实验报告：基于感知器网络的字母识别

王瑞哲　PB19071509

信息科学技术学院

摘　要：根据感知器网络设计方法，结合 Matlab 程序中 Neural network Toolbox 的程序，设计、编写并实现英文 26 个字母的识别网络（并以此拓展至十个阿拉伯数字或其他字符的识别网络）。为提高网络的识别精度，同时训练无噪声和含有噪声 0.0:0.02:0.2 情况下的网络。之后，测试当被识别的字母在含有噪声随机为 0.0:0.02:0.2 情况下，输入 100 次测试的正确识别率。

关键词：神经网络；感知器网络；Matlab；字母识别；噪声测试

人工神经网络(Artificial Neural Network, 简称 ANN）是由大量简单的处理单元组成的非线性、自适应、自组织系统。它是在现代神经科学研究成果的基础上，试图通过模拟人类神经系统对信息进行加工、记忆和处理的方式，设计出的一种具有人脑风格的信息处理系统。

感知器是 Frank Rosenblatt 在 1957 年所发明的一种人工神经网络。它可以被视为一种最简单形式的前馈式人工神经网络，是一种二元线性分类器。

1 实验目的与实验原理

1.1 实验目的

根据感知器网络设计方法，结合 Matlab 程序中 Neural network Toolbox 的程序，设计、编写并实现英文 26 个字母的识别网络（并以此拓展至十个阿拉伯数字或其他字符的识别网络）。为提高网络的识别精度，同时训练无噪声和含有噪声 0.0:0.02:0.2 情况下的网络。之后，测试当被识别的字母在含有噪声随机为 0.0:0.02:0.2 情况下，输入 100 次测试的正确识别率。

在本次实验中，应在理解前向神经网络基本构造的基础上，熟练运用 Matlab 工具箱，进一步加深对于感知器网络设计和训练方法的理解；同时借助人工添加噪声，理解和体会神经网络识别精度训练的重要性，并掌握评估训练识别效果的一般方法。

1.2 实验原理

神经网络的训练应当是有监督地训练出输入端的 26 组分别表示字母 A 到 Z 的数组，能够对应到输出端 1 到 26 的具体位置。首先必须对每个字母进行数字化处理，以便构造输入样本，考虑用 5×7 的布尔值可以清楚地表示出每个字母，即用 35 个元素表示一个字母，例如字母 A 和 Z 可以分别用 0、1 矩阵表示为：

```
letterA = [0 0 1 0 0 ...      letterZ = [ 1 1 1 1 1 ...
           0 1 0 1 0 ...                  0 0 0 0 1 ...
           0 1 0 1 0 ...                  0 0 0 1 0 ...
           1 0 0 0 1 ...                  0 0 1 0 0 ...
           1 1 1 1 1 ...                  0 1 0 0 0 ...
           1 0 0 0 1 ...                  1 0 0 0 0 ...
           1 0 0 0 1]';                   1 1 1 1 1]';
```

其中矩阵后的一撇'表示转置。字母的表示如图 1 所示：

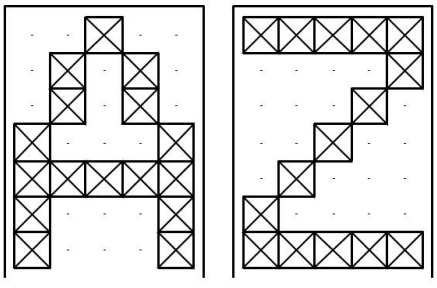

图 1　用 5×7 矩阵表示字母 A 和 Z

然后将这些字母送入 35×26 的矩阵 alphabet 中，作为神经网络的输入矩阵：

```
alphabet = [letterA,letterB,letterC,letterD,letterE,...
            letterG,letterH,letterI,letterJ,letterK,...
            letterL,letterM,letterN,letterO,letterP,...
            letterQ,letterR,letterS,letterT,letterU,...
            letterV,letterW,letterX,letterY,letterZ];
```

