西北工业大学

《基于 MATLAB 的数字信号处理》实验报告

学　　院：　电子信息学院

学　　号：　2018201544

姓　　名：　方阳

专　　业：　电路与系统

西北工业大学

2019 年 04 月

实验一　　MATLAB 基本编程实验

一、实验目的及要求

1．熟悉 MATLAB 运行环境；

2. 掌握 MATLAB 的基本语法和函数；

3. 掌握 MATLAB 的基本绘图功能

二、实验设备（环境）及要求

1． 计算机

2． Matlab 软件编程实验平台

三、实验内容与步骤

1．求下列线性方程组的解

$$
\begin{aligned}
&6x_1+3x_2+4x_3=3\\
&-2x_1+5x_2+7x_3=-4\\
&8x_1-4x_2-3x_3=-7
\end{aligned}
$$

2. 分别用 for 或 while 循环结构编写程序，求出 $K=\sum_{i=1}^{106}\frac{\sqrt{3}}{2^i}$。并考虑一种避免循环语句的程序设计算法实现同样的运算。

3. 在同一坐标系下绘制以下 3 条曲线,并作标记。

$$y_1=\sin x, y_2=\sin x\sin(10x), y_3=-\cos x,\quad x\in(0,\pi)$$

四、设计思路

1.1 求解线性方程组：使用 A\B。

1.2 for 循环求和：以 i 的值构建循环，求和数列。

公式求和：直接使用等比数列求和公式进行求和。

1.3 生成 figure，使用 plot 命令画图。

五、程序代码及注释

程序 1.1：

```
%% 线性方程组的解
A = [6,3,4; -2,5,7; 8,-4,-3];
B = [3; -4; -7];
x = A\B;
%x1,x2,x3 分别为 0.6000,7.0000,-5.4000
```

程序 1.2：

```
%% 等比数列求和
%-----------for 循环-----------%
m = sqrt(3);
sum = 0;
for i =1:106
    sum = sum + 1/(2^i);
end
sum1 = m*sum
%----构建等比数列求和公式----%
a1 = sqrt(3)/2;
q = 1/2;
n = 106;
sum2 = a1*(1 - q^n)/(1 - q);
```

程序 1.3：

```
%% 同一坐标系绘制曲线
```

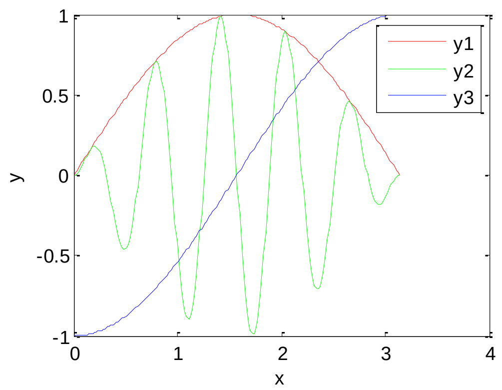
```
x = linspace(0, pi, 200);

y1 = sin(x);

y2 = sin(x).*sin(10*x);

y3 = -cos(x);

figure();

plot(x, y1, 'r');

hold on;

plot(x, y2, 'g');

hold on;

plot(x, y3, 'b');

xlabel('x');

ylabel('y');

legend('y1', 'y2', 'y3');
```

六、实验结果

实验一：x1,x2,x3 分别为 0.6000,7.0000,-5.4000

实验二：求和结果为 1.7321

实验三：图形结果如下：



实验二　基于 MATLAB 的数字信号处理实验

一、实验目的及要求

1. 回顾数字信号处理的主要内容；
2. 掌握利用 MATLAB 进行信号处理的方法；
3. 了解信号处理工具箱中一些函数的功能；

二、实验设备（环境）及要求

1. 计算机
2. Matlab 软件编程实验平台

三、实验内容

1. 设序列

$$x(n)=[3,1,7,0,-1,4,2],\quad -3\le n\le 3;$$
$$y(n)=x(n-2)+w(n),$$

其中，$w(n)$是均值为0，方差为1 的高斯随机序列.

利用 2 种方法计算 x(n)和 y(n)之间的互相关，并画出互相关序列图.

2．一数字滤波器由 $H(e^{j\omega})=\dfrac{1+e^{-j4\omega}}{1-0.8145e^{-j4\omega}}$ 频率响应函数描述

1）写出其差分方程表示；

2）画出上面滤波器的幅频和相频图；

3）产生信号 $x(n)=\sin(\pi n/2)+5\cos(\pi n)$ 的 200 个样本，通过该滤波器得到输出 y(n), 试将输出 y(n)的稳态部分与 x(n)作比较，说明这两个正弦信号的幅度和相位是如何受该滤波器影响的。

3. 用 3 种方法设计带通滤波器（Butterworth、椭圆和窗函数），采样频率 fs ＝2000Hz, 通带频率 300Hz—600Hz，阶数自选，画出滤波器的频率响应，并对三种方法设计的滤波器性能进行分析比较。

四、设计思路

2.1：　方法一可以通过 xcorr 函数求取互相关，方法二则是通过卷积的方式求取互相关。

2.2：由系统的差分方程 $y(n)-0.8145y(n-4)=x(n)+x(n-4)$，得到方程的各个项的系数，进而可求出系统的幅频和相频响应

2.3：首先选定带通滤波器的阶数，根据采样频率、上、下限截止频率求得滤波器的各个参数，然后得出窗函数的频率特性

五、程序代码及注释

程序 2.1：

```
%% 互相关
nx=[-3,-2,-1,0,1,2,3];
x1=[3 1 7 0 -1 4 2];
x2 = x1;
k = length(x2);
e = randn(1, k);
ny = nx + 2;
y = x2 + e;
% 使用xcorr函数
figure();
subplot(1, 2, 1);
r1 = xcorr(x1, y);
nx_len = length(nx);
n = [nx(1)+ny(1): nx(nx_len)+ny(nx_len)];
stem(n, r1);
xlabel('x');
title('xcorr求互相关')
% 使用conv求互相关
```

```
subplot(1, 2, 2);
x1 = fliplr(x1);
conv_x1y = conv(y, x1);
conv_x1y = fliplr(conv_x1y);
stem(n, conv_x1y);
xlabel('x');
title('conv 求互相关')
```

程序 2.2.2：

```
%% 幅频与相频图
clc
clear all
fs=1000;
b=[1 0 0 0 1];
a=[1 0 0 0 -.8145];
[h,f]=freqz(b,a,512,fs);
mag=abs(h);%幅度
ph=angle(h);%相位
subplot(2,1,1);
ph=ph*180/pi;%由弧度转换为角度
plot(f,mag);
grid;
```

```
xlabel('Frequency/Hz');

ylabel('Magnitude');

title('幅频响应');

subplot(2,1,2);

plot(f,ph);

grid;

xlabel('Frequency/Hz');

ylabel('Phase');

title('相频响应');
```

程序 2.2.3：

```
%% 稳态部分

clc

clear all

N=200;

n=linspace(-100,100,N);

x=sin(pi*n/2)+5*cos(pi*n);

N_fft=2^nextpow2(2*N);

w=linspace(0,2*pi,N_fft);

h_fft=(1+exp(-1j*4*w))./(1-0.8145*exp(-1j*4*w));

x_fft=fft(x,N_fft);

y_fft=x_fft.*h_fft;
```

```
y_temp=fftshift(ifft(y_fft));

y=y_temp(N_fft/2:N_fft/2+N-1);

figure;

plot(w,abs(h_fft),'b','LineWidth',2);

hold on;

plot(w,angle(h_fft),'g','LineWidth',2);

legend('幅度','相位')

figure;

plot(n,x,'b');

hold on;

plot(n,real(y),'g');

legend('x(n)','y(n)稳态部分')
```

程序 2.3：

```
%% 带通滤波器

clc

clear all

fs=2000; %采样频率

fc1=300;  %下限截止频率

fc2=600;%上限截止频率

N=20;      % 滤波器的阶数

wlp=fc1/(fs/2);
```

```
whp=fc2/(fs/2);
wn=[wlp,whp];;   %滤波器归一化后的上下限截止频率
%用不同的方法设计 N 阶的滤波器
[b1 a1] = butter(N,wn, 'bandpass'); %butterworth
% 3dB 的通带纹波，40dB 的阻带衰减
[b2 a2] = ellip(N,3, 40, wn,'bandpass')%椭圆
w3=hamming(N);  %海明窗
b3=fir1(N-1,wn,w3);
%求出滤波器的频率响应
[H1 f1]=freqz(b1,a1);
[H2 f2]=freqz(b2,a2);
[H3 f3]=freqz(b3,1,512,fs);
figure;
subplot(2,1,1);
plot(f1,20*log10(abs(H1)));
xlabel('频率/Hz');
ylabel('振幅/dB');
title('butterworth 的幅频特性');
grid on;
subplot(2,1,2);
plot(f1,180/pi*unwrap(angle(H1)));
xlabel('频率/Hz');
```

```
ylabel('相位');
title('butterworth的相频特性');
grid on;
figure;
subplot(2,1,1);
plot(f2,20*log10(abs(H2)));
xlabel('频率/Hz');
ylabel('振幅/dB');
title('椭圆的幅频特性');
grid on;
subplot(2,1,2);
plot(f2,180/pi*unwrap(angle(H2)));
xlabel('频率/Hz');
ylabel('相位');
title('椭圆的相频特性');
grid on;
figure;
subplot(2,1,1);
plot(f3,20*log10(abs(H3)));
xlabel('频率/Hz');
ylabel('振幅/dB');
title('海明窗的幅频特性');
```

```
grid on;

subplot(2,1,2);

plot(f3,180/pi*unwrap(angle(H3)));

xlabel('频率/Hz');

ylabel('相位');

title('海明窗的相频特性');

grid on;
```

