第3章 神经网络的通用函数

神经网络工具箱是在 MATLAB 环境下开发出来的许多工具箱之一。它以人工神经网络理论为基础，利用 MATLAB 编程语言构造出许多典型神经网络的框架和相关的函数。这些工具函数主要分为两大部分：一部分函数是特别针对某一种类型的神经网络的，如感知器的创建函数、BP 网络的训练函数等；另外一部分函数则是通用的，几乎可以用于所有类型的神经网络，如神经网络仿真函数、初始化函数和训练函数等。本章主要介绍神经网线的通用函数，其他函数将在后面章节逐一介绍。

在 MATLAB 的命令窗口中输入 help nnet，即可得到神经网络工具箱的有关版本信息及函数列表。

```
>> help nnet
  Neural Network Toolbox
  Version 8.4 (R2015b) 13-Aug-2015        % 版本信息

  Graphical user interface functions.      % GUI 函数
    nnstart       - Neural Network Start GUI
    nctool        - Neural Cassification app
    nftool        - Neural Fitting app
    nntraintool   - Neural Network Training Tool
    nprtool       - Neural Pattern Recognition app
    ntstool       - Neural Time Series app
    nntool        - Neural Network Toolbox graphical user interface
    view          - View a neural network.

    Network creation functions.            % 网络创建函数
    cascadeforwardnet - Cascade-forward neural network.
    competlayer       - Competitive neural layer.
    distdelaynet      - Distributed delay neural network.
    elmannet          - Elman neural network.
    feedforwardnet    - Feed-forward neural network.
    fitnet            - Function fitting neural network.
    layrecnet         - Layered recurrent neural network.
    linearlayer       - Linear neural layer.
    lvqnet       - Learning vector quantization (LVQ) neural network.
    narnet       - Nonlinear auto-associative time-series network.
```

```
  narxnet       - Nonlinear auto - associative time - series network with external input.
  newgrnn           - Design a generalized regression neural network.
  newhop            - Create a Hopfield recurrent network.
  newlind           - Design a linear layer.
  newpnn            - Design a probabilistic neural network.
  newrb             - Design a radial basis network.
  newrbe            - Design an exact radial basis network.
  patternnet        - Pattern recognition neural network.
  perceptron        - Perceptron.
  selforgmap        - Self - organizing map.
  timedelaynet      - Time - delay neural network.

Using networks.                                  % 网络仿真及初始化函数
  network     - Create a custom neural network.
  sim         - Simulate a neural network.
  init        - Initialize a neural network.
  adapt       - Allow a neural network to adapt.
  train       - Train a neural network.
  disp        - Display a neural network's properties.
  display     - Display the name and properties of a neural network.
  adddelay    - Add a delay to a neural network's response.
  closeloop   - Convert neural network open feedback to closed feedback loops.
  formwb      - Form bias and weights into single vector.
  getwb       - Get all network weight and bias values as a single vector.
  noloop      - Remove neural network open and closed feedback loops.
  openloop    - Convert neural network closed feedback to open feedback loops.
  removedelay - Remove a delay to a neural network's response.
  separatewb  - Separate biases and weights from a weight/bias vector.
  setwb       - Set all network weight and bias values with a single vector.

Simulink support.                                % Simulink 支持函数
  gensim      - Generate a Simulink block to simulate a neural network.
  setsiminit  - Set neural network Simulink block initial conditions.
  getsiminit  - Get neural network Simulink block initial conditions.
  neural      - Neural network Simulink blockset.

Training functions.                              % 网络训练函数
  trainb      - Batch training with weight & bias learning rules.
  trainbfg    - BFGS quasi - Newton backpropagation.
  trainbr     - Bayesian Regulation backpropagation.
  trainbu     - Unsupervised batch training with weight & bias learning rules.
  trainbuwb   - Unsupervised batch training with weight & bias learning rules.
  trainc      - Cyclical order weight/bias training.
  traincgb    - Conjugate gradient backpropagation with Powell - Beale restarts.
  traincgf    - Conjugate gradient backpropagation with Fletcher - Reeves updates.
  traincgp    - Conjugate gradient backpropagation with Polak - Ribiere updates.
  traingd     - Gradient descent backpropagation.
  traingda    - Gradient descent with adaptive lr backpropagation.
```

```
  traingdm   - Gradient descent with momentum.
  traingdx   - Gradient descent w/momentum & adaptive lr backpropagation.
  trainlm    - Levenberg-Marquardt backpropagation.
  trainoss   - One step secant backpropagation.
  trainr     - Random order weight/bias training.
  trainrp    - RPROP backpropagation.
  trainru    - Unsupervised random order weight/bias training.
  trains     - Sequential order weight/bias training.
  trainscg   - Scaled conjugate gradient backpropagation.

Plotting functions.                          % 网络绘图函数
  plotconfusion   - Plot classification confusion matrix.
  ploterrcorr     - Plot autocorrelation of error time series.
  ploterrhist     - Plot error histogram.
  plotfit         - Plot function fit.
  plotinerrcorr   - Plot input to error time series cross-correlation.
  plotperform     - Plot network performance.
  plotregression  - Plot linear regression.
  plotresponse    - Plot dynamic network time-series response.
  plotroc         - Plot receiver operating characteristic.
  plotsomhits     - Plot self-organizing map sample hits.
  plotsomnc       - Plot Self-organizing map neighbor connections.
  plotsomnd       - Plot Self-organizing map neighbor distances.
  plotsomplanes   - Plot self-organizing map weight planes.
  plotsompos      - Plot self-organizing map weight positions.
  plotsomtop      - Plot self-organizing map topology.
  plottrainstate  - Plot training state values.
  plotwb          - Plot Hinton diagrams of weight and bias values.

  % 其他的神经网络实现功能列表函数
Lists of other neural network implementation functions.
  nnadapt         - Adapt functions.
  nnderivative    - Derivative functions.
  nndistance      - Distance functions.
  nndivision      - Division functions.
  nninitlayer     - Initialize layer functions.
  nninitnetwork   - Initialize network functions.
  nninitweight    - Initialize weight functions.
  nnlearn         - Learning functions.
  nnnetinput      - Net input functions.
  nnperformance   - Performance functions.
  nnprocess       - Processing functions.
  nnsearch        - Line search functions.
  nntopology      - Topology functions.
  nntransfer      - Transfer functions.
  nnweight        - Weight functions.
```