另一方面，因为目标矢量是希望在每一个字母输出时，在 26 个字母中它所排顺序的位置上输出为 1，而在其他位置上输出为 0。为此，取目标矩阵为对角线上为 1 的 26×26 的单位阵：

targets=eye(26);

所有以上关于输入和输出矢量的操作都写入名为 prodat.m 的函数文件中备用：

```
function[alphabet,targets]=prodat
    ...          %函数具体内容
end
```

而神经网络的具体训练函数，可以利用 matlab 中的 Neural network Toolbox 工具箱实现，其核心操作函数为：

```
net = newp(minmax(P),S);      %新建感知器网络
[net,tr] = train(net,P,T);          %训练网络
```

在接下来的实验过程中，先根据例程复现代码，再依据实验要求作相应的修改和调整，最后给出分析与体会。

2 示例程序复现与分析

2.1 初始化与无噪声训练

依据感知器网络设计原理，以 35×26 的矩阵 alphabet 作为输入矩阵 P，26×26 的单位阵 targets 作为目标矩阵 T，设定目标误差为 0.001，最大训练次数设为 40，可写出初始化程序为：

```
% exp1

% 初始化
[alphabet,targets]=prodat;
P = alphabet;
T = targets;
[R,Q] = size(P);
[S,Q] = size(T);
net = newp(minmax(P),S);      % 注意，这里采用的是感知器网络
net.trainParam.goal = 0.0001;      %定义目标误差
net.trainParam.epochs = 40;      %定义最大循环次数
[net,tr] = train(net,P,T);        %训练网络
W = net.iw{1,1};      %训练后的网络权值
B = net.b{1};         %训练后的网络偏差
```

图 2　初始化与无噪声训练程序部分

此部分训练为感知器单层网络的训练，它具有 35 个输入节点和 26 个输出节点。这一感知器对 26 个英文字母进行分类时，其问题转化为：在由标准输入矢量为坐标轴所组成的 7×5=35 维的输入矢量空间里，每个字母位于该输入矢量空间中的不同的位置，通过训练来获取网络的权值 W 和 b，使在其作用下，将输入矢量的 35 维空间分成 26 个区域，通过所设计的 W*P+b=0 的轨迹能够对输入矢量按期望输出值进行分类。

