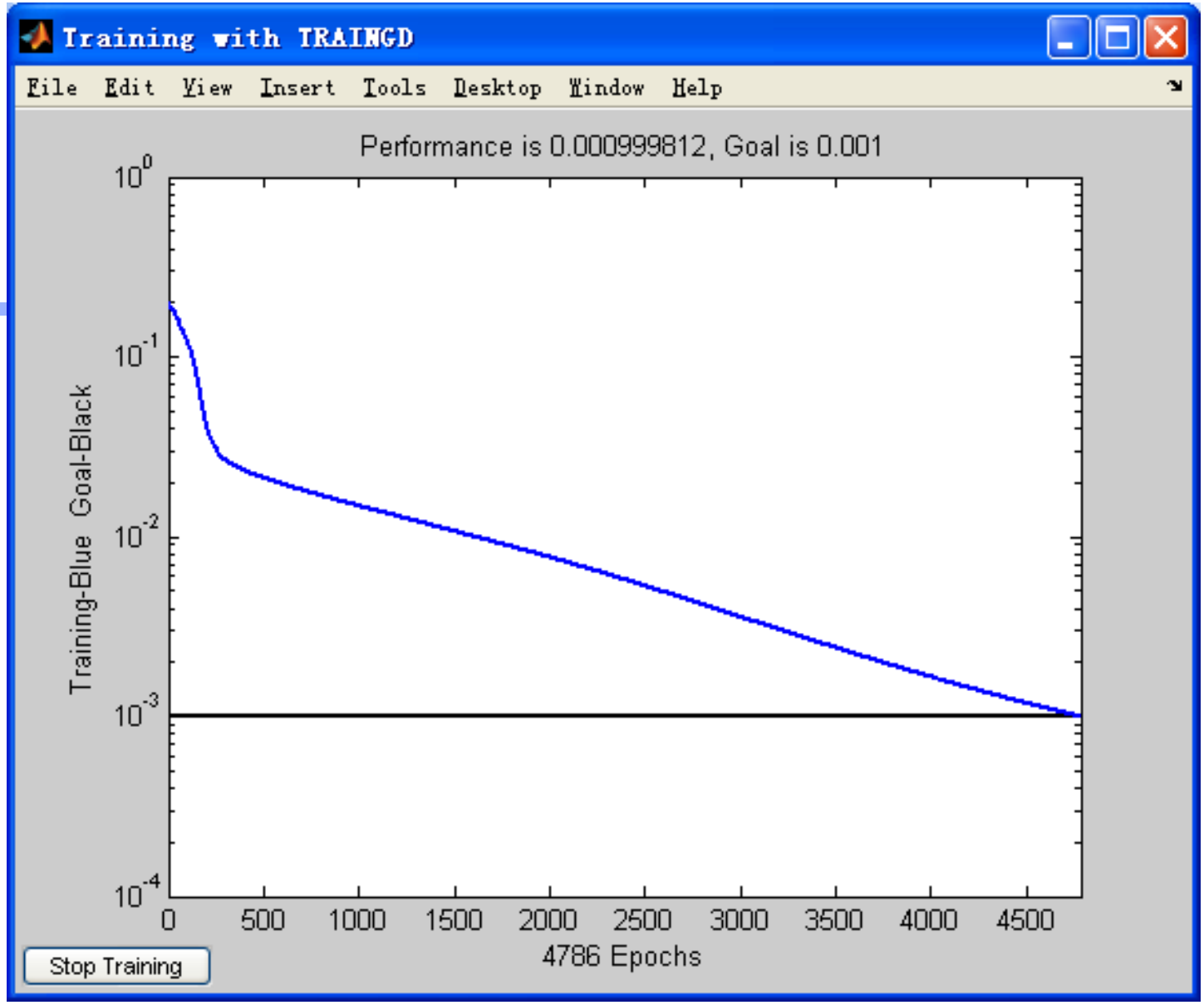

Training with TRAINGD
File Edit View Insert Tools Desktop Window Help
Performance is 0.000999812, Goal is 0.001
Training-Blue Goal-Black
10^0
10^-1
10^-2
10^-3
10^-4
0 500 1000 1500 2000 2500 3000 3500 4000 4500
4786 Epochs
Stop Training

图 训练误差曲线

本程序运行后命令行窗口显示结果：

……

TRAINGD, Epoch 4775/20000, MSE 0.00100617/0.001, Gradient 0.00241053/1e-010

TRAINGD, Epoch 4786/20000, MSE 0.000999812/0.001, Gradient 0.00239887/1e-010

TRAINGD, Performance goal met.