3.1 神经网络仿真函数

在 MATLAB 中，提供了 sim()函数用于实现神经网络的仿真。函数的调用格式为：

```
[Y,Pf,Af,E,perf] = sim(net,P,Pi,Ai,T)
```

其中，输入参数 Y 为网络的输出；Pf 表示最终的输入延时状态；Af 表示最终的层延时状态；E 为实际输出与目标向量之间的误差；perf 为网络的性能值；NET 为要测试的网络对象；P 为网络的输入向量矩阵；Pi 为初始的输入延时状态(可省略)；Ai 为初始的层延时状态(可省略)；T 为目标向量(可省略)。

[Y,Pf,Af] = sim(net,{Q TS},Pi,Ai)：参数 Q 为批处理数据的个数，TS 为网络仿真的时间步数。

【例 3-1】 设计一个输入为二维向量的感知器网络，其边界值已定。

```
>> clear all;
net = newp([ - 2 2; - 2 2],1);                    % 创建感知神经网络
% 给权值与阈值赋值
net.IW{1,1} = [ - 1,1];
net.b{1} = [1];
% 下面来看这个感知器网络对两个输入信号的输出如何，两个信号分别位于感知器两个边界
p1 = [1;1];                                       % 第一个输入信号
a1 = sim(net,p1)                                  % 对第一个输入信号仿真
p2 = [1; - 1];                                    % 第二个输入信号
a2 = sim(net,p2)                                  % 对第二个输入信号进行仿真
% 若将两个输入信号组成一个数列，则输出量也为一个数列
p3 = {[1;1],[1; - 1]};
a3 = sim(net,p3)                                  % 对数列进行仿真
```

运行程序，输出如下：

```
a1 =
     1
a2 =
     0
a3 =
    [1]    [0]
```

3.2 神经网络训练函数

MATLAB 提供了几个通用函数用于实现神经网络的训练。下面分别予以介绍。

1. train 函数

在 MATLAB 中，提供了 train 函数用于实现神经网络的训练。函数的调用格式为：

```
[net,tr,Y,E,Pf,Af] = train(net,P,T,Pi,Ai)
```

其中，输出参数 net 为训练后的网络；tr 为训练记录；Y 为网络输出向量；E 为误差向量；Pf 为训练终止时的输入延迟状态；Af 为训练终止时的层延迟状态。输入参数 net 为训练前的网络；P 为网络的输入向量矩阵；T 表示网络的目标矩阵，默认值为 0；Pi 表示初始输入延时，默认值为 0；Ai 表示初始的层延时，默认值为 0；VV 为验证向量(可省略)；TV 为测试向量(可省略)。