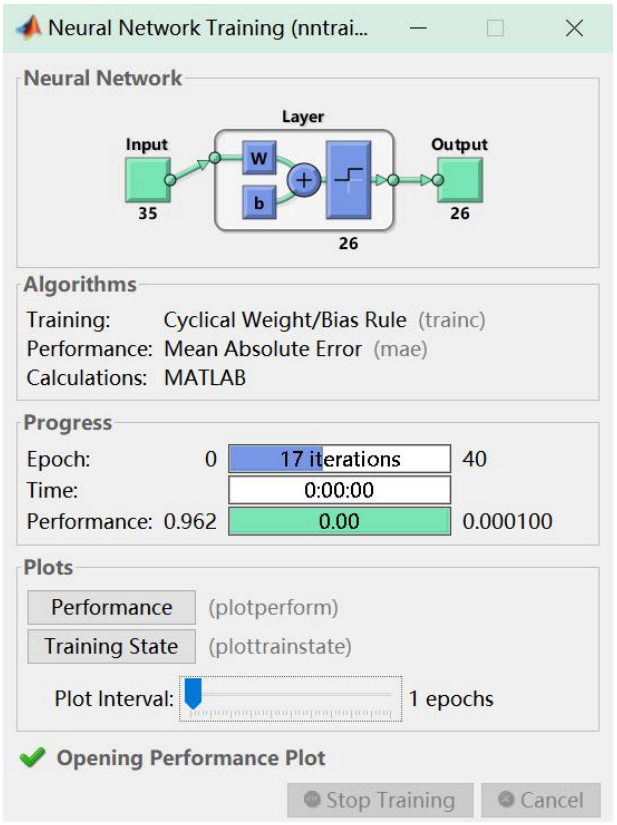

图 3　matlab NNT 工具箱界面

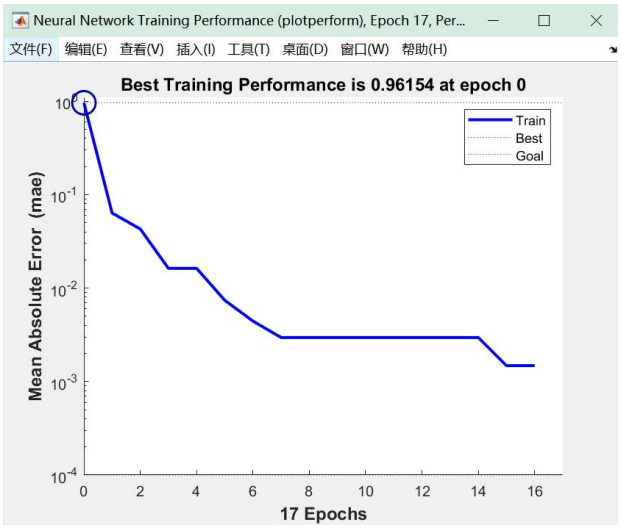

图 4　无噪声理想字母训练结果

matlab 工具箱对神经网络的训练结果提供了可视化的分析。图 3 展示了神经网络的结构、分析算法和进程；图 4 展现了随着训练循环次数的增长，训练效果（根据某一指标来界定）的变化情况。根据训练结果可知，在无噪声理想字母的训练中，网络一共经过了 17 次循环，小于所设定的 40 次，使其平均绝对值误差（Mean Absolute Error, mae）达

到所设定的 0.001 值。

2.2 有噪声训练

为了获得一个对噪声不敏感的网络，首先需要训练一个具有两组理想输入矢量 alphabet 和两组带有噪声的输入矢量，目标矢量为 4 组期望矢量 targets，噪声矢量均为均值为 0.1 到 0.2 的随机噪声。对于有噪声的训练，最大训练次数减为 300 次，误差仍为 0.001：

```
% 具有噪声的输入矢量识别网络——训练一个网络，其具有两组理想输入矢量alphabet和
% 目标矢量为4组期望矢量targets，噪声矢量为均值0.1-0.2的随机噪声
netn = net;
netn.trainParam.goal = 0.0001;      %定义目标误差
netn.trainParam.epochs = 300;       %定义最大循环次数
for pass=1:10
    T=[targets targets targets targets];
    fprintf('Pass=%.0f\n',pass);
    P=[alphabet,alphabet,(alphabet+randn(R,Q)*0.1),(alphabet+randn(R,Q)*0.2)];
    [netn,tr]=train(netn,P,T);
end
% 有必要再用理想无噪声输入矢量对网络再训练一次，以保证网络可以准确识别理想字母
P = alphabet;
T = targets;
[net,tr] = train(net,P,T);          %训练网络
```