Yp =

2290 1625 1635 1856 1554 1889

4-9份月销售量预测值

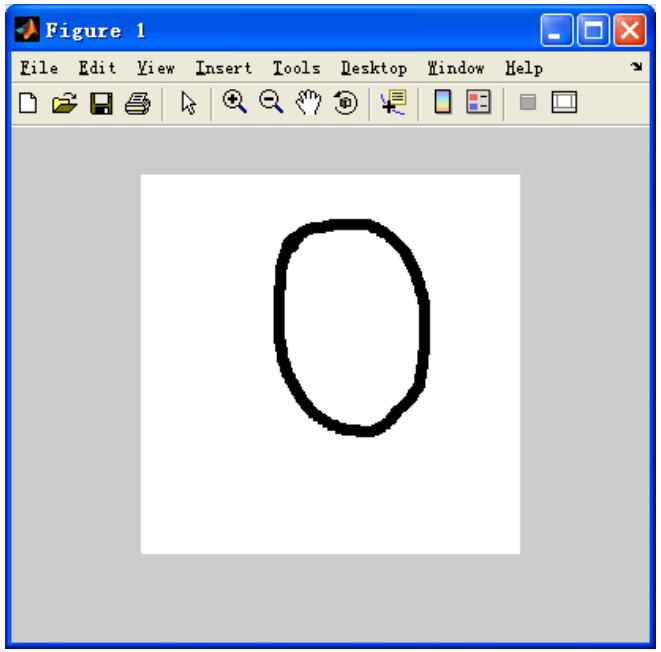
y =

1515

由7、8、9三个月模式样本预测出第10个月的销售量

从上面结果可知， 预测结果与表1中给出的实际结果存在一定的误差，此误差可以通过增加训练次数和提高目标误差精度进一步缩小。

BP网络举例2：试采用BP神经网络设计并编写识别0~9数字的MATLAB程序。



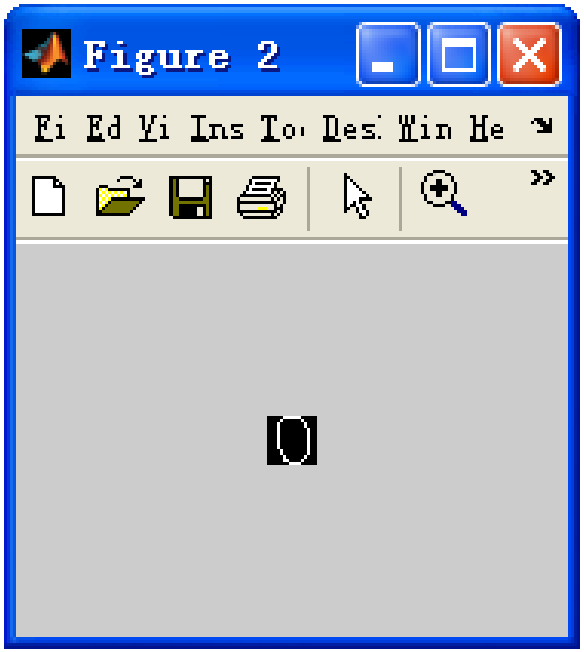


图　240x240大小的某数字0的原图像及其对应的16x16二值图像

BP网络数字识别MATLAB程序清单：

```
%Filename:ex9_2.m
%BP网络数字识别
%加载数字0-9图像文件(每个数字有10个文件，共100个文件)
%取出图像的最大有效区域，归一化(Normalize)为16×16的二值图像
%16×16的二值图像按行顺序排列得到该数字的模式特征向量(16×16=256维)
%生成数字0-9的模式特征向量和目标向量(每个数字有10个，共100个)
clc;                %清除命令窗口
clear all;          %清除内存变量
'Loading ......'
```

```
for kk=0:99
    p1=ones(16,16);
    m=strcat('C:\Documents and Settings\Administrator\桌面\PR_Student\lecture9\nums\',int2str(kk),'.bmp');
    x=imread(m);
    %将灰度图像转换成黑白图像
    bw=im2bw(x,0.5);     %设threshold=0.5,将亮度大于threshold的像素位置置1(白色)，亮度小于threshold的像素位置置为0(黑色)
    [i,j]=find(bw==0);   %检索矩阵bw中等于0的元素的行下标和列下标
```

```
imin=min(i);
imax=max(i);
jmin=min(j);
jmax=max(j);
bw1=bw(imin:imax,jmin:jmax);
rate=16/max(size(bw1));
bw1=imresize(bw1,rate);    %调整图像大小
[i,j]=size(bw1);
i1=round((16-i)/2);
j1=round((16-j)/2);
p1(i1+1:i1+i,j1+1:j1+j)=bw1;
p1=-1.*p1+ones(16,16);    %归一化后的16×16大小二值图像
```

```
for m=0:15
    p(m*16+1:(m+1)*16,kk+1)=p1(1:16,m+1);%将16×16大小二值图像转成256维BP网络输入特征向量
  end
  switch kk
    case{0,10,20,30,40,50,60,70,80,90}
      t(kk+1)=0;
    case{1,11,21,31,41,51,61,71,81,91}
      t(kk+1)=1;
    case{2,12,22,32,42,52,62,72,82,92}
      t(kk+1)=2;
    case{3,13,23,33,43,53,63,73,83,93}
      t(kk+1)=3;
```

```