网络训练函数是一种通用的学习函数，训练函数重复地把一组输入向量应用到一个网络上，每次都更新网络，直到达到了某种准则，停止准则可能是达到最大的学习步数、最小的误差梯度或误差目标等。

【例 3-2】 创建一个两层前馈网络，并进行训练。

```
>> clear all;
x = [0 1 2 3 4 5 6 7 8];
t = [0 0.84 0.91 0.14 -0.77 -0.96 -0.28 0.66 0.99];
plot(x,t,'o')                          %效果如图 3-1 所示
>> net = feedforwardnet(10);           %创建一个两层前馈网络.该网络有一个隐含层有十元
net = configure(net,x,t);
y1 = net(x)
plot(x,t,'o',x,y1,'x')                 %效果如图 3-2 所示
y1 =
   -1.8260   -0.1460   1.5486   2.1347   3.5209   4.6727   4.3351   3.4458   4.0257
%对网络进行训练
>> net = train(net,x,t);               %效果如图 3-3 所示
y2 = net(x)
>> plot(x,t,'o',x,y1,'x',x,y2,'*')     %效果如图 3-4 所示
y2 =
    0.0000   0.4179   0.8690   0.1400   -0.7700   -0.9600   -0.2800   0.6600   0.9900
```

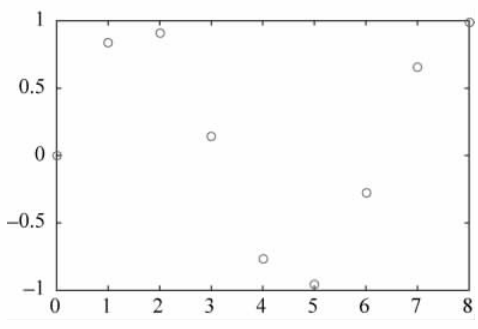

图 3-1 散点图

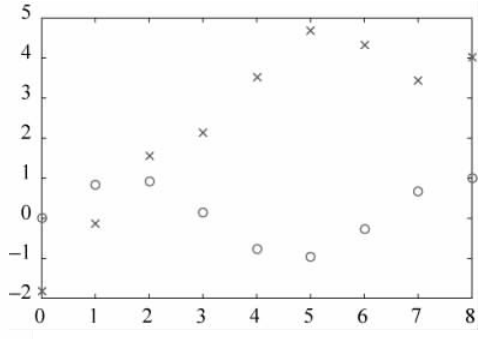

图 3-2 包含十个隐含神经元散点图

2. trainb 函数

在 MATLAB 中，提供了 trainb 用于神经网络权值和阈值的训练。函数的调用格式为：

```
net.trainFcn = 'trainb'
```

用于设置神经网络的训练为 trainb 训练。

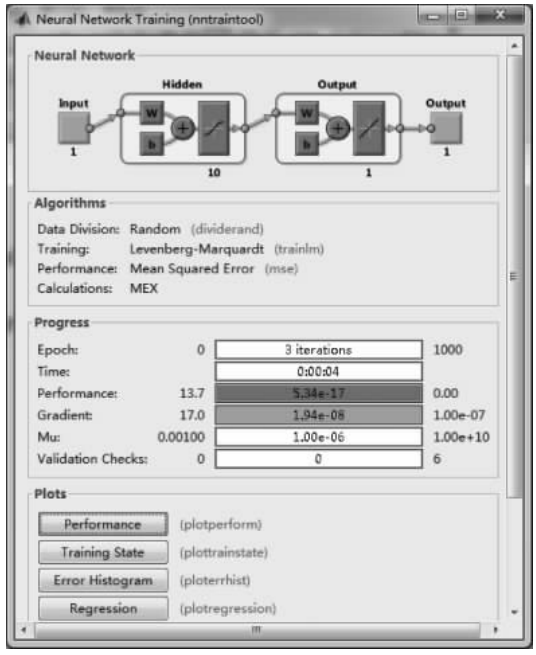

图 3-3 网络训练过程图

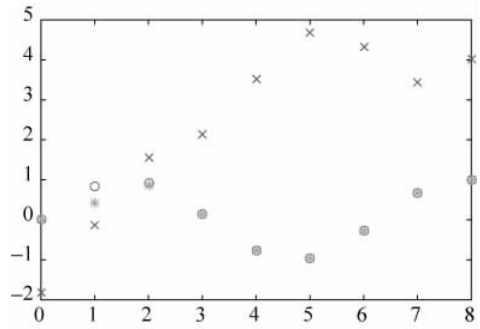

图 3-4 训练后散点图

[net,tr] = train(net,…)：net 为训练后返回的神经网络，tr 为每一步的训练记录。训练后需要设定以下参数，默认值如表 3-1 所示。

表 3-1 训练参数

训练参数名称	默认值	属 性
net. trainParam. epochs	100	最大训练次数
net. trainParam. goal	0	性能参数