六、实验结果

实验结果 2.1：

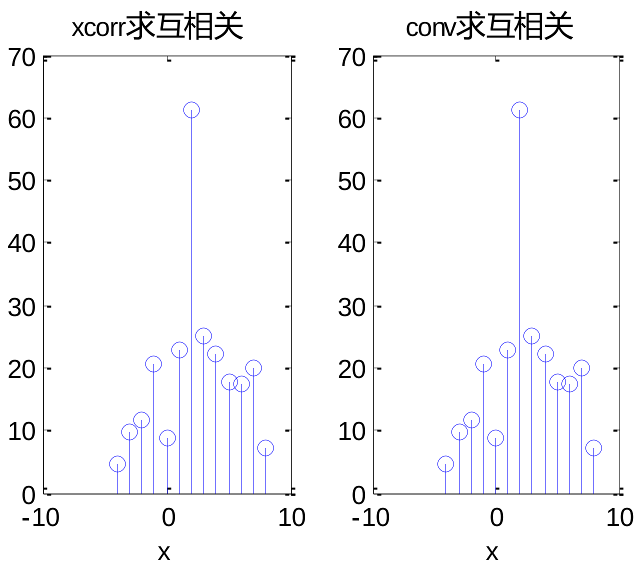


图 2-1 x(n)与 y(n)的互相关序列图

由实验结果可知，x(n)与 y(n)的互相关只在区间[-4,8]上有能力，刚好是区间[-3,3]与右移后的区间[-1,5]两端点之和，与结论一致。且互相关在 2 处达到最大。

实验结果 2.2.2:

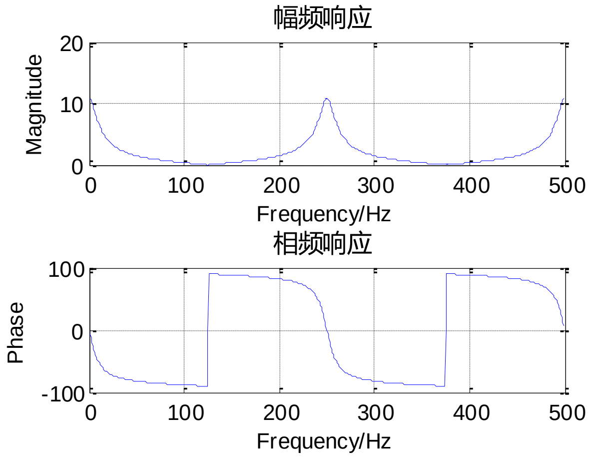


图 2-2 滤波器的幅频与相频图

实验结果 2.2.3：

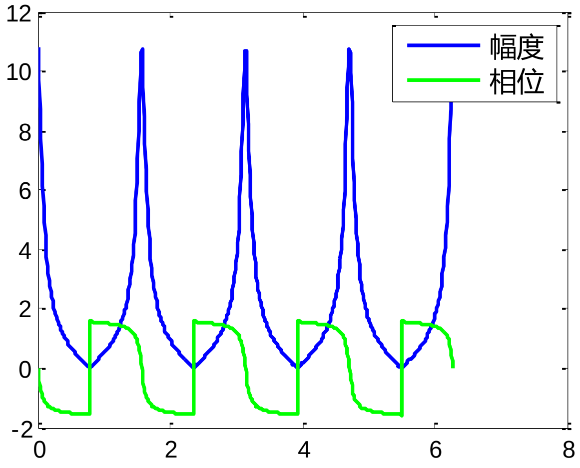


图 2-3 滤波器的幅度和相位变化

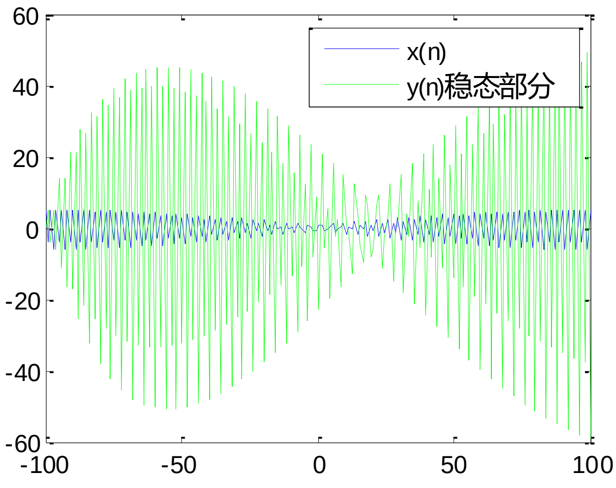


图 2-4 两信号波形

由结果知，输出信号相对于输入信号有一小小的延迟，基本上 x(n)的频点都通过了，滤波器是个梳状 filter，正好在想通过的点附近相位为 0，也就是附加延迟为 0

实验结果 2.3：

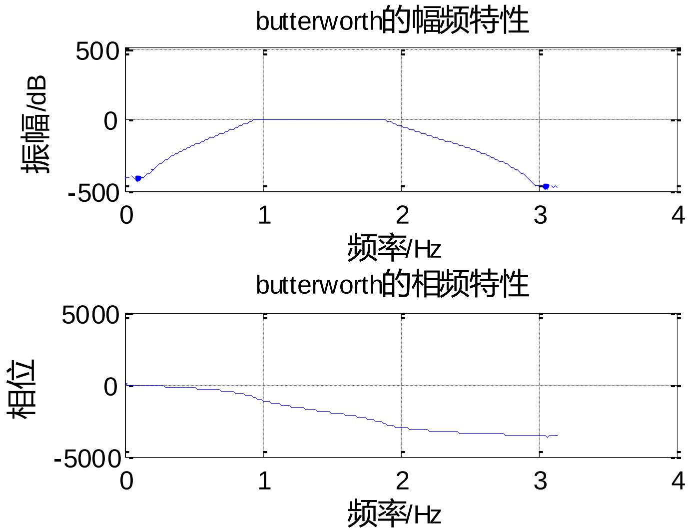


图 2-5 butterworth 频率特性

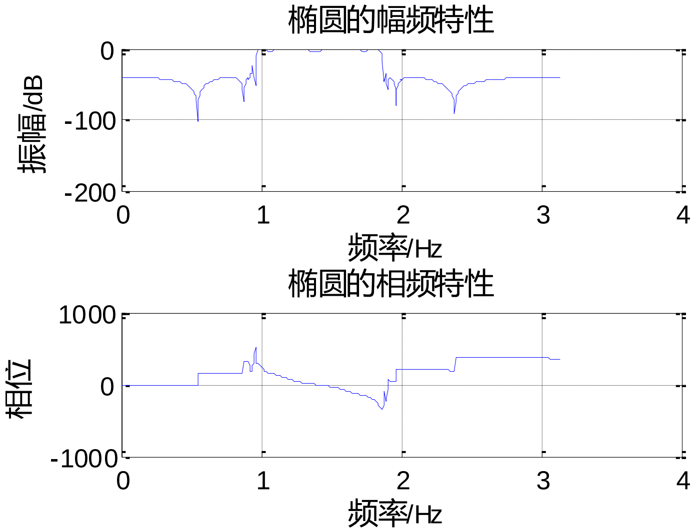
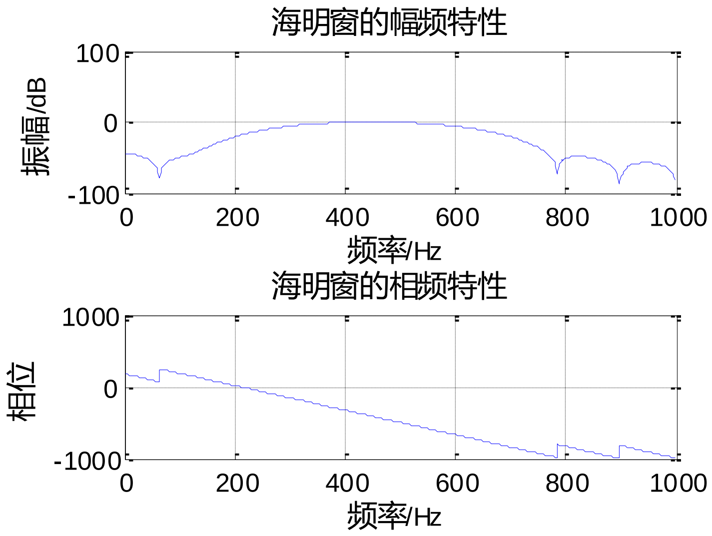