图 5 有噪声训练程序部分

网络训练完成后，有必要用理想的无噪声输入矢量对网络再训练一次，以保证网络能够准确无误地识别出理想的字母。

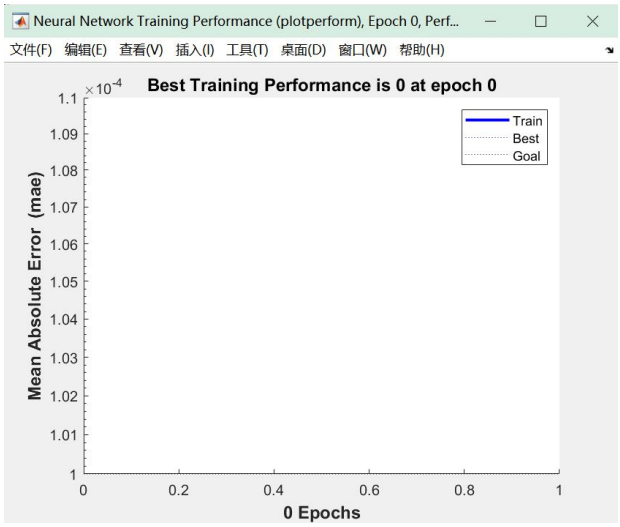

图 6 有噪声训练结果（最后一次）

图 6 展示了最后一次训练的结果，可见此时给入理想无噪声输入后，网络可以准确无误地输出预期结果，没有任何误差。

2.3 系统性能测试

神经网络进行模式识别的可靠性，可以通过使用上百个具有随机噪声的输入矢量来测试网络而获得。我们对所设计的神经网络输入任意字母，并在其上加入具有平均值为 0.1 到 0.2 的噪声，由此随机产生 100 个输入矢量，通过网络识别后输出。测试结束后，利用 matlab 工具箱绘图工具，绘制了未经噪声训练和经噪声训练后的网络的辨识出错率随所加噪声水平的变化，以直观辨别两网络的性能优劣。

```
noise_range=0:0.02:0.2;
max_test=100;
network1=[];
network2=[];
T=targets;
for noiselevel=noise_range
    fprintf('Testing networks with noise level of %.2f.\n',noiselevel);
    errors1=0;
    errors2=0;
    for i=1:max_test
        P=alphabet+randn(R,Q)*noiselevel;
        A=sim(net,P);       %测试用理想字母训练的网络
        errors1=errors1+sum(sum(abs(A-T)))/2;
        An=sim(netn,P);     %测试噪声字母训练的网络
        errors2=errors2+sum(sum(abs(An-T)))/2;
    end

network1=[network1 errors1/26/100];
network2=[network2 errors2/26/100];
end
figure(1);      % 显示上述两种网络辨识结果的出错率
plot(noise_range,network1*100,'--',noise_range,network2*100);
xlabel('Noise Level');
ylabel('Percentage of Recognition');
```

图 7 性能测试程序部分

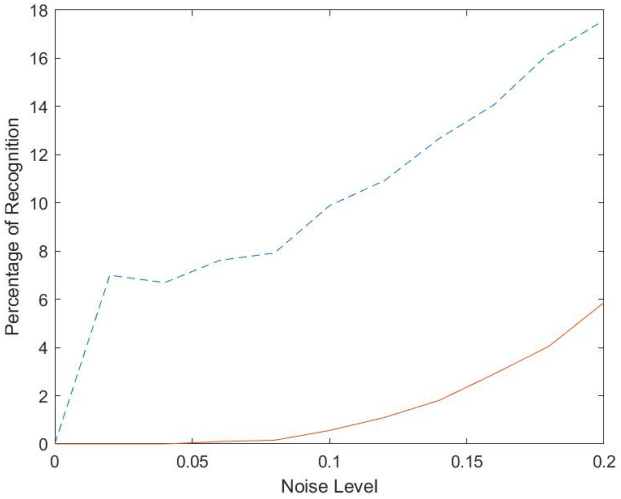

图 8 字母辨识出错率对比

上图中，蓝线为未经噪声训练的网络，红线为经噪声训练后的网络。可以看出，若单纯采用无噪声的标准字母训练出的网络对字符进行工作识别，当输入网络的字符出现与标准字母有偏差的噪声时，网络识别立即出现错误，且随噪声水平的上涨，出错率急剧增高，最终接近于 20%。而采用有噪声样本训练出的网络具有一定的抗干扰能力：在 0.07 左右噪声水平之前可以准确无误地识别，而当噪声水平上涨时，其表现也远小于无噪声训练网络，最