case{4,14,24,34,44,54,64,74,84,94}
        t(kk+1)=4;
    case{5,15,25,35,45,55,65,75,85,95}
        t(kk+1)=5;
    case{6,16,26,36,46,56,66,76,86,96}
        t(kk+1)=6;
    case{7,17,27,37,47,57,67,77,87,97}
        t(kk+1)=7;
    case{8,18,28,38,48,58,68,78,88,98}
        t(kk+1)=8;
    case{9,19,29,39,49,59,69,79,89,99}
        t(kk+1)=9;
  end
end
```

```
'Load Ok.'
save ex9_2pt p t; %将手写数字0-9的模式特征矩阵p和目标向量t存为二进制文件ex9_2pt.mat
%创建和训练BP神经网络
clear all;
load ex9_2pt p t; %将二进制文件ex9_2pt.mat的内容读入内存(读入模式特征矩阵和目标向量)
pr(1:256,1)=0;
pr(1:256,2)=1;
net=newff(pr,[25 1],{'logsig','purelin'},'traingd');
```

```
net.trainParam.epochs=15000;
net.trainParam.goal=0.001;
net.trainParam.show=10;
net.trainParam.lr=0.05;
net=train(net,p,t);
save ex9_2net net; %将创建的神经网络net存为二进制文件ex9_2net.mat
```

```
for times=0:99
    clear all;
    p(1:256,1)=1;
    p1=ones(16,16);
    load ex9_2net net;  %读入神经网络
    test=input('请输入待识别的完整数字文件名：','s');
    x=imread(test);
    figure (1);
    imshow(x); %显示某数字的原图像
```

%将灰度图像转换成黑白图像

bw=im2bw(x,0.5); %设threshold=0.5,将亮度大于threshold的像素位置置1(白色)，亮度小于threshold的像素位置置为0(黑色)

[i,j]=find(bw==0); %检索矩阵bw中等于0的元素的行下标和列下标

imin=min(i);

imax=max(i);

jmin=min(j);

jmax=max(j);

```
bw1=bw(imin:imax,jmin:jmax);
rate=16/max(size(bw1));
bw1=imresize(bw1,rate);
[i,j]=size(bw1);
i1=round((16-i)/2);
j1=round((16-j)/2);
p1(i1+1:i1+i,j1+1:j1+j)=bw1;
p1=-1.*p1+ones(16,16);
```

```
    figure (2);
    imshow(p1);    %显示原数字图像的16x16大小特征图像
    for m=0:15
       p(m*16+1:(m+1)*16,1)=p1(1:16,m+1);
    end
    y=sim(net,p);
    y=round(y)
end
```