图 2-6 椭圆频率特性



海明窗的幅频特性
振幅/dB
100
0
-100
0
200
400
600
800
1000
频率/Hz
海明窗的相频特性
相位
1000
0
-1000
0
200
400
600
800
1000
频率/Hz

图 2-7 海明窗频率特性

实验三　　基于 MATLAB 的图像处理实验

一、实验目的及要求

1．了解图像处理的基本概念和功能；

2. 掌握利用 MATLAB 进行图像处理的方法；

3. 了解图像处理工具箱中一些函数的功能；

二、实验设备（环境）及要求

1． 计算机

2． Matlab 软件编程实验平台

三、实验内容

1．对分别添加了椒盐噪声（密度为 0.03）和高斯白噪声（均值为 0，方差为 0.02）的图像，利用三种方法进行去噪，显示原始图像、加噪图像和去噪图像并对实验结果进行分析。

2．对 cameraman.tif 图像进行 DCT 变换，分别选取绝对值最大的 1/4、1/8、1/16 个变换系数(其余置为 0)，进行反 DCT 得到重构图像，显示原图像和各重构图像并分别计算重构图像的峰值信噪比。

四、设计思路

3.1：利用matlab自带函数imnoise()向图像中加入椒盐噪声和高斯噪声，对噪声图像进行中值、均值、中值加均值滤波处理。均值滤波是一种线性滤波，也是低通滤波。中值滤波是一种统计滤波器，是非线性的。分别向图像中加入高斯噪声和椒盐噪声，利用不同的滤波方法，比较结果，得出结论。由实验可以看出，中值滤波对椒盐噪声的处理效果好，均值滤波对高斯噪声的处理效果好。

3.2：由于 DCT 变换有使图像能量几种在左上方的特性，因此重构图像保留了原始图像大部分的图像特征，其视觉效果与原始图像相差不大。对比重构前后的图像易知，重构后的图像稍显模糊，这是因为该压缩算法为有损压缩，压缩后的图像丢失了原始图像部分数据信息。

五、程序代码及注释

程序 3.1：

```
%% 中值滤波
img_orig = imread('cameraman.tif');
img_noise=imnoise(img_orig, 'salt & pepper', 0.03);
img_noise=imnoise(img_noise, 'gaussian', 0, 0.02);
img_recover1 = medfilt2(img_noise);
figure();
subplot(1, 3, 1);
imshow(img_orig);
title('原图');
subplot(1, 3, 2);
imshow(img_noise);
title('噪声图');
subplot(1, 3, 3);
imshow(img_recover1);
title('中值滤波恢复图');
%% 均值滤波
H1 = fspecial('average', 3);
img_recover2 = imfilter(img_noise, H1);
figure();
subplot(1, 3, 1);
imshow(img_orig);
```

```
title('原图');
subplot(1, 3, 2);
imshow(img_noise);
title('噪声图');
subplot(1, 3, 3);
imshow(img_recover2);
title('均值滤波恢复图');
%% 中值加均值滤波
img_recover3 = medfilt2(img_noise);
img_recover3 = imfilter(img_recover3, H1);
figure();
subplot(1, 3, 1);
imshow(img_orig);
title('原图');
subplot(1, 3, 2);
imshow(img_noise);
title('噪声图');
subplot(1, 3, 3);
imshow(img_recover3);
title('中值加均值滤波恢复图');
```

程序 3.2：

```
%% DCT
%读取源图像
I=imread('cameraman.tif');
%对图像进行离散余弦变换
J=dct2(I);
v = flip(sort(abs(J(:))));
%1/4
c1 = v(length(v)/4);
[col row] = size(find(abs(J)< c1));
A=col*row;%置为 0 的变换系数的个数
%1/8
c2 = v(length(v)/8);
[col row]= size(find(abs(J)< c2));
B=col*row;%置为 0 的变换系数的个数
%1/16
c3 = v(length(v)/16);
[col row]= size(find(abs(J)< c3));
C=col*row;%置为 0 的变换系数的个数
%将小于 1/4 的最大值变换系数置为 0 后做离散余弦反变换
J(abs(J) < c1 ) = 0;I1=idct2(J);
%将小于 1/8 的最大值变换系数置为 0 后做离散余弦反变换
J(abs(J) < c2) = 0;I2=idct2(J);
```

```
%将小于 1/16 的最大值变换系数置为 0 后做离散余弦反变换
J(abs(J) < c3) = 0;I3=idct2(J);
%显示原图及反变换结果
figure(2);
subplot(2,2,1);
imshow(I);
title('原图');
subplot(2,2,2);
imshow(I1,[0,255]);
title('小于 1/4 最大值');
subplot(2,2,3);
imshow(I2,[0,255]);
title('小于 1/8 最大值');
subplot(2,2,4);
imshow(I3,[0,255]);
title('小于 1/16 最大值');
%计算反重构时，DCT 的变换系数的置 0 个数小于 5 时的峰值信噪比及置为 0 的变换系数的个数
I = double(I);
I1 = double(I1);
[Row,Col] = size(I);
[Row,Col] = size(I1);
```

```
MSE1 = sum(sum((I-I1).^2))/(Row * Col);
PSNR1 = 10 * log10(255^2/MSE1);
fprintf('图像的峰值信噪比：MSE1=%f\n',MSE1);
fprintf('置为0的变换系数的个数为：PSNR1=%f\n',PSNR1);
%计算反重构时，DCT的变换系数的置0个数小于10时的峰值信噪比及置为0的变换系数的个数
I = double(I);
I2 = double(I2);
[Row,Col] = size(I);
[Row,Col] = size(I2);
MSE2 = sum(sum((I-I2).^2))/(Row * Col);
PSNR2 = 10 * log10(255^2/MSE2);
fprintf('图像的峰值信噪比：MSE2=%f\n',MSE2);
fprintf('置为0的变换系数的个数为：PSNR2=%f\n',PSNR2);
%计算反重构时，DCT的变换系数的置0个数小于20时的峰值信噪比及置为0的变换系数的个数
I = double(I);
I3 = double(I3);
[Row,Col] = size(I);
[Row,Col] = size(I3);
MSE3 = sum(sum((I-I3).^2))/(Row * Col);
PSNR3 = 10 * log10(255^2/MSE3);
```

```
fprintf('图像的峰值信噪比：MSE3=%f\n',MSE3);
fprintf('置为 0 的变换系数的个数为：PSNR3=%f\n',PSNR3);
```

六、实验结果

实验结果 3.1：

图 3-1 中值滤波

图 3-2 均值滤波

图 3-3 中值加均值滤波

实验结果 3.2：

图 3-4 不同阈值的 DCT 重构

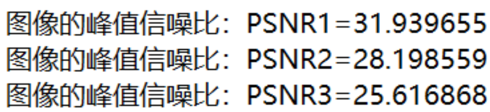

图像的峰值信噪比：PSNR1=31.939655
图像的峰值信噪比：PSNR2=28.198559
图像的峰值信噪比：PSNR3=25.616868

图 3-5 不同阈值的峰值信噪比（与图 3-4 对应）

实验四　　基于 MATLAB 神经网络编程实验

一、实验目的及要求

1．了解神经网络的基本概念和原理；

2. 掌握用 MATLAB 实现神经网络的思路和方法；

3. 了解神经网络工具箱函数的功能。

二、实验设备（环境）及要求

1． 计算机

2． Matlab 软件编程实验平台

三、实验内容

1、产生 2 维 20 组二类可分数据，进行标记并构成训练集（输入输出模式对），利用 2 输入的 MP 模型实现二类分类问题，给出实验结果并分析。

2、用多层前向网络的 BP 算法拟合下列函数

$$f(x) = 0.12e^{-0.23x} + 0.54e^{-0.17x}\sin(1.23x),\ 0 < x < 4\pi$$

说明：1）网络结构为三层（输入层、1 个隐层和输出层），隐层神经元个数自选；

2）获取两组数据，一组作为训练集，一组作为测试集；

3）用训练集训练网络；

4）用测试集测试网络性能，给出拟合结果，并计算出均方误差。

四、设计思路

4.1：随机产生两个类的数据，确保可分，对两个类的数据分别相反加上偏置，然后使用感知机进行分类，最后绘制最后的分类结果。

4.2：BP 神经网络是误差反向传播神经网络的简称，是神经网络的一个重要分支，是有监督学习网络，也是在实际应用中最常见的网络。基于并行的网络结构，主要由输入层、隐含层、输出层组成，能够实现 m 到 n 维的非线性映射，网络的学习方法采用梯度下降法。当一对学习模式提供给网络后，各神经元获得网络的输入响应并产生神经元之间的连接权值，然后按照减小希望输出与实际输出误差的方向传播，这个过程即为前向传播。网络连接的权值的修改从输出层开始，经各