大识别出错率不足 6%。

3 代码修改与调试

3.1 更换输入矢量

识别字母的实例告诉我们，神经网络具有进行模式识别的能力，这是一个非常有使用价值的功能。受此启发，在示例代码的基础上，可以将输入矢量更改为阿拉伯数字，进行数字识别的测试。例如数字 2 和数字 9 分别表示为：

```
num2 = [0 1 1 1 0 ...        num9= [ 0 1 1 1 0 ...
        1 0 0 0 1 ...                1 0 0 0 1 ...
        0 0 0 0 1 ...                1 0 0 0 1 ...
        0 0 0 1 0 ...                0 1 1 1 1 ...
        0 0 1 0 0 ...                0 0 0 0 1 ...
        0 1 0 0 1 ...                1 0 0 0 1 ...
        1 1 1 1 1]';                 0 1 1 1 0]';
```

这样输入矢量 num 就成为了 35×10 的矩阵，每一列代表一个数字：

```
num = [num1,num2,num3,num4,num5,num6, ...
       num7,num8,num9,num0];
```

目标矢量对应变为对角线上为 1 的 10×10 的单位阵：

```
targets=eye(10);
```

所有以上关于输入和输出矢量的操作都写入名为 prodat2.m 的函数文件中备用

初始化程序与前相似：

```
% expl-alter

% 初始化
% 修改：利用prodat2数据初始化，即改为数字识别
[num,targets]=prodat2;
P = num;
T = targets;
[R,Q] = size(P);
[S,Q] = size(T);
net = newp(minmax(P),S);      % 注意，这里采用的是感知器网络
net.trainParam.goal = 0.0001;      %定义目标误差
net.trainParam.epochs = 40;      %定义最大循环次数
[net,tr] = train(net,P,T);        %训练网络
W = net.iw{1,1};      %训练后的网络权值
B = net.b{1};         %训练后的网络偏差
```

图 9　无噪声数字识别训练程序部分

由图 11 可知，该网络一共经过 11 次循环训练，但最终误差仅达到 10^{-2} 量级，并未达到预设的 0.0001 目标。而且并未达到最大循环次数 40，这说明训练效果已经达到最佳。出现这种情况可能是因为，数字识别中 6.8.9 这些数字比较相似，代表它们的列向量中可能仅仅只有一两个元素不同，从而导致网络训练不能达到很精细的效果。

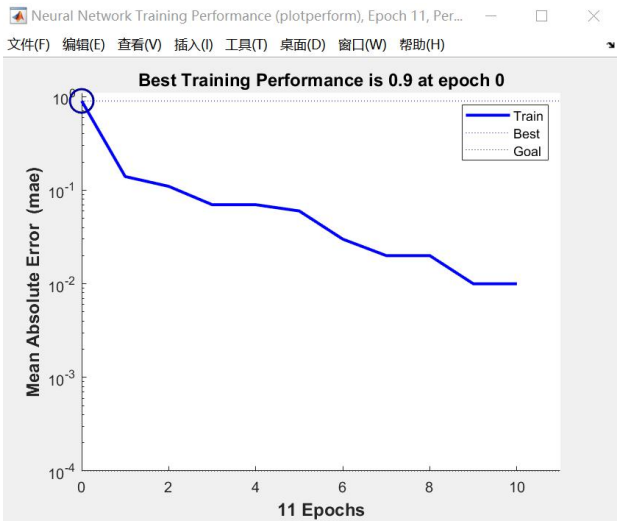

图 10　无噪声理想数字训练结果

3.2 扩大噪声范围

在噪声训练中，扩大噪声范围至 0.2—0.5，以期得到更好的训练效果，其他参数不变：

```
% 修改：训练噪声矢量范围扩大至0.2-0.5
netn = net;
netn.trainParam.goal = 0.0001;      %定义目标误差
netn.trainParam.epochs = 300;      %定义最大循环次数
for pass=1:10
    T=[targets targets targets targets];
    fprintf('Pass=%.0f\n',pass);
    P=[num,num,(num+randn(R,Q)*0.2),(num+randn(R,Q)*0.5)];
    [netn,tr]=train(netn,P,T);
end
% 有必要再用理想无噪声输入矢量对网络再训练一次，以保证网络可以准确识别理想字母
P = num;
T = targets;
[net,tr] = train(net,P,T);        %训练网络
```