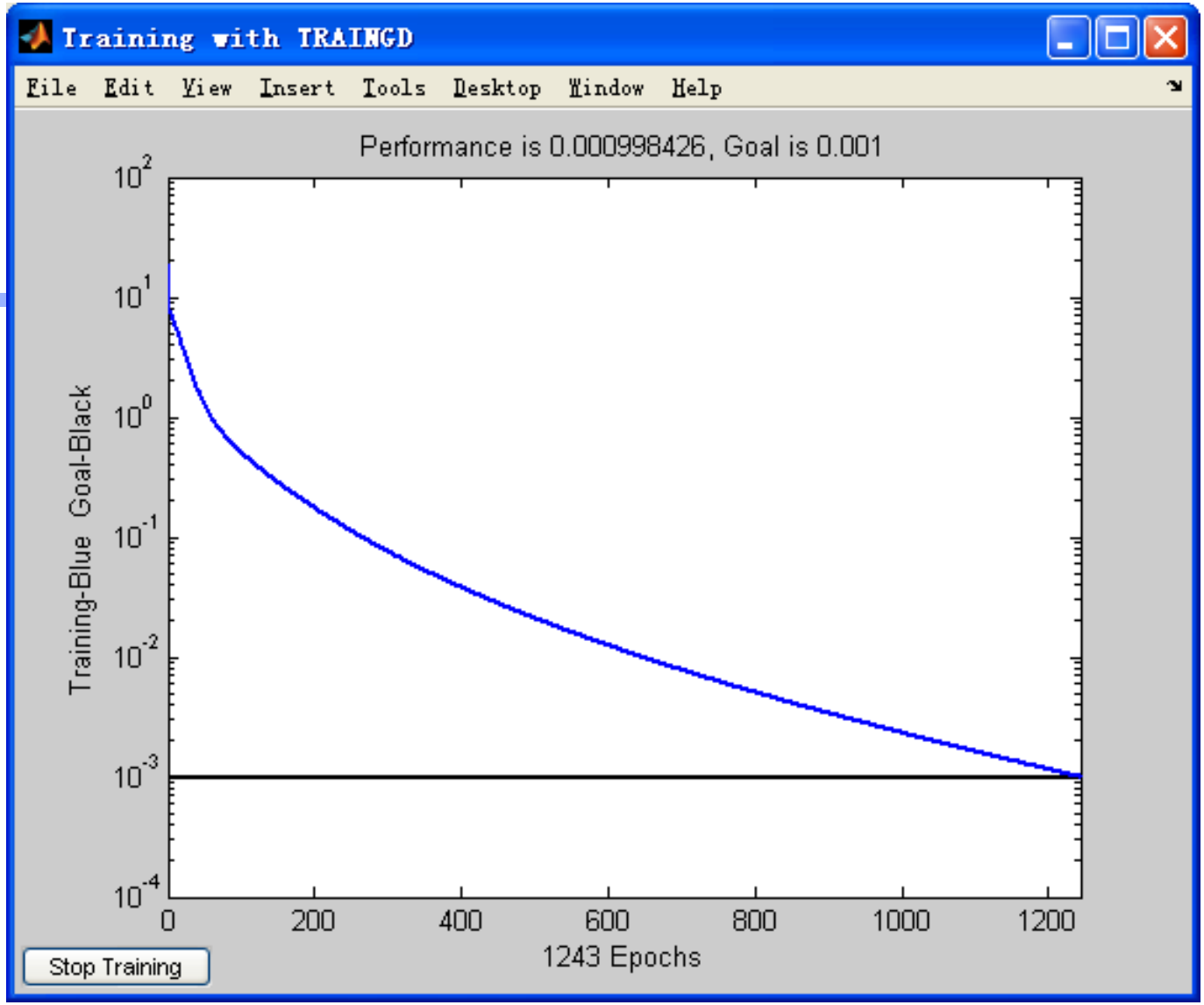

Training with TRAINGD
File Edit View Insert Tools Desktop Window Help
Performance is 0.000998426, Goal is 0.001
Training-Blue Goal-Black
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
200
400
600
800
1000
1200
1243 Epochs
Stop Training

图 训练误差曲线

……
TRAINGD, Epoch 1230/15000, MSE 0.00104294/0.001, Gradient 0.00837013/1e-010
TRAINGD, Epoch 1240/15000, MSE 0.00100851/0.001, Gradient 0.00821917/1e-010
TRAINGD, Epoch 1243/15000, MSE 0.000998426/0.001, Gradient 0.00817453/1e-010
TRAINGD, Performance goal met.