中间层到输入层(误差的反向传输过程)。正向传输和反向传输过程反复进行 j 直到网络的全局误差趋向于设定的极小值，这就是 BP 算法，采用这种算法的多级非循环网络被称之为 BP 神经网络。

所以设计神经网络时，首先要确定隐层层数、隐层节点数，设置网络参数，同时选定训练集。若要提高判断或识别的准确率，可适当地增大测试集数据。

五、程序代码及注释

程序 4.1：

```
%% 二分类
clc;
close all;
clear all;
% 每个类别数量为 10
N = 10;
%随机初始化两个类的数据
data1 = rand(2, 10);
data2 = rand(2, 10);
%设置两个类的 label
data1_label = zeros(1,10);
data2_label = ones(1,10);
%将两个类的数据分开
data1(1,:) = data1(1,:) - 0.5;
data1(2,:) = data1(2,:) + 0.5;
data2(1,:) = data2(1,:) + 0.5;
```

```
data2(2,:) = data2(2,:) - 0.5;
%组合 data 和 label
data = [data1, data2];
label = [data1_label, data2_label];
% 定义感知器神经元并对其初始化
net=newp([0 1;0 1],1);
net.initFcn='initlay';
net.layers{1}.initFcn='initwb';
net.inputWeights{1,1}.initFcn='rands';
net.layerWeights{1,1}.initFcn='rands';
net.biases{1}.initFcn='rands';
net=init(net);
% 训练感知器神经元
net=train(net,data,label);
cell2mat(net.iw);
cell2mat(net.b);
%绘制结果
figure();
scatter(data1(1,:), data1(2,:));
hold on;
scatter(data2(1,:), data2(2,:));
plotpc(net.iw{1,1},net.b{1});
```

程序 4.2：

```
%% 回归
clc;
clear all;
% 产生训练样本与测试样本
x1= 0:0.5:4*pi;
x2= 0:0.12:4*pi;
 P1 = 0.12*exp(-0.23*x1) + 0.54*exp(-0.17*x1).*sin(1.23*x1);% 训练样本
T1 = P1; % 训练目标
P2 = 0.12*exp(-0.23*x2) + 0.54*exp(-0.17*x2).*sin(1.23*x2);% 测试样本
T2 = P2; % 测试目标
% 归一化
[PN1,minp,maxp,TN1,mint,maxt] = premnmx(P1,T1);
PN2 = tramnmx(P2,minp,maxp);
TN2 = tramnmx(T2,mint,maxt);
% 设置网络参数
HideNum=1;    % 隐层层数
NodeNum = 5; % 隐层节点数
TypeNum = 1; % 输出维数
TF1 = 'tansig';TF2 = 'purelin'; % 判别函数(缺省值)
net = newff(minmax(PN1),[NodeNum TypeNum],{TF1 TF2});
net.trainFcn = 'trainlm';
```

```
net.trainParam.show = 20; % 训练显示间隔

net.trainParam.lr = 0.3; % 学习步长 - traingd,traingdm

net.trainParam.mc = 0.95; % 动量项系数 - traingdm,traingdx

net.trainParam.mem_reduc = 1; % 分块计算 Hessian 矩阵
net.trainParam.epochs = 1000; % 最大训练次数

net.trainParam.goal = 1e-4; % 最小均方误差

net.trainParam.min_grad = 1e-20; % 最小梯度

net.trainParam.time = inf; % 最大训练时间

net = train(net,PN1,TN1); % 训练

YN1 = sim(net,PN1); % 训练样本实际输出

YN2 = sim(net,PN2); % 测试样本实际输出

MSE1 = mean((TN1-YN1).^2) % 训练均方误差

MSE2 = mean((TN2-YN2).^2) % 测试均方误差

Y2 = postmnmx(YN2,mint,maxt); % 反归一化

% 结果作图

plot(1:length(T2),T2,'r',1:length(Y2),Y2,'b');

fprintf('测试均方误差为：%f\n', MSE2);

legend('测试集','训练集');

title('期望输出与实际输出对比');
```