图 11　有噪声数字训练程序部分

3.3 系统性能测试

在输入上加入具有平均值为 0.2 到 0.5 的噪声，由此随机产生 100 个输入矢量，通过网络识别后输出，绘制辨识出错率变化图：

```
noise_range=0:0.05:0.5;
max_test=100;
network1=[];
network2=[];
T=targets;
for noiselevel=noise_range
    fprintf('Testing networks with noise level of %.2f.\n',noiselevel);
    errors1=0;
    errors2=0;
    for i=1:max_test
        P=num+randn(R,Q)*noiselevel;
        A=sim(net,P);        %测试用理想字母训练的网络
        errors1=errors1+sum(sum(abs(A-T)))/2;
        An=sim(netn,P);      %测试噪声字母训练的网络
        errors2=errors2+sum(sum(abs(An-T)))/2;
    end

network1=[network1 errors1/10/100];
network2=[network2 errors2/10/100];
end
```

```
figure(1);        % 显示上述两种网络辨识结果的出错率
plot(noise_range,network1*100,'--',noise_range,network2*100);
xlabel('Noise Level');
ylabel('Percentage of Recognition');
```

图 12　系统性能测试程序部分

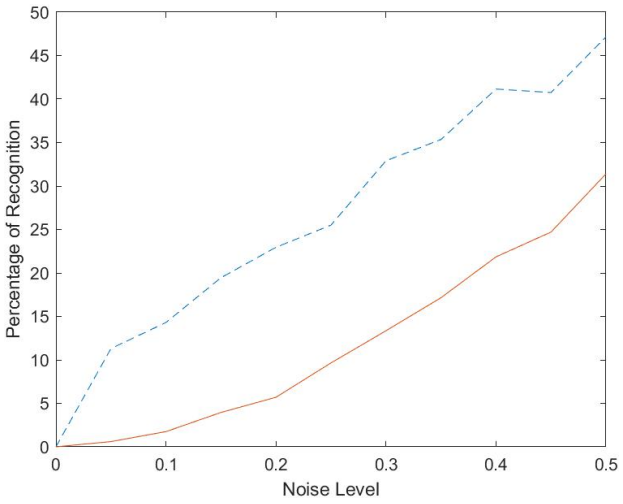

图 13　数字辨识出错率对比

从图中可以看出，辨识的误差很大；在噪声率达到 0.5 时，无噪声训练网络的出错率甚至接近 50%，有噪声训练网络也达到了 30%。但其实情有可原——输入矢量本身就是 0/1 的组合；在噪声达到 0.5 范围时，已经达到了 0 和 1 的中间数，这样就更有可能因为噪声的影响而混淆 0 和 1；同时，数字识别中 6.8.9 这些数字比较相似，代表它们的列向量中可能仅仅只有一两个元素不同，从而导致网络训练不能达到很精细的效果。误差对于其中关键位置的细小改变，很可能就会引起识别结果的错误。

为此，可以考虑对输入矢量即数字的点阵像素图进行调整。在上述训练中，以数字 8 和数字 9 举例，表示他们的点阵像素为：

```
num8 = [0 1 1 1 0 ...        num9= [ 0 1 1 1 0 ...
        1 0 0 0 1 ...                1 0 0 0 1 ...
        1 0 0 0 1 ...                1 0 0 0 1 ...
        0 1 1 1 0 ...                0 1 1 1 1 ...
        1 0 0 0 1 ...                0 0 0 0 1 ...
        1 0 0 0 1 ...                0 0 0 0 1 ...
        0 1 1 1 0]';                 0 1 1 1 0]';
```