请输入待识别的完整数字文件名：0.bmp

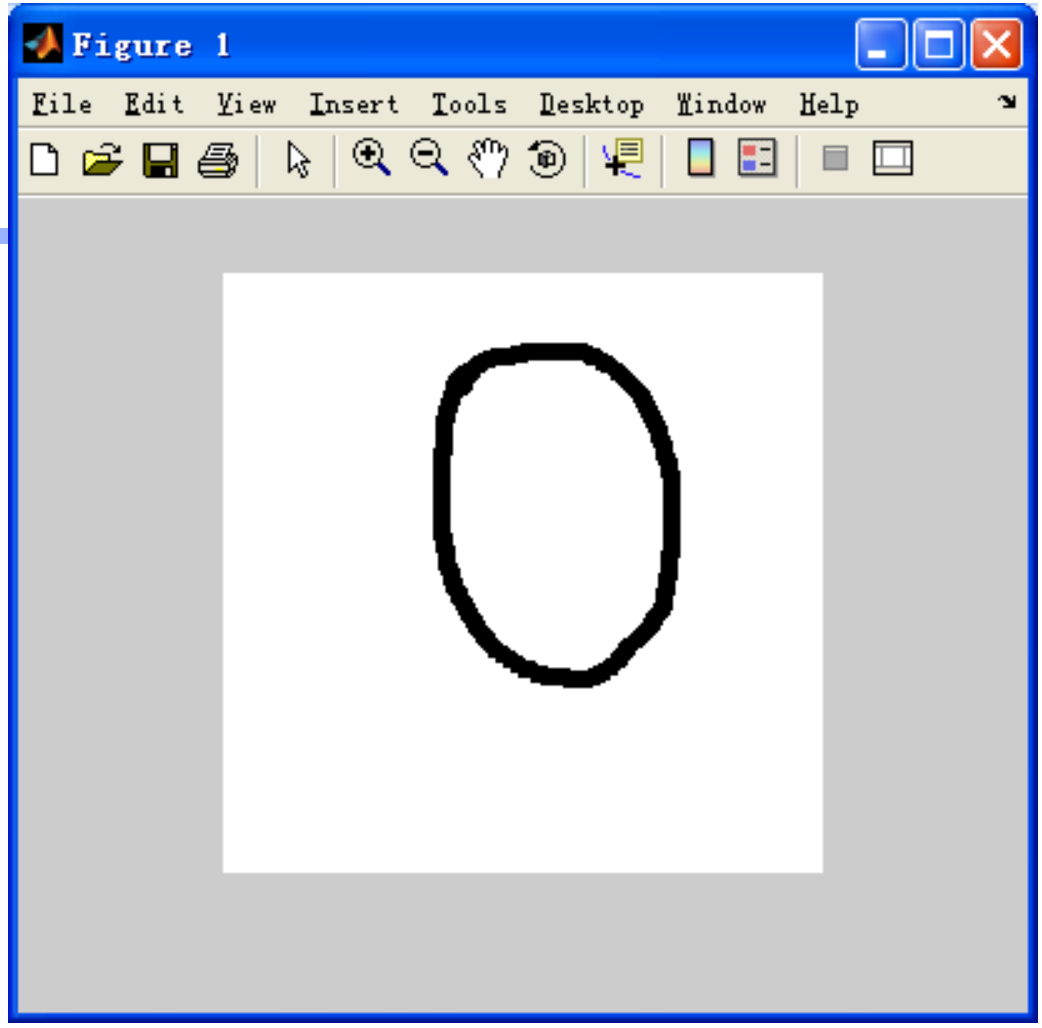


图　显示加载的数字0的原图像(240x240大小)