net. trainParam. max_fail	6	确认失败的最大次数
net. trainParam. min_grad	1e−6	最低性能梯度
net. trainParam. show	25	两次显示之间的训练步数(无显示时取 NaN)
net. trainParam. showCommandLine	false	生成命令行输出
net. trainParam. showWindow	true	显示训练窗口
net. trainParam. time	inf	最大训练时间(单位，秒)

注意：该函数并不能被直接调用，而是通过函数 train 隐含调用，train 通过设置网络属性 NET. trainFcn 为 trainb 来调用 trainb 对网络进行训练。

【例 3-3】 创建网络，并利用 trainb 函数训练网络。

```
>> clear all;
p = [0 1 2 3 4 5];
t = [0 0 0 1 1 1];
net = feedforwardnet(3,'trainb');
net = train(net,p,t);
y = net(p)                              %仿真
```

运行程序，输出如下，效果如图 3-5 所示。

```
y =
    0.7460    0.4492    0.5497    1.1004    1.1924    0.7741
```

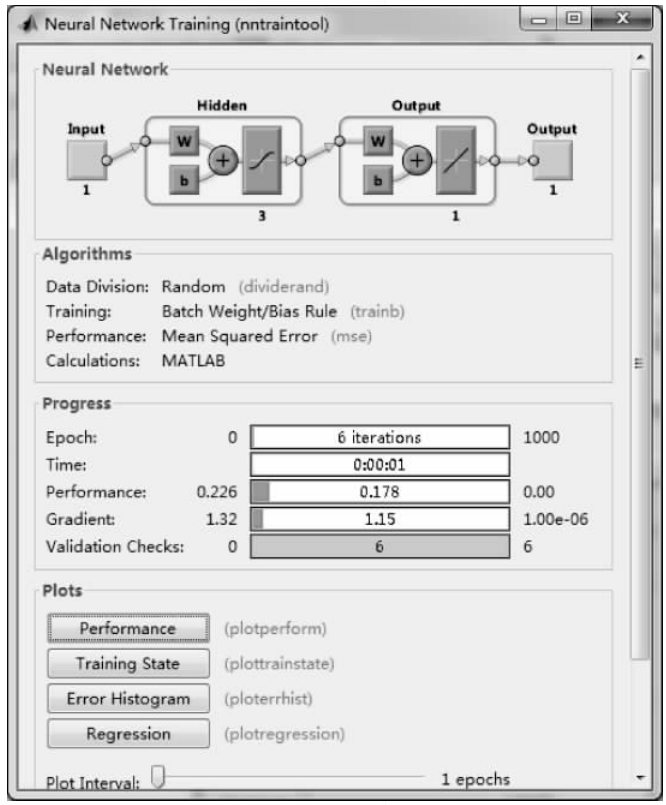

图 3-5　trainb 训练过程图

3. adapt 函数

在 MATLAB 中，提供了 adapt 函数用于对神经网络进行自适应调节。函数的调用格式为：

```
[net,Y,E,Pf,Af] = adapt(net,P,T,Pi,Ai)
```

其中，输入参数 net 为待自适应的神经网络；P 为网络输入；T 为网络目标，默认为 0；Pi 为初始输入延迟，默认为 0；Ai 为初始层延迟，默认为 0；输出 net 参数为自适应后的神经网络；Y 为网络输出；E 为网络误差；Pf 为最终输入延迟；Af 为最终层延迟。

【例 3-4】 对创建的感知器网络进行自适应训练。

```
>> clear all;
p1 = {-1  0 1 0 1 1 -1  0 -1 1 0 1};
t1 = {-1 -1 1 1 1 2  0 -1 -1 0 1 1};
%创建一个线性层，输入范围为[-1,1],0 和 1 的输入延迟，学习率为 0.1
net = linearlayer([0 1],0.1);
[net,y,e,pf] = adapt(net,p1,t1);      %自适应调整
```

```
mse(e)                                    % 平均绝对误差
```