六、实验结果

实验结果 4.1：

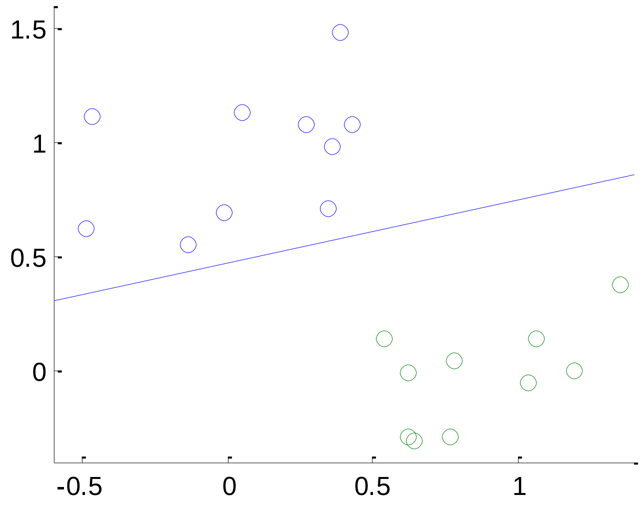


图 4-1 分类结果

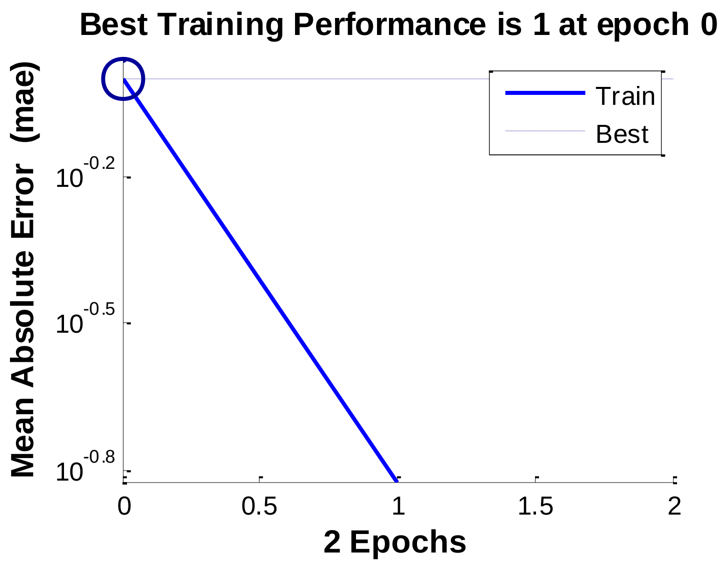


图 4-2 误差曲线

实验结果 4.2：

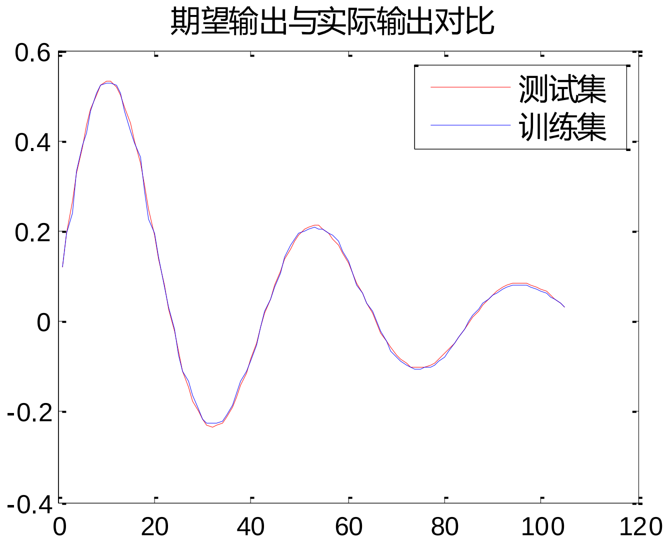


图 4-3 拟合曲线

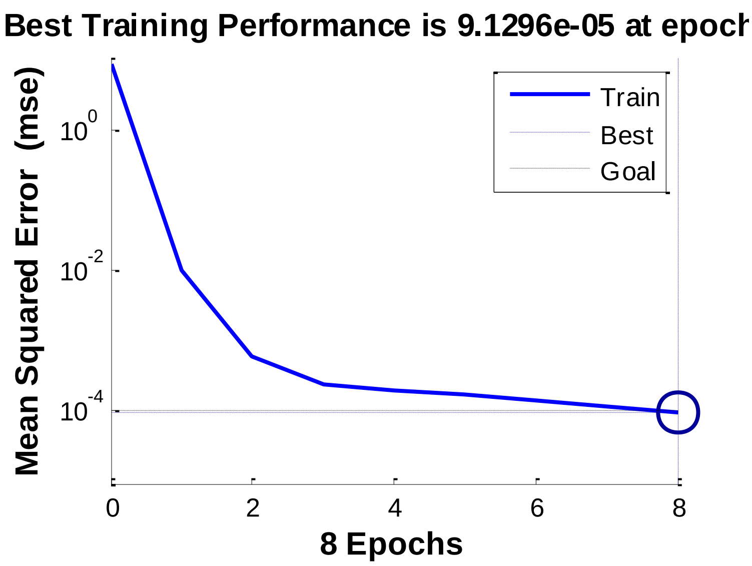


图 4-4 误差曲线

测试均方误差为：0.000278

图 4-5 测试的均方误差值

实验五　　MATLAB 的综合实验

一、实验目的及要求

培养学生利用 Matlab 解决专业问题的能力。

二、实验设备（环境）及要求

1. 计算机
2. Matlab 软件编程实验平台

三、实验内容（三题中选择一题）

1、根据所学内容，设计实现一个数字信号处理的仿真系统，要求程序具有界面，并能实现以下功能：

1）产生多种数字信号和噪声，读入语音信号，以及叠加噪声（噪声强度从键盘或鼠标输入）的数字信号，并显示时域波形；

2）具有对数字信号进行 DFT、DCT 和 DWT 变换和经典功率谱估计功能，并显示变换域及功率谱估计波形的功能；

3）具有 3 种数字信号去噪算法；

4）分别对产生的数字信号和读入的语音信号，显示时域和变换域波形，在信号上叠加噪声（信噪比为 0dB-30dB），显示时域和变换域波形，通过去噪算法对其降噪，得到输出信号的频域特性和时间序列，并计算去噪后信号的信噪比。

2、基于数字图像处理，实现一个汽车标志定位和分割的仿真系统。要求程序具有界面并实现以下功能：

1）读入自然场景下包含汽车标志的汽车图像；

2）预处理（去噪和增强）及特征提取；

3）汽车标志定位；

4）多种图像分割功能及形态学滤波功能；

5）显示中间处理结果和最终分割结果。

3、结合所学专业，实现一个神经网络或深度学习应用的实例，要求程序具有界面，并且至少 3 种方法实现（可以包括不同神经网络方法或经典方法）。

四、设计思想

3：期望设计一个手写字符识别的程序，用到了 Deep learning toolbox(链接：https://github.com/rasmusbergpalm/DeepLearnToolbox)，方案有三种，一种

使用传统的神经网络（nn）对手写字符进行识别，第二种是先对深度置信网络（dpn）进行训练，然后将深度置信网络的参数导入到神经网络中，对神经网络进行训练，第三种是先进行一个堆自编码器(sae)的训练，然后将堆自编码器的参数导入到神经网络中，对神经网络进行训练，最后看看这三种方式的训练情况与测试结果。

五、程序代码（界面除外）及注释

```
clear all;

close all;

load mnist_uint8; %导入mnist数据集


train_x = double(train_x) / 255; %归一化

test_x  = double(test_x)  / 255; %归一化

train_y = double(train_y);

test_y  = double(test_y);


%% 深度置信网络(dbn) + 神经网络(nn)

rand('state',0)

%训练dpn

dbn.sizes = [100 100]; %784->100->100->784

opts.numepochs =   3; %dpn epochs

opts.batchsize = 100;  % dpn batchsize

opts.momentum  =   0; % sgd no momentum

opts.alpha     =   1;  %dpn参数

dbn = dbnsetup(dbn, train_x, opts); % 初始化dpn

dbn = dbntrain(dbn, train_x, opts); %训练dpn


%将深度置信网络参数导入神经网络

nn_1 = dbnunfoldtonn(dbn, 10);

nn_1.learningRate = 1;   %nn 学习率

nn_1.activation_function = 'sigm'; %nn激活函数

nn_1.output  = 'softmax'; %输出softmax

nn_1.weightPenaltyL2 = 1e-4; %权重二范数惩罚
```

```
opts.numepochs =  20; %nn epochs
opts.batchsize = 100; %nn batchsize
opts.plot = 1;  %使能画图
%训练 nn
nn_1 = nntrain(nn_1, train_x, train_y, opts);
%测试 nn
[er1, bad1] = nntest(nn_1, test_x, test_y);%测试
fprintf('dpn+nn test error: %f\n', er1);
%% 神经网络(nn)
rand('state',0)
nn_2 = nnsetup([784 100 10]); %784->100->10
nn_2.learningRate = 1;   %nn 学习率
nn_2.activation_function = 'sigm';%nn 激活函数
nn_2.output  = 'softmax'; %输出 softmax
nn_2.weightPenaltyL2 = 1e-4; %权重二范数惩罚
opts.numepochs =  20; %nn epochs
opts.batchsize = 100; %nn batchsize
opts.plot = 1; %使能画图
%训练 nn
[nn_2, L] = nntrain(nn_2, train_x, train_y, opts);
%测试 nn
[er2, bad2] = nntest(nn_2, test_x, test_y);
fprintf('nn test error: %f\n', er2);


%% 堆栈式自编码器(sae)+神经网络(nn)
rand('state',0)
```

```
sae = saesetup([784 100]); %784->100->100->784
sae.ae{1}.activation_function  = 'sigm'; %激活函数
sae.ae{1}.learningRate  = 1;  %学习率
sae.ae{1}.inputZeroMaskedFraction   = 0.5;
opts.numepochs =   3;%epochs
opts.batchsize = 100;%batchsize
%训练 sae
sae = saetrain(sae, train_x, opts);
%设置 nn 结构
nn_3 = nnsetup([784 100 10]); %784->100->10
nn_3.activation_function    = 'sigm';%激活函数
nn_3.learningRate  = 1;%学习率
nn_3.output  = 'softmax'; %输出 softmax
nn_3.weightPenaltyL2 = 1e-4;%权重二范数惩罚
nn_3.W{1} = sae.ae{1}.W{1};%初始化 nn 权重
opts.numepochs =   20;%nn epochs
opts.batchsize = 100;%nn batchsize
opts.plot = 1; %使能画图
%训练 nn
nn_3 = nntrain(nn_3, train_x, train_y, opts);
%测试 nn
[er3, bad3] = nntest(nn_3, test_x, test_y);
fprintf('nn test error: %f\n', er3);
```