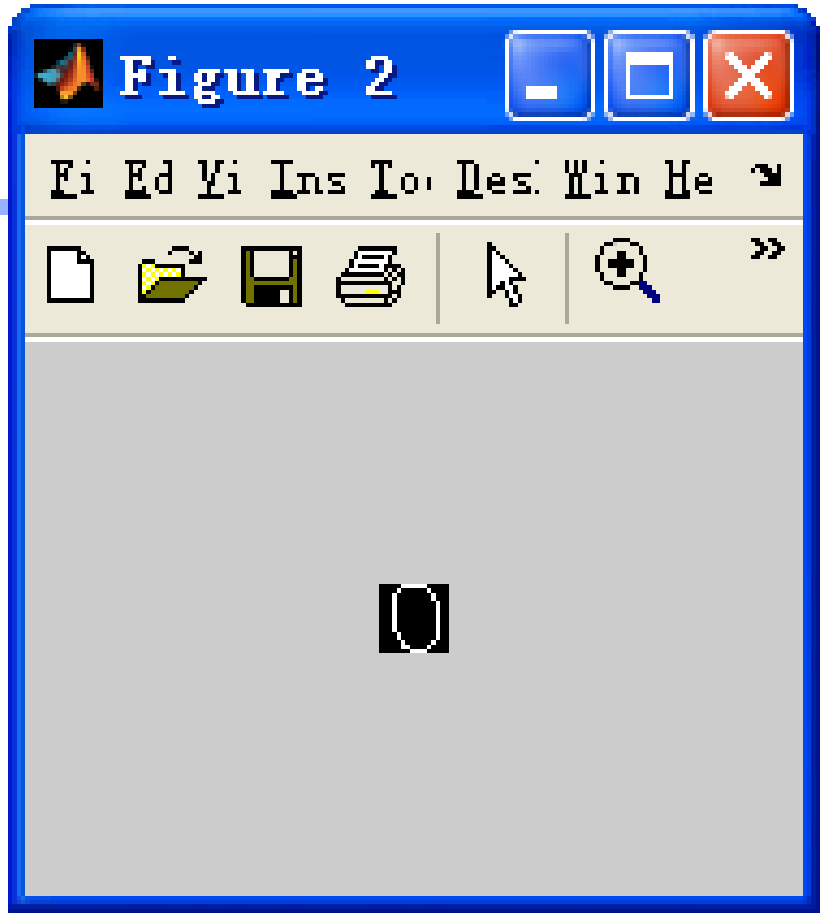


图　显示加载的数字0对应的16x16二值图像

数字0识别结束时命令窗口显示结果：

```
y =

     0
```

请输入待识别的完整数字文件名：输入下一个待识别的文件名

(略)

说明：本例是一个基本的数字识别程序，它能对印刷体或手写体数字进行识别。在此基础上，读者可对其进行修改、完善甚至是拓展，如对算法设计进行改进以提高识别率以及增加字符的识别功能等。

八.隐含层神经元数目的讨论

BP网络隐含层神经元数目对网络性能影响很大，因此需要适当地选取。

1988年Cybenko指出，当各神经元均采用S型函数时，一个隐含层就足以解决任意分类问题；两个隐含层则足以实现输入的任意函数；在线性可分情况下，不需要隐含层(即采用感知器即可)。

设n、m、h分别表示输入节点数(输入向量维数)、输出神经元数目和隐含神经元数，**N**为训练样本数。

1.单隐含层

(1)$h = 2n + 1$;

(2)对于医疗诊断神经网络系统，通常n较大，

$h_{\max} = m(n+1)$或$h_{\max} = 3m$ ；

(3)对于$n > m$小型神经网络系统；$h = \sqrt{n \cdot m}$ ；

(4)在图像识别中，当输入维数较多时，$h_{\min} = 0.02 \times n$ ；

(5) 当用于统计过程控制时，$h = 4n$ ；

(6)当用于数据压缩时，$h = log(2n)$ ；

(7)$h = \sqrt{n+m} + a, a \in [1,10]$。

2.双(多)隐含层

(1)用于图像识别时，第二隐含层神经元数目$h=2m$；

(2)当为高维输入向量时，第一隐含层对第二隐含层的比例为3:1。

3.隐含神经元数目与训练样本数的关系

(1)$h=\log_2 \mathrm{N}$；

(2)训练次数最小的隐含层最佳神经元数目$h\approx N-1$。

上面是从一些文献中收集的确定隐含层神经元数目的经验性结论，供同学们设计神经网络模型时参考。需要特别强调的是，由于应用的目的和条件不同，所得结果彼此有所不同。根据本人经验，在解决实际工程应用问题时，首先是选择已有的经验公式大致确定隐含层神经元数目，然后在实验中反复对隐含层神经元数目进行调整，以使神经网络达到良好的工作性能。

6.4 Hopfield神经网络模型(：*了解)

Hopfield神经网络是1982年美国物理学家J.Hopfield提出的一种反馈神经网络类型。

与前向型神经网络不同，前向神经网络不考虑输出与输入之间在时间上的滞后影响，其输出与输入之间仅仅是一种映射关系。而Hopfield网络不同，它采用反馈连接，考虑输出与输入在时间上存在延时，它表示的是一个动态过程，需要用差分方程或微分方程描述。因而Hopfield神经网络是一种由非线性零件构成的反馈系统，其稳定状态的分析比前向神经网络要复杂得多。(具体内容，此略)