运行程序,输出如下:

```
ans =
    0.7006
```

3.3 神经网络学习函数

在 MATLAB 中,也提供了相关通用学习函数,下面予以介绍。

1. learnp 函数

在 MATLAB 中,提供了 learnp 函数用于对感知器神经网络权值和阈值的学习。函数的调用格式为:

```
[dW,LS] = learnp(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
```

其中,参数 dW 为权值变化矩阵;LS 为当前学习状态;W 为 S×R 的权值矩阵(可省略);P 为 R×Q 的输入向量矩阵;Z 为 S×Q 的输入层的权值矩阵(可省略);N 为 S×Q 的网络输入矩阵(可省略);E 为误差向量(E=T-Y);T 表示网络的目标向量(可省略);A 表示网络的实际输出向量(可省略);gW 为 S×R 的与性能相关的权值梯度矩阵(可省略);gA 为 S×Q 的与性能相关的输出梯度值矩阵(可省略);D 为 S×S 的神经元距离矩阵(可省略);LP 为学习参数(可省略);LS 学习函数声明(可省略);db 为返回阈值调整量;b 为 S×1 的阈值向量;ones(1,Q)为 1×Q 的全为 1 的向量。

在 info = learnp('code')中,针对不同的 code 返回相应的有用信息,包括:

- 当 code=pnames 时,返回函数全称;
- 当 code=pdefaults 时,返回默认的训练参数;
- 当 code=needg 时,如果函数使用了 gW 或 gA,则返回 1。

【例 3-5】 利用 leanrp 函数学习一个感知网络,使其同样能够完成"或"的功能。

```
>> clear all;
err_goal = 0.0015;                        % 设置期望误差最小值
max_epoch = 9999;                         % 设置训练的最大次数
X = [0 1 0 1;0 1 1 0];                    % 样本数据
T = [0 1 1 1];                            % 目标数据
net = newp([0 1;0 1],1);                  % 创建感知器神经网络
net = init(net);                          % 初始化
W = rand(1,2);
b = rand;
net.iw{1,1} = W;
net.b{1} = b;
for epoch = 1:max_epoch
    y = sim(net,X);
    E = T - y;
    sse = mae(E);                         % 计算网络权值修正后的平均绝对误差
```

```
    if(sse < err_goal)
        break;
    end
    dW = learnp(W,X,[],[],[],[],E,[],[],[],[],[]);  % 调整输出层加权系数和偏值
    db = learnp(b,ones(1,4),[],[],[],[],E,[],[],[],[],[]);
    W = W + dW;
    b = b + db;
    net.iw{1,1} = W;
    net.b{1} = b;
end
epoch,W,y
```

运行程序，输出如下：

```
epoch =
     2
W =
    0.2785    0.5469
y =
     0     1     1     1
```

2. learnpn 函数

该函数也是一个权值和阈值学习函数，但它在输入向量的幅值变化非常大或存在奇异值时，其学习速度比 learnp 要快得多。函数的调用格式为：

```
[dW,LS] = learnpn(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
info = learnpn('code')
```

参数含义可参见 learnp 函数说明。

【例 3-6】 定义一个二单元随机输入 P 和误差 E 的三个神经元，并比较用不同学习函数用需时间。

```
>> clear all;
p = rand(2,1);
e = rand(3,1);
tic;
dW1 = learnpn([],p,[],[],[],[],e,[],[],[],[],[])
toc
tic;
dW2 = learnp([],p,[],[],[],[],e,[],[],[],[],[])
toc
```

运行程序，输出如下：

```
dW1 =
    0.1360    0.1447
    0.3349    0.3563
    0.3228    0.3434
时间已过 0.295035 秒。
dW2 =
```

```
    0.1958    0.2083
    0.4822    0.5130
    0.4647    0.4944
时间已过 0.096269 秒。
```

3.4 神经网络初始化函数

在 MATLAB 中，提供了若干通用初始化函数，下面予以介绍。

1. revert 函数

该函数用于将更新后的权值和阈值恢复到最后一次初始化的值。函数的调用格式为：

```
net = revert (net)
```

如果网络结构已经发生了变化，也就是说，如果网络的权值和阈值之间的连接关系，以及输入、每层的长度与原来的网络结构有所不同，那么该函数无法将权值和阈值恢复到原来的值。在这种情况下，函数将权值和阈值都设置为 0。