六、实验结果及分析

实验结果：

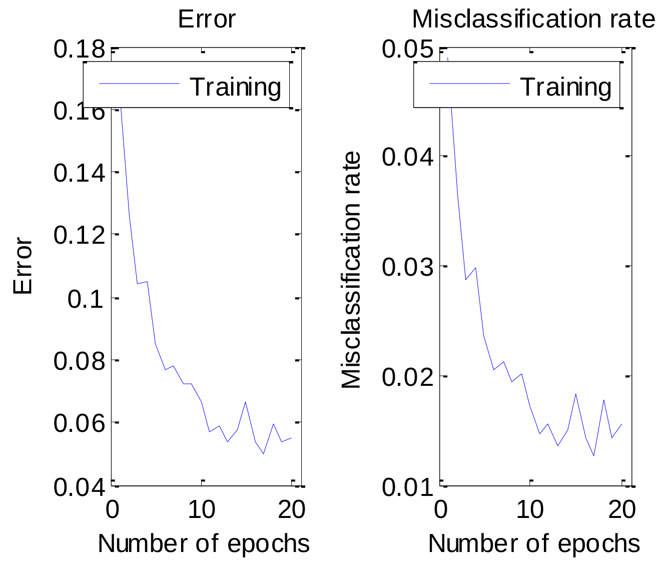


图 5-1 dpn+nn 误差与误分类比率

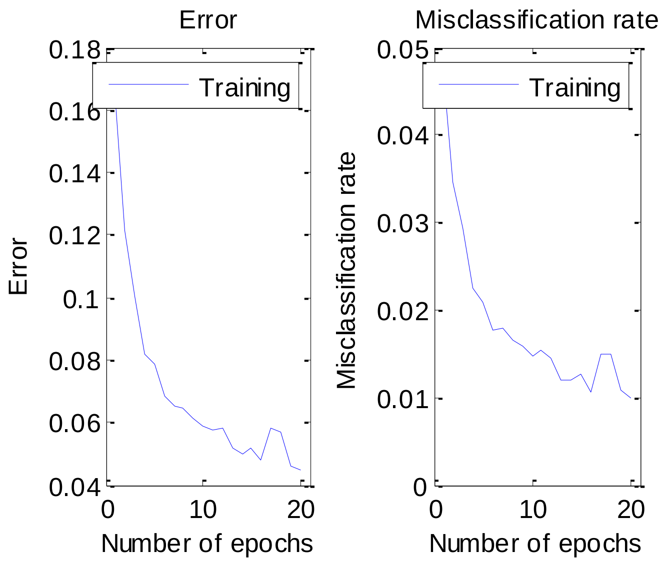


图 5-2 nn 误差与误分类比率

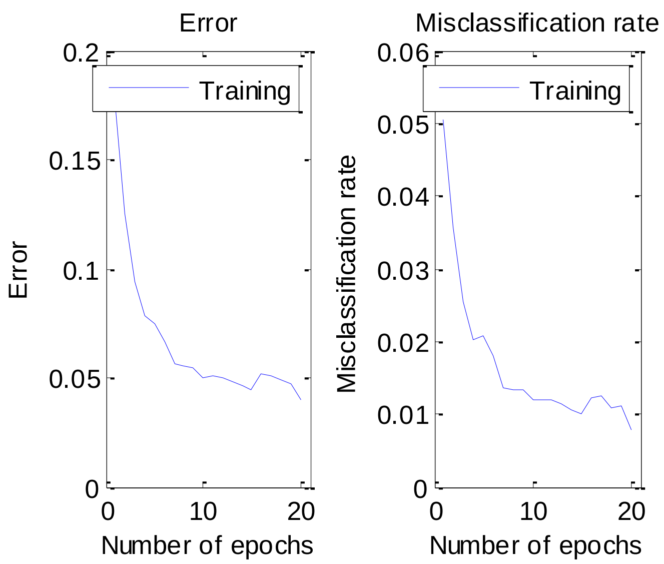


图 5-3 sae+nn 误差与误分类比率

```
epoch 1/20. Took 2.4825 seconds. Mini-batch mean squared error on training set is 0.28216; Full-batch train err = 0.166694
epoch 2/20. Took 2.8462 seconds. Mini-batch mean squared error on training set is 0.15581; Full-batch train err = 0.126148
epoch 3/20. Took 2.9725 seconds. Mini-batch mean squared error on training set is 0.12544; Full-batch train err = 0.104003
epoch 4/20. Took 2.9385 seconds. Mini-batch mean squared error on training set is 0.10776; Full-batch train err = 0.104713
epoch 5/20. Took 3.1018 seconds. Mini-batch mean squared error on training set is 0.096161; Full-batch train err = 0.085009
epoch 6/20. Took 3.213 seconds. Mini-batch mean squared error on training set is 0.088584; Full-batch train err = 0.076568
epoch 7/20. Took 3.951 seconds. Mini-batch mean squared error on training set is 0.082398; Full-batch train err = 0.077968
epoch 8/20. Took 2.9542 seconds. Mini-batch mean squared error on training set is 0.078115; Full-batch train err = 0.072004
epoch 9/20. Took 3.1459 seconds. Mini-batch mean squared error on training set is 0.074146; Full-batch train err = 0.072195
epoch 10/20. Took 3.1316 seconds. Mini-batch mean squared error on training set is 0.071982; Full-batch train err = 0.066553
epoch 11/20. Took 3.3366 seconds. Mini-batch mean squared error on training set is 0.068921; Full-batch train err = 0.057110
epoch 12/20. Took 3.7648 seconds. Mini-batch mean squared error on training set is 0.067677; Full-batch train err = 0.058806
epoch 13/20. Took 4.4059 seconds. Mini-batch mean squared error on training set is 0.065425; Full-batch train err = 0.053850
epoch 14/20. Took 2.8151 seconds. Mini-batch mean squared error on training set is 0.06333; Full-batch train err = 0.057363
epoch 15/20. Took 2.6727 seconds. Mini-batch mean squared error on training set is 0.062031; Full-batch train err = 0.066521
epoch 16/20. Took 2.6466 seconds. Mini-batch mean squared error on training set is 0.060431; Full-batch train err = 0.053968
epoch 17/20. Took 2.78 seconds. Mini-batch mean squared error on training set is 0.05943; Full-batch train err = 0.049654
epoch 18/20. Took 2.7932 seconds. Mini-batch mean squared error on training set is 0.058535; Full-batch train err = 0.059247
epoch 19/20. Took 2.7678 seconds. Mini-batch mean squared error on training set is 0.056265; Full-batch train err = 0.053624
epoch 20/20. Took 2.7681 seconds. Mini-batch mean squared error on training set is 0.055364; Full-batch train err = 0.054915
dpn+nn test error: 0.029400
```

图 5-4 dpn+nn 训练过程

```
epoch 1/20. Took 2.1938 seconds. Mini-batch mean squared error on training set is 0.35166; Full-batch train err = 0.165940
epoch 2/20. Took 2.5484 seconds. Mini-batch mean squared error on training set is 0.14813; Full-batch train err = 0.121565
epoch 3/20. Took 2.4642 seconds. Mini-batch mean squared error on training set is 0.11584; Full-batch train err = 0.100694
epoch 4/20. Took 2.4078 seconds. Mini-batch mean squared error on training set is 0.099243; Full-batch train err = 0.081632
epoch 5/20. Took 2.2141 seconds. Mini-batch mean squared error on training set is 0.087391; Full-batch train err = 0.078896
epoch 6/20. Took 2.3283 seconds. Mini-batch mean squared error on training set is 0.079806; Full-batch train err = 0.068298
epoch 7/20. Took 2.5067 seconds. Mini-batch mean squared error on training set is 0.074294; Full-batch train err = 0.065381
epoch 8/20. Took 2.3676 seconds. Mini-batch mean squared error on training set is 0.070568; Full-batch train err = 0.064882
epoch 9/20. Took 2.2601 seconds. Mini-batch mean squared error on training set is 0.066835; Full-batch train err = 0.061136
epoch 10/20. Took 2.3229 seconds. Mini-batch mean squared error on training set is 0.063783; Full-batch train err = 0.058926
epoch 11/20. Took 2.4504 seconds. Mini-batch mean squared error on training set is 0.062173; Full-batch train err = 0.057399
epoch 12/20. Took 2.3015 seconds. Mini-batch mean squared error on training set is 0.060317; Full-batch train err = 0.058347
epoch 13/20. Took 2.2348 seconds. Mini-batch mean squared error on training set is 0.058493; Full-batch train err = 0.051910
epoch 14/20. Took 2.2538 seconds. Mini-batch mean squared error on training set is 0.056612; Full-batch train err = 0.050028
epoch 15/20. Took 2.3102 seconds. Mini-batch mean squared error on training set is 0.055397; Full-batch train err = 0.051826
epoch 16/20. Took 2.2153 seconds. Mini-batch mean squared error on training set is 0.055068; Full-batch train err = 0.048136
epoch 17/20. Took 2.2668 seconds. Mini-batch mean squared error on training set is 0.053751; Full-batch train err = 0.058461
epoch 18/20. Took 2.3734 seconds. Mini-batch mean squared error on training set is 0.053328; Full-batch train err = 0.056756
epoch 19/20. Took 2.3422 seconds. Mini-batch mean squared error on training set is 0.052669; Full-batch train err = 0.045832
epoch 20/20. Took 2.2247 seconds. Mini-batch mean squared error on training set is 0.051513; Full-batch train err = 0.044874
nn2 test error: 0.021800
```

图 5-5 nn 训练过程

```
epoch 1/20. Took 2.4216 seconds. Mini-batch mean squared error on training set is 0.31451; Full-batch train err = 0.177794
epoch 2/20. Took 2.2856 seconds. Mini-batch mean squared error on training set is 0.15111; Full-batch train err = 0.124919
epoch 3/20. Took 2.2826 seconds. Mini-batch mean squared error on training set is 0.11302; Full-batch train err = 0.094451
epoch 4/20. Took 2.345 seconds. Mini-batch mean squared error on training set is 0.093012; Full-batch train err = 0.078179
epoch 5/20. Took 2.2932 seconds. Mini-batch mean squared error on training set is 0.080271; Full-batch train err = 0.074887
epoch 6/20. Took 2.2345 seconds. Mini-batch mean squared error on training set is 0.073044; Full-batch train err = 0.066722
epoch 7/20. Took 2.266 seconds. Mini-batch mean squared error on training set is 0.067698; Full-batch train err = 0.056641
epoch 8/20. Took 2.3143 seconds. Mini-batch mean squared error on training set is 0.06375; Full-batch train err = 0.055124
epoch 9/20. Took 2.2022 seconds. Mini-batch mean squared error on training set is 0.060857; Full-batch train err = 0.054823
epoch 10/20. Took 2.1973 seconds. Mini-batch mean squared error on training set is 0.05862; Full-batch train err = 0.050378
epoch 11/20. Took 2.3348 seconds. Mini-batch mean squared error on training set is 0.057182; Full-batch train err = 0.051022
epoch 12/20. Took 2.2964 seconds. Mini-batch mean squared error on training set is 0.055806; Full-batch train err = 0.049921
epoch 13/20. Took 2.2325 seconds. Mini-batch mean squared error on training set is 0.053882; Full-batch train err = 0.048574
epoch 14/20. Took 2.2744 seconds. Mini-batch mean squared error on training set is 0.053523; Full-batch train err = 0.046597
epoch 15/20. Took 2.3267 seconds. Mini-batch mean squared error on training set is 0.051853; Full-batch train err = 0.044199
epoch 16/20. Took 2.2403 seconds. Mini-batch mean squared error on training set is 0.050871; Full-batch train err = 0.051487
epoch 17/20. Took 2.224 seconds. Mini-batch mean squared error on training set is 0.050324; Full-batch train err = 0.050597
epoch 18/20. Took 2.5556 seconds. Mini-batch mean squared error on training set is 0.049967; Full-batch train err = 0.048841
epoch 19/20. Took 2.323 seconds. Mini-batch mean squared error on training set is 0.049382; Full-batch train err = 0.046981
epoch 20/20. Took 2.2311 seconds. Mini-batch mean squared error on training set is 0.049203; Full-batch train err = 0.039962
sae+nn test error: 0.020500
```