6.5 神经网络的发展-深度网络简介

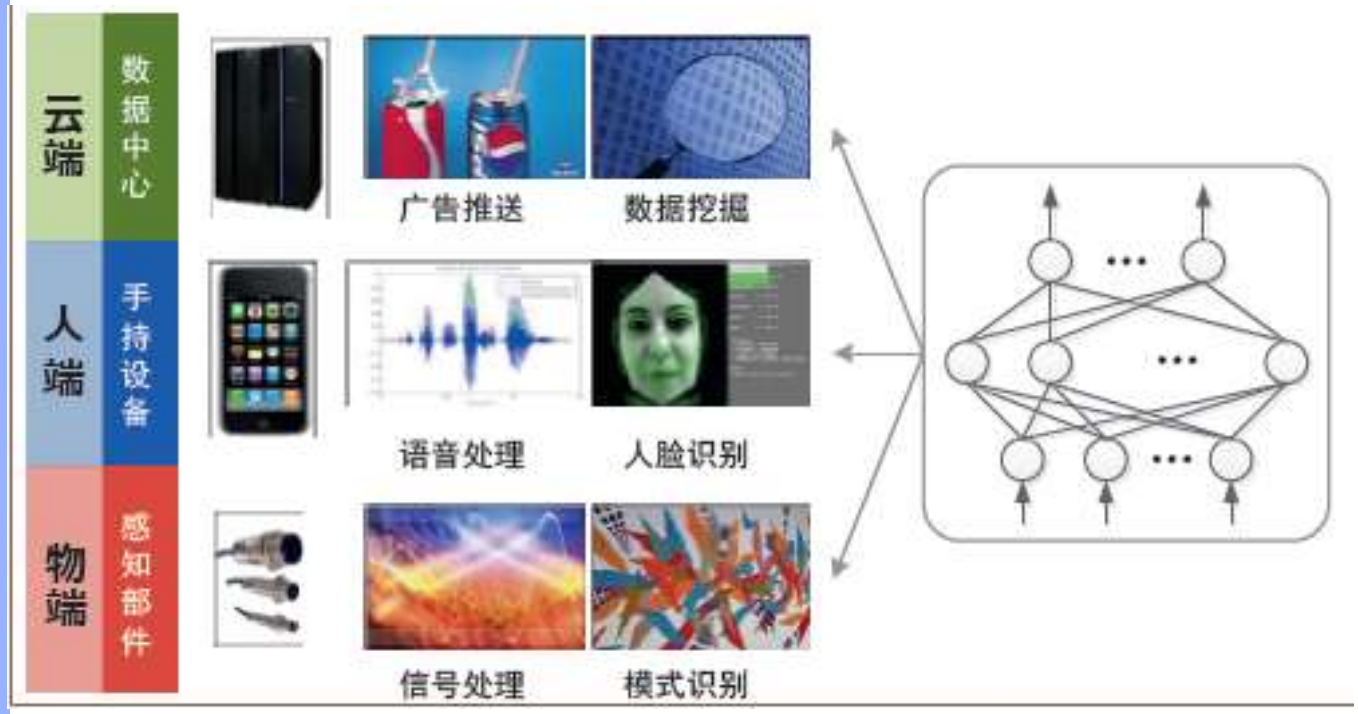


图 人工神经网络的应用覆盖了云端、人端和物端

学术界已经提出了许多不同的人工神经网络结构模型和算法。这些方法在对很多图像、声音和文字的理解和分析上表现出了很好的性能。无论是云端的大数据处理（如广告推荐、数据挖掘和视频图像自动标记），还是个人移动终端任务（如语音识别和自动翻译），都能看到人工神经网络的身影。

一、深度神经网络简介

以深度信念网络(Deep Belief Network, DBN) 和卷积神经网络(Convolutional Neural Network, CNN) 为代表的“深度学习”（Deep Learning, 一种多层的神经网络），更是当前学术界和工业界流行的名词。如，Google、微软、科大讯飞、百度利用深度神经网络在语音识别和图像处理方面做了许多有价值的工作，现已经进入产品化阶段。2011年，谷歌和斯坦福大学合作构建了一个有10 亿个突触(Synapse)的深度神经网络来进行猫脸识别。他们用16000个处理器核花了好几天时间训练此神经网络，使得其对猫脸识别的准确率比前人最好的结果提高了70%。