【例 3-7】 创建一个输入大小为 2，一个神经元的感知器，并用 revert 对更改后的初始值进行恢复处理。

```
>> clear all;
net = perceptron;
net.inputs{1}.size = 2;                    %输入大小为 2
net.layers{1}.size = 1;                    %层数为 1
disp('显示初始化网络权重与阈值: ')
net.iw{1,1}, net.b{1}
%更改以下这些值:
net.iw{1,1} = [1 2];
net.b{1} = 5;
disp('显示更改后的权值与阈值: ')
net.iw{1,1}, net.b{1}
%按如下恢复网络的初始值
net = revert(net);
disp('显示恢复网络的权值与阈值: ')
net.iw{1,1}, net.b{1}
```

运行程序，输出如下：

```
显示初始化网络权重与阈值:
ans =
     0     0
ans =
     0
显示更改后的权值与阈值:
ans =
     1     2
ans =
```

```
     5
显示恢复网络的权值与阈值:
ans =
     0     0
ans =
     0
```

2. init 函数

在 MATLAB 中,init 用于对神经网络进行初始化。函数的调用格式为:

```
net = init(net)
```

其中,返回参数 net 为已经初始化后的神经网络;输入参数 net 为待初始化的神经网络。

【例 3-8】 根据给定的输入向量 P 和目标向量 T,创建一个感知器网络,对其进行训练并初始化。其中,

```
P=[0 1 0 1;0 0 1 1],T=[1 0 0 0]
```

其 MATLAB 实现代码为:

```
>> clear all;
%创建一个感知器网络
P = [0 1 0 1; 0 0 1 1];
T= [0 0 0 1];
net = perceptron;
net = configure(net,x,t);
net.iw{1,1}
net.b{1}
%对感知器进行训练
net = train(net,x,t);
net.iw{1,1}
net.b{1}
%初始化感知器
net = init(net);
net.iw{1,1}
net.b{1}
```

运行程序,输出如下,训练过程如图 3-6 所示。

```
ans =
     0     0
ans =
     0
ans =
     1     2
ans =
    -3
ans =
     0     0
ans =
     0
```

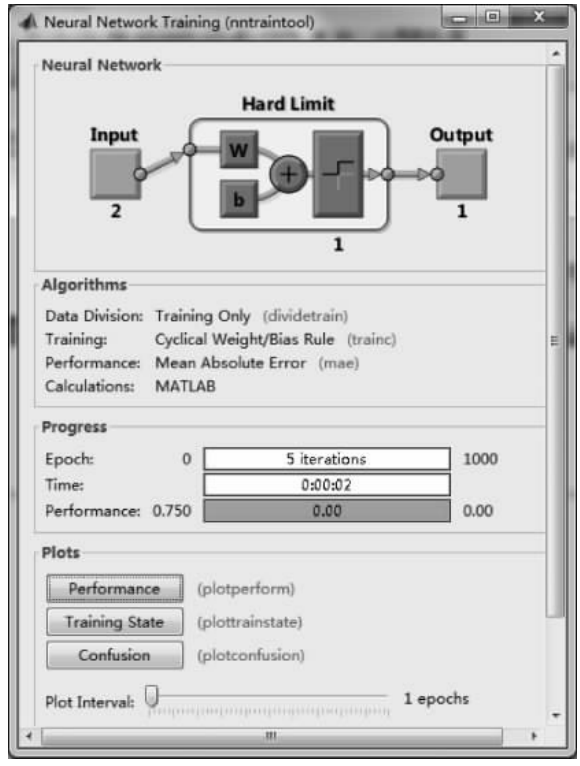

图 3-6　网络的训练过程图

由以上结果可见，在感知器刚刚创建，尚未进行训练以前，权值和阈值都为 0；训练后，权值和阈值分别为[1,2]和－3；对训练好的网络进行初始化后，恢复到以前的值，都为 0。

3. initlay 函数

在 MATLAB 神经网络工具箱中，提供了 initlay 函数用于对线性神经网络的某层进行初始化。函数的调用格式为：

```
net = initlay(net)
```

其中，输入参数 net 为某一层初始化后的网络；输出参数 net 为初始化后得到的网络。

info = initlay('code')：依据 code 值的不同，返回有关函数的信息，包括：

- 当 code＝fpnames 时返回函数参数的名称。
- 当 code＝fpdefaults 时返回默认的函数参数。

4. initnw 函数

该函数是一个层初始化函数，它按照 Nguyen-Widrow 准则对某层的权值和阈值进行初始化。函数的调用格式为：

```
net = initnw(net,i)
```