可以看出，整个 7×5 矩阵中只有标注出的两个位置的数值不同。若噪声的引入恰好模糊了此两处数字的界限，网络的识别就极有可能出错，更别提 0.5 的噪声峰值了。类似的数字 6，数字 5 也相差很少，不再重复分析。对此，微调各输入矢量，比如将数字 9 的边角“锐化”，使之与其他数字有更明显的区别：

```
num8 = [0 1 1 1 0 ...        num9= [ 0 1 1 1 0 ...
        1 0 0 0 1 ...                1 0 0 0 1 ...
        1 0 0 0 1 ...                1 0 0 0 1 ...
        0 1 1 1 0 ...                0 1 1 1 1 ...
        1 0 0 0 1 ...                0 0 0 0 1 ...
        1 0 0 0 1 ...                0 0 0 0 1 ...
        0 1 1 1 0]';                 1 1 1 1 1]';
```

对于其他的数字也做类似的微调。据此重新执行上述代码，得到新的运行结果：

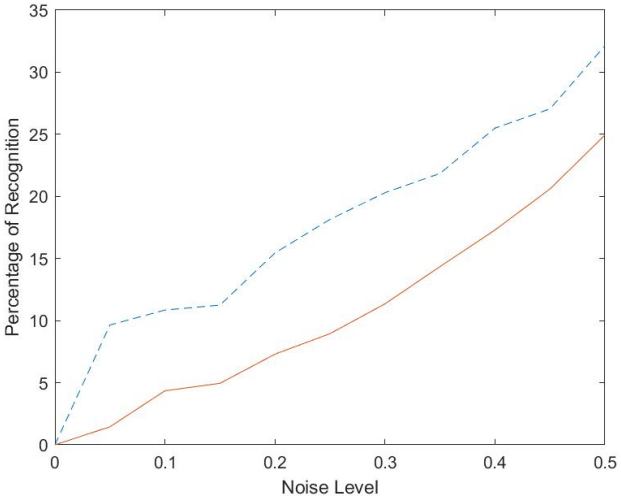

图 14　数字辨识出错率对比（调整后）

对比图 13 和图 14，尤其是其纵坐标，可以看出，在修改输入矢量使其各列向量即每个数字之间的差别增大后，出错率有了略微的下降，尤其是无噪声训练网络在高噪声水平下的出错率由接近 50%下降到了不足 35%，有噪声训练网络在高噪声水平下的出错率也由超过 30%下降到了 25%左右。

4 实验分析及心得体会

本次实验借助 matlab 神经网络工具箱，分析、设计、编写并实现了利用感知器实现 26 个英文字母和 10 个阿拉伯数字的识别网络，尤其是着重分析了他们在输入矢量具有一定量的均匀噪声时的识别性能。结论是：经过噪声训练的网络在低噪声下具有一定的抗噪能力，可以较好地识别 26 个英文字母、10 个阿拉伯数字抑或是其他模式识别问题；而在较高噪声下，本实验所设计的各位网络均有着或多或少的误差偏移，一方面可能是因为输入矢量精度不高，另一方面也与感知器网络自身只有单层的局限性有关。在这一方面，BP 网络或者

Hopfield 网络可能会有更好的表现。

通过本次实验，对感知器网络原理有了更深层次的认识，进一步加深了对于感知器网络设计和训练方法的理解，体会了神经网络识别精度训练的重要性，并掌握评估训练识别效果的一般方法；同时掌握了 matlab 软件的基本操作，基本掌握了 matlab 神经网络工具箱的使用方法，体会到了 matlab 软件在进行矩阵运算时的方便快捷。

参考文献:

[1] 丛爽. 面向 MATLAB 工具箱的神经网络理论与应用（第三版）[M]. 合肥：中国科学技术大学出版社, 2009.4: 293-305.