图 5-6 sae+nn 训练过程

可以看到在网络结构等超参相同的情况下，sae+nn 提供的初始化参数收敛更好，nn 其次，dpn+nn 是最差的。

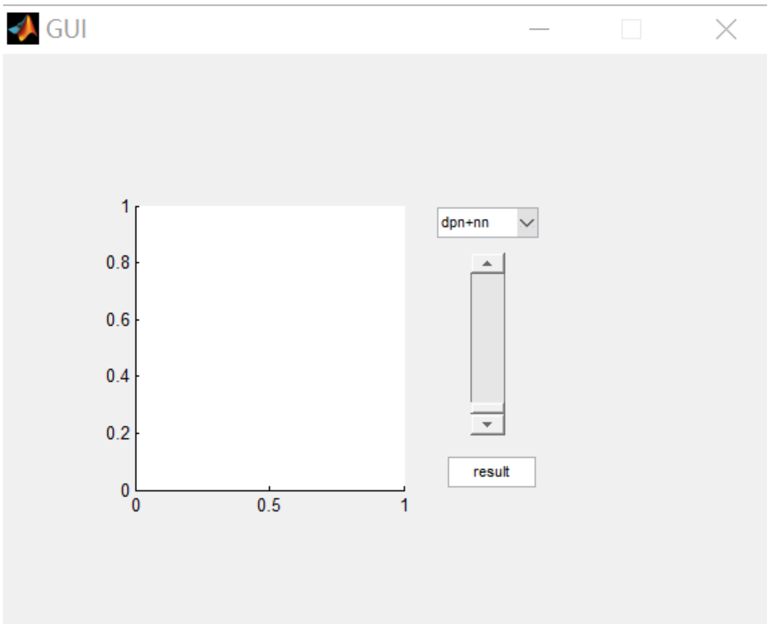


图 5-7 GUI 设计界面（根据滑条值自动识别）

实验结果（dpn+nn）：

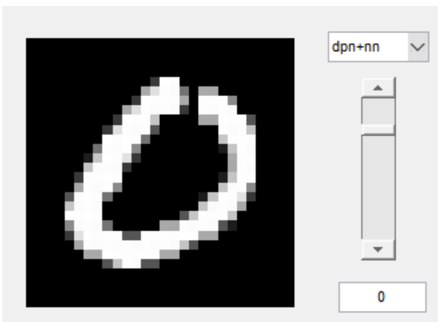


图 5-8 dpn+nn 识别 0

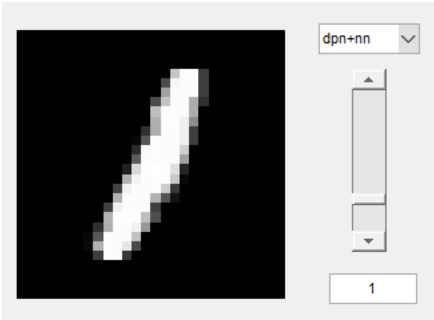


图 5-9 dpn+nn 识别 1

图 5-10 dpn+nn 识别 2

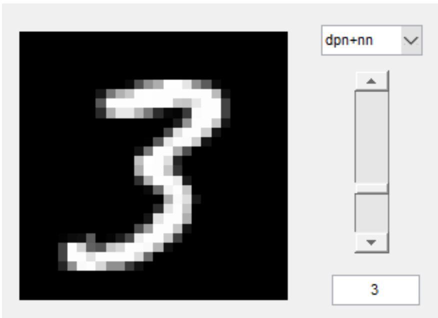


图 5-11 dpn+nn 识别 3

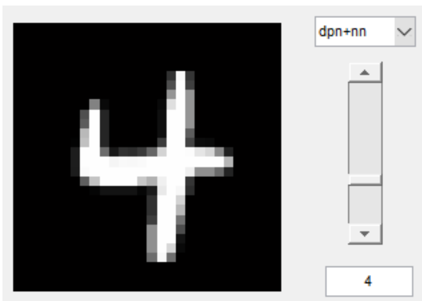


图 5-12 dpn+nn 识别 4

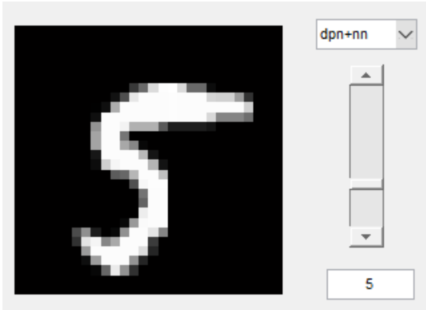


图 5-13 dpn+nn 识别 5

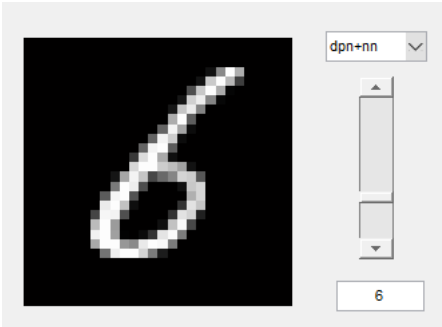


图 5-14 dpn+nn 识别 6

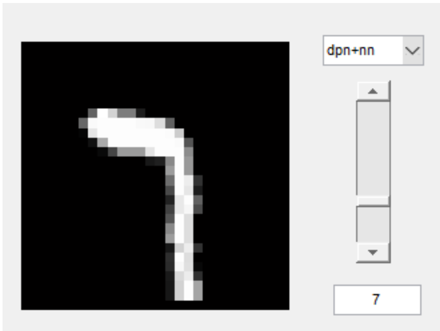


图 5-15 dpn+nn 识别 7

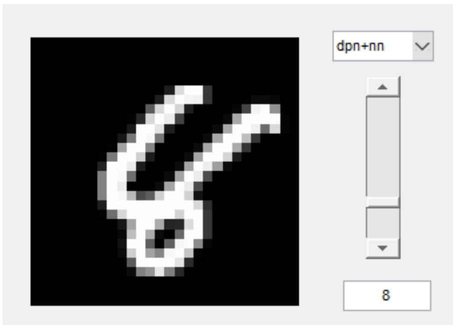


图 5-16 dpn+nn 识别 8

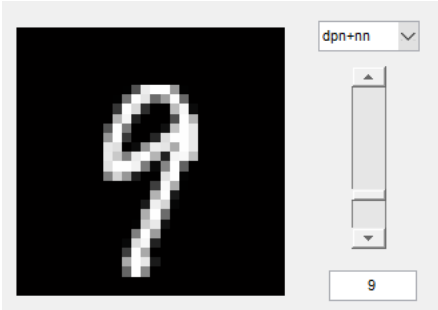


图 5-17 dpn+nn 识别 9

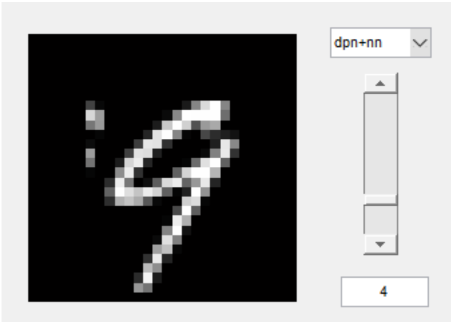


图 5-18 dpn+nn 识别 9 错误

实验结果（nn）：

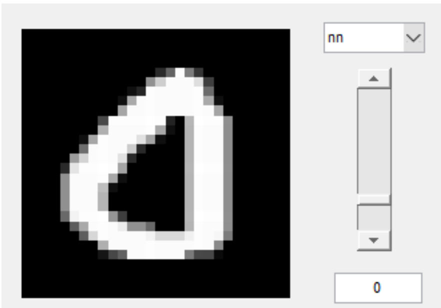


图 5-19 nn 识别 0

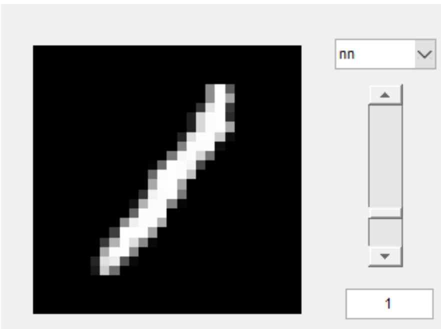


图 5-20 nn 识别 1

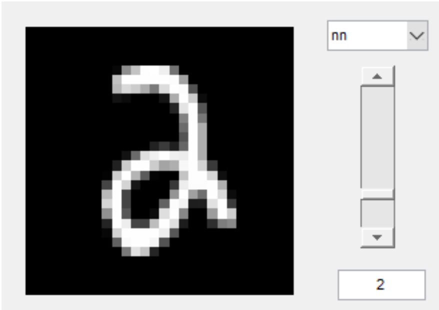


图 5-21 nn 识别 2

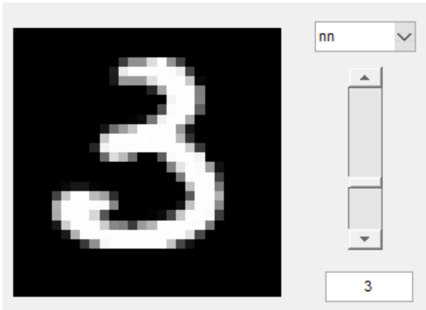


图 5-22 nn 识别 3

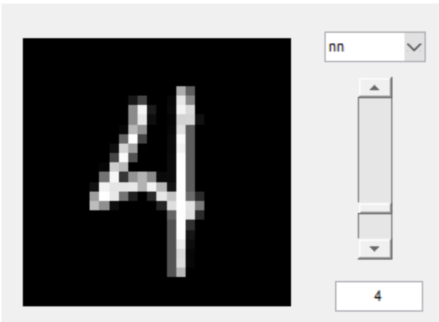


图 5-23 nn 识别 4