其中，输入参数 net 为待初始化的神经网络；i 为层次索引；输出参数 net 为初始化后的

神经网络。

3.5 神经网络输入函数

在 MATLAB 中，提供了若干函数用于实现神经网络的输入，下面予以介绍。

1. netsum 函数

该函数是一个输入求和函数，它通过将某一层的加权输入和阈值相加作为该层的输入。函数的调用格式为：

```
N = netsum({Z1,Z2,…,Zn},FP)
```

参数 Z1，Z2，…，Zn 表示第 i 个输入，它的数目可以是任意个，FP 为功能参数单元阵列，可忽略。

info = netsum('code')：依据 code 值的不同，返回不同的信息，包括：

- 当 code＝name 时表示返回传输函数的全称。
- 当 code＝ type 时表示返回输出值的类型。
- 当 code＝fpcheck 时表示返回抛出非法函数参数。
- 当 code＝fullderiv 时返回导数的次数。
- 当 code＝fpnames 时返回函数参数的名称。
- 当 code＝fpdefaults 时返回默认的函数参数。

【例 3-9】 计算给定输入量的加权和。

```
>> clear all;
z1 = [1 2 4; 3 4 1];
z2 = [-1 2 2; -5 -6 1];
b = [0; -1];
n = netsum({z1,z2,concur(b,3)})
```

运行程序，输出如下：

```
n =
     0     4     6
    -3    -3     1
```

2. netprod 函数

与 netsum 的计算框架类似，不过该函数是输入求积函数，它将某一层的权值和阈值相乘作为该层的输入。函数的调用格式为：

```
N = netprod({Z1,Z2,…,Zn})
```

参数 Z1，Z2，…，Zn 表示输入，它的数目可以是任意个。

info = netprod('code')：依据 code 值的不同，返回不同的信息，包括：

- 当 code＝name 时表示返回传输函数的全称。
- 当 code＝deriv 时表示返回微分函数。

- 当 code＝fullderiv 时返回导数的次数。
- 当 code＝fpnames 时返回函数参数的名称。
- 当 code＝fpdefaults 时返回默认的函数参数。

【例 3-10】 计算给定输入量的加权积。

```
>> clear all;
Z1 = [1 2 4;3 4 1];
Z2 = [-1 2 2; -5 -6 1];
Z = {Z1,Z2};
N = netprod({Z})
N =
    [2x3 double]    [2x3 double]
>> N{1}
ans =
     1     2     4
     3     4     1
>> N{2}
ans =
    -1     2     2
    -5    -6     1
```

3. concur 函数

该函数的作用在于使得本来不一致的权值向量和阈值向量的结构一致，以便进行相加或相乘运算。函数的调用格式为：

```
concur(B,Q)
```

其中，B 为 N×1 维的权值向量；Q 为要达到一致化所需要的长度。

返回值为一个已经一致化的矩阵。

【例 3-11】 对两个输入向量使其权值向量与阈值向量结构一致。

```
>> clear all;
>> b = [1; 3; 2; -1];
concur(b,3)
```

运行程序，输出如下：

```
ans =
      1      1      1
      3      3      3
      2      2      2
     -1     -1     -1
```

程序运行结果显示了函数 conur 的运行机理，即修正后的矩阵就是由向量的副本组合而成的。

3.6 神经网络传递函数

传递函数的作用是将神经网络的输入转换为输出，在工具箱中也提供基本函数实现传递功能，下面予以介绍。

1. hardlim 函数

在 MATLAB 中，提供了 hardlim 函数为硬限幅传输函数，可以通过计算网络的输入得到该层的输出。如果网络的输入达到门限，硬限幅函数的输出为 1，否则输出为 0。函数的调用格式为：

```
A = hardlim(N,FP)
```

在给定的网络的输入向量矩阵 N 时，返回该层的输出向量矩阵 A，当 N 中的元素大于等于零时，返回的值为 1，否则为 0；FP 为性能参数(可忽略)。

dA_dN = hardlim('dn',N,A,FP)：返回 A 关于 N 的导数。如果 A 或 FP 为空，则 FP 为默认参数，A 的值依据 N 来计算。

info = hardlim('code')：依据 code 值的不同，返回不同的信息。具体返回内容为：

- 当 code＝name 时即返回传输函数的全称。
- 当 code＝output 时即返回包含有传输函数输出范围最小、最大值的二元向量。
- 当 code＝active 时即返回包含传输函数输入范围最小值、最大值的二元向量。
- 当 code＝fullderiv 时即返回依据 N 返回 1 或 0。
- 当 code＝fpnames 时即返回函数参数的名称。
- 当 code＝fpdefaults 时即返回函数默认参数。