二、深度学习方法

2006 年，辛顿(Hinton) 等人提出了深度学习方法。深度学习方法在传统的人工神经网络训练中增加了一个预训练阶段，即用无监督学习方法对每层网络进行专门训练，然后用有监督学习方法对整个网络进行总体训练。通过深度学习方法，人工神经网络的很多应用效果赶上甚至显著超过了支持向量机(关于SVM，见教材P196 9.4节)等其他机器学习方法，IBM、谷歌、微软、科大讯飞、百度等公司在很多工业级图像和语音处理应用上取得了非常好的成果。既然人工神经网络重新成为最有效的认知任务处理算法（至少是之一），专门的神经网络计算机当然对认知应用有很高的价值。

6.6 神经网络处理器芯片简介

●机器学习芯片：IBM“类人脑”神经网络处理器芯片，处理器集成了54亿个晶体管，是人类当代集成度最高的智能IC芯片。

关键字：类人脑芯片，SyNAPSE，
人工神经网络处理器

目前，世界最像人脑的计算机芯片一直由IBM所主导开发，IBM在Cornell Univ.等单位的携手合作下，为美国国防部先进研究计划局(DARPA: Defense Advanced Research Projects Agency)的神经形态自适应塑料可微缩电子系统(SyNAPSE[1-2]: System of Neuromorphic Adaptive Plastic Scalable Electronics)打造出一种先进的“类人脑芯片”。研究团队利用试制出的半导体芯片，成功完成了对人物图像等的识别工作，而功耗仅70mW。

------A million spiking-neuron integrated circuit with a scalable communication network and interface, Science 8 Aug. 2014

Abstract: Inspired by the brain’ s structure, we have developed an efficient, scalable, and flexible non–von Neumann architecture that leverages contemporary silicon technology. To demonstrate, we built a 5.4-billion-transistor chip with 4096 neurosynaptic cores interconnected via an intrachip network that integrates 1 million programmable spiking neurons and 256 million configurable synapses. Chips can be tiled in two dimensions via an interchip communication interface, seamlessly scaling the architecture to a cortexlike sheet of arbitrary size. The architecture is well suited to many applications that use complex neural networks in real time, for example, multiobject detection and classification. With 400-pixel-by-240-pixel video input at 30 frames per second, the chip consumes 63 milliwatts.

---- Science 8 Aug., Pages: 668-672, 2014

这里我们再回顾一下生物神经网络：人脑大约有1000亿(10^{11})个神经元，这些神经元通过1000万亿(10^{15})个连接构成一个大规模的神经网络系统。神经元是神经网络的基本信息处理单元。

生物神经元的基本组成：

a. 细胞体(soma/cell body)；

b. 树突(dendrite)；

c. 轴突(axon)；

d. 突触(synapse)。

SyNAPSE“类人脑”芯片研究计划由IBM院士兼IBM研究院脑启发运算研究中心首席科学家Dharmendra Modha牵头，该研究计划由DARPA提供赞助总经费约5300万美元，于2008年正式启动研究计划，分为四个阶段进行。

SyNAPSE芯片拥有1百万个神经元(类脑细胞)和2.56亿个突触synapse(储存单元)，以及4096个称为“神经突触”(neurosynaptic)的处理核心执行作业，并整合内存、运算、通信，以一种异步事件驱动、平行与容错的方式作业。SyNAPSE是至今国际上最大型的芯片，但功耗低到仅70mW。

为了衡量该款庞大芯片的性能，IBM发明了一种新的度量标准—每秒突触运算(Synaptic Operations Per Second；SOPS)，以取代每秒浮点运算(FLOPS)性能。该芯片能提供每瓦460亿的SOPS，仅相当于一张邮票大小的超级计算机芯片重量却轻如羽毛，所需的电源也仅约一款助听器的用量。

思考题：

1.画出人工神经元示意图，并写出数学表达式。

2.人工神经网络分为哪4种类型？

3.Hebb学习规则和δ学习规则的内容分别是什么？

4.简述基本的BP算法思想。

5.BP网络神经元的传输函数有什么特殊要求？