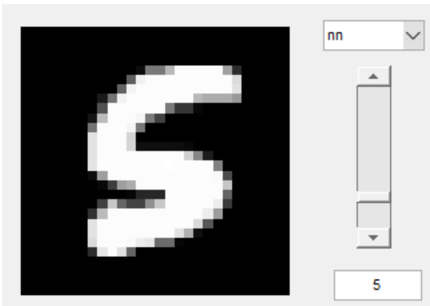


图 5-24 nn 识别 5

实验结果（sae+nn）：

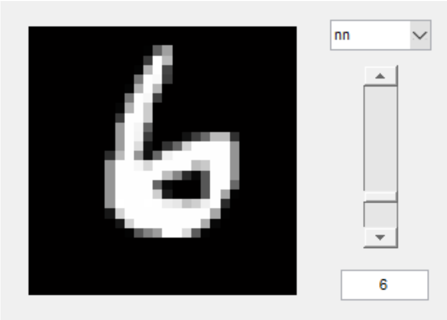


图 5-25 nn 识别 6

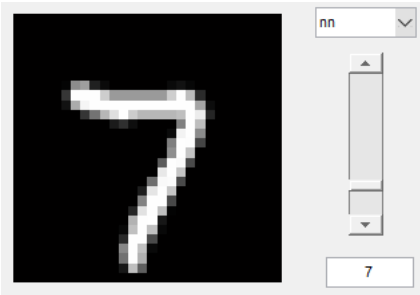


图 5-26 nn 识别 7

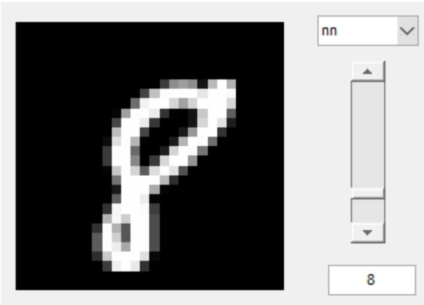


图 5-27 nn 识别 8

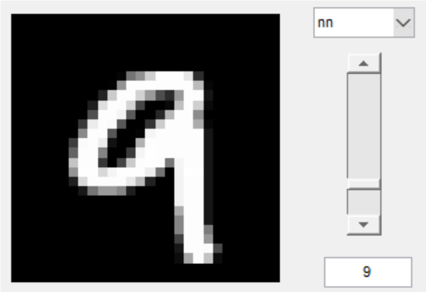


图 5-28 nn 识别 9

实验结果（sae+nn）：

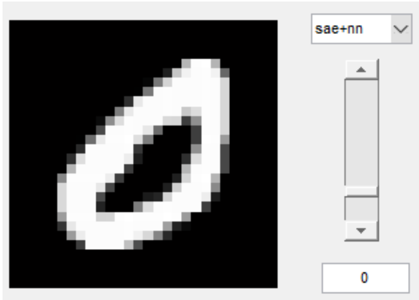


图 5-29 sae+nn 识别 0

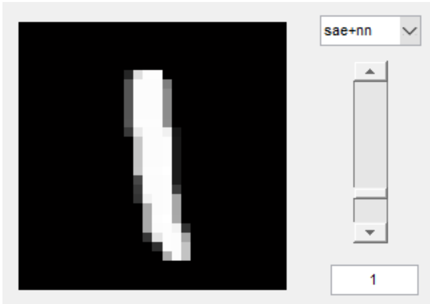


图 5-30 sae+nn 识别 1

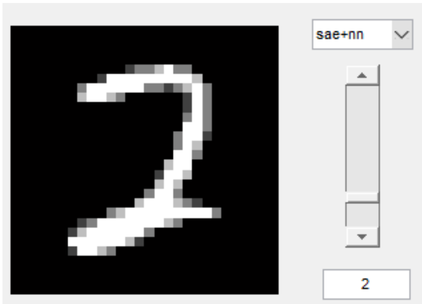


图 5-31 sae+nn 识别 2

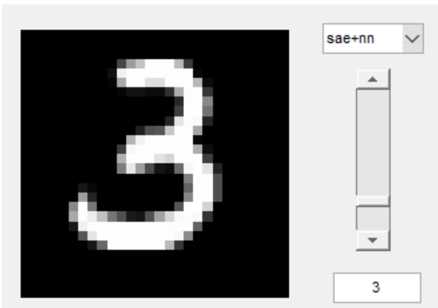


图 5-32 sae+nn 识别 3

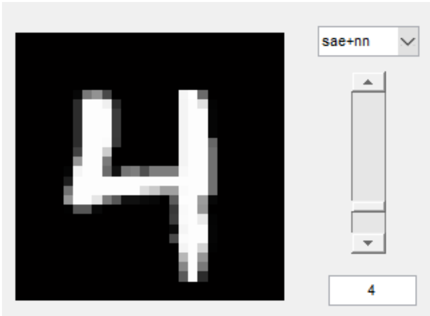


图 5-33 sae+nn 识别 4

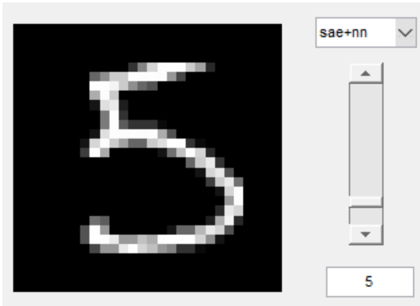


图 5-34 sae+nn 识别 5

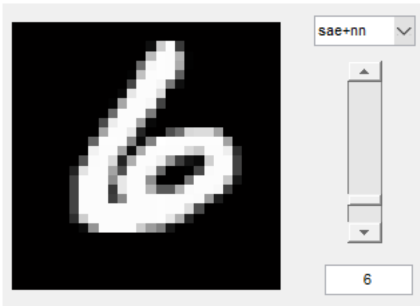


图 5-35 sae+nn 识别 6

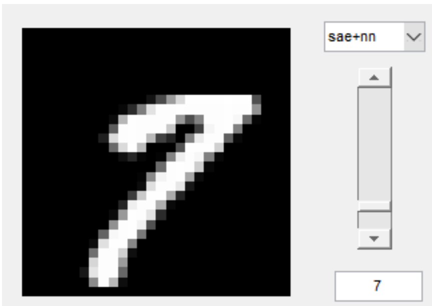


图 5-36 sae+nn 识别 7

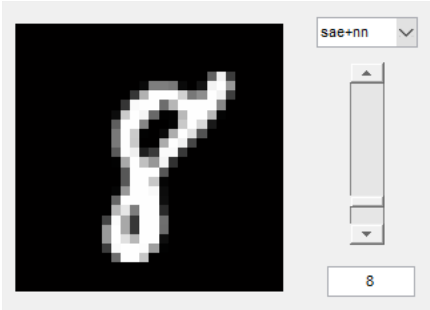


图 5-37 sae+nn 识别 8

图 5-38 sae+nn 识别 9

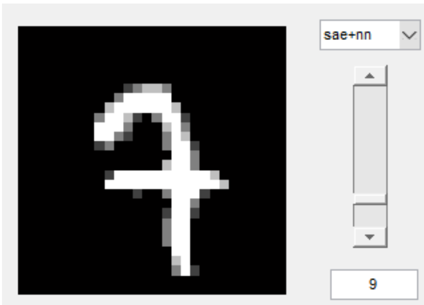


图 5-39 sae+nn 识别 7 错误

七、遇到的问题及解决的过程

主要遇到的问题是要学习 GUI 的编写，期间遇到一个排查很久的问题，将这一章节的代码包装成函数，用于 GUI 程序的调用，由于没有去除 clear 函数，导致在 GUI 界面遇到了很奇怪的问题，程序直接就退出了，没有报错，一步步调试后才发现这个原因，找了很久。

成 绩	