该函数的原型函数为：

$$\text{hardlim}(n)=\begin{cases}1, & n\geqslant 0\\ 0, & n<0\end{cases}$$

【例 3-12】 绘制硬限幅函数的曲线。

```
>> clear all;
n = - 5:0.1:5;
a = hardlim(n);
plot(n,a,'linewidth',3);
```

运行程序，效果如图 3-7 所示。

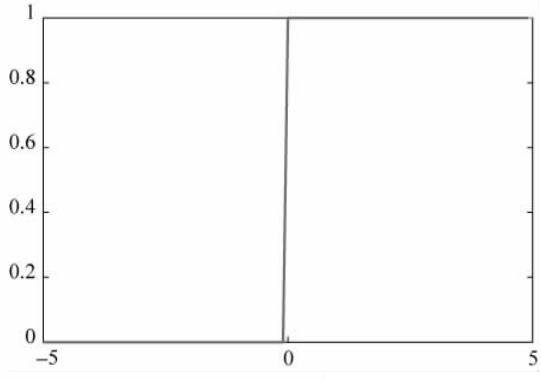

图 3-7 硬限幅曲线

注意：可通过将 NET.layers{i}.transferFcn 设定为 hardlim 来调用该函数，这设置了神经网络中第 i 层的传递函数。

2. hardlims 函数

在 MATLAB 中，提供了 hardlims 函数用于获取对称硬限幅传输函数，也用来计算网线的输入得到该层的输出。对于该函数，如果输入达到门限，则输出值为 1，否则输出值为－1。函数的调用格式为：

```
A = hardlims(N,FP)
dA_dN = hardlims('dn',N,A,FP)
info = hardlims('code')
```

该函数的参数含义与 hardlim 函数的参数含义相同。

该函数的原型函数为：

$$\text{hardlims}(n)=\begin{cases}1, & n\geqslant 0\\ -1, & n<0\end{cases}$$

【例 3-13】 利用 MATLAB 给出一个数组，调用 hardlim 和 hardlims 对其进行分类。

```
>> clear all;
n=-5:0.1:5;                                    % 以 0.1 为步长，建立一个数据
a=hardlim(n);
b=hardlims(n);
plot(n,a,'bo');
hold on;
plot(n,b,'r.');
```

运行程序，效果如图 3-8 所示。

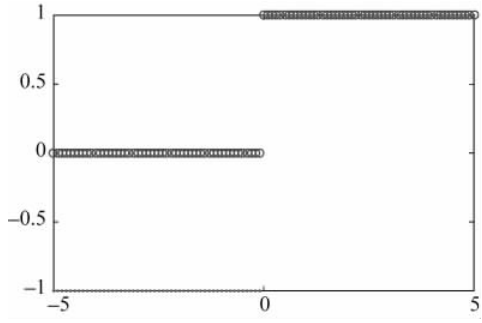

图 3-8 利用传递函数进行分类效果图

图中实点部分是函数 hardlims 的分类结果；圆圈部分是 hardlim 的分类结果。可以看出，两个函数都成功地对数组进行了分类。

3.7 其他函数

在 MATLAB 中，提供了 dotprod 函数用于对权值求点积，它求得权值与输入之间的点积作为加权输入。函数的调用格式为：

```
Z = dotprod(W,P,FP)
```

其中，W 为 S×R 维的权值矩阵；P 为 Q 组 R 维的输入向量；FP 为功能参数的结构，可忽略。

dim = dotprod('size',S,R,FP)：取层的 S 的大小为 size，输入大小为 R，并返回权值维数。

dw = dotprod('dw',W,P,Z,FP)：返回 Z 的导数为 W。

info = dotprod('code')：依据 code 值的不同，返回不同的信息。具体返回内容为：

- 当 code=deriv 时表示返回微分函数。
- 当 code=pfullderiv 时返回导数的次数。
- 当 code=wfullderiv 时返回权值的次数。
- 当 code=name 时返回传输函数的全称。
- 当 code=fpnames 时返回函数参数的名称。
- 当 code=fpdefaults 时返回默认的函数参数。

【例 3-14】 建立两个矩阵(或向量)，计算对应权值的点积。

```
>> clear all;
W = rand(4,3);                    % 建立一个 4×3 的矩阵,所有元素都小于 1
P = rand(3,1);                    % 建立一个 3×1 的矩阵,所有元素都小于 1
Z = dotprod(W,P)
```

运行程序，输出如下：

```
Z =
    0.5148
    1.0951
    0.3571
    0.6322
```