北京航空航天大學
BEIHANG UNIVERSITY

模式识别大作业

院(系)名称	高等理工学院
学　　号	16231235
姓　　名	李谨杰
指导教师	李阳

2019 年 4 月

目录

实验一　感知器，SVM学习与分类

一、实验目的

利用 MATLAB，Python 或者其它编程语言，实现用感知器、SVM(支持向量机)对二维样本的分类。我选用 MATLAB

二、实验要求

以 P 作为给定训练样本数据，T 作为训练数据的标签，使用感知器、SVM 两种方法训练分类模型。以 Q 作为测试数据，检测训练好的模型的性能。

P=[1 2 2 1 3 5 3 4 1 2 4；2 1 2 4 3 4 2 3 1 3 2];

T = [-1 -1 -1 1 1 1 1 1 -1 -1 1];

Q=[1 1 5 5; 1.5 3 3 4];

三、实验原理

1. 感知器

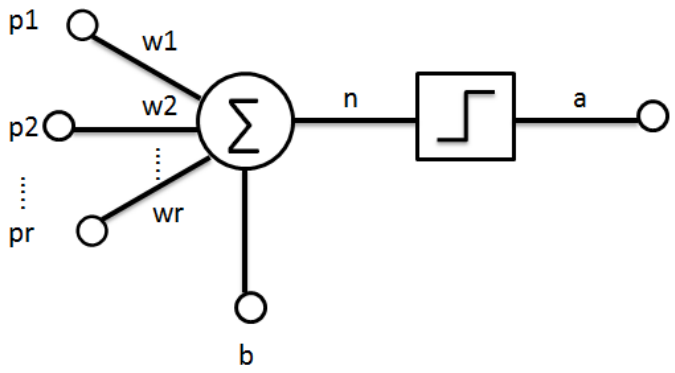


图 1.3.1 感知器模型

1958 年，两层神经元组成的感知器模型由计算科学家 Rosenblatt 提出。每一个输入分量通过一个权值分量进行加权求和，并作为阈值函数的输入。激活函数一般选用 hardlim()函数，即阈值函数：

$$a = f(n) = \text{hardlim}(n) = \begin{cases} 1 & (n \geq 0) \\ 0 & (n < 0) \end{cases}$$

即：

$$a = \text{hard}\lim\left(p_1 \bullet w_1 + p_2 \bullet_{W_2} + \ldots + p_r \bullet_{W_r} + b\right)$$

其中 a 为实际输出，pi 为第 i 个输入，b 为偏差，表征训练结果对下一次迭代的影响。

权值调整选用 learnp 函数。定义误差 e=t-a，其中 t 为目标输出，a 为实际输出，训练的目标为使 e=0；每一步学习过程对神经元的调整算法为：

$$w(k+1) = w(k) + e \cdot p^T$$

$$b(k+1)=b(k)+e$$

其中 e 为误差，w(k)为 k 时刻的权值向量。该算法的核心为：权值的变化量等于正负输入矢量。由于 e 的取值只有 0，1 或-1，则权重的调整较为剧烈，可以看到在之后的仿真中随训练次数增加，MAE 有较大振荡。

另外，如果输入数据的量级差距太大，则会造成上述公式无法收敛，此时可以用 learnpn()函数进行归一化，之后再进行训练。

2. 线性不可分支持向量机 SVM

对于线性可分数据集，SVM 的基本思路是试图找到一个超平面

$$g(x)=w^Tx+b, w, x\in R^d, b\in R$$

使得 $w^Tx+b=1$ 和 $w^Tx+b=-1$ 这两条线之间的间隔最大。

根据超平面的几何意义，我们可以定义间隔 γ 为 $\gamma=\frac{2}{\|w\|}$

为使 γ 最大，我们可以转换为求

$$\min_{w,b}\frac{1}{2}\|w\|^2 \qquad \text{s.t.} \quad y_i\left(w^Tx_i+b\right)\geq 1 \quad i=1,2,\cdots,m$$

应用拉格朗日乘子法，求存在约束条件的极值。定义拉格朗日函数：

$$L(w,b,\alpha)=\frac{1}{2}\|w\|^2+\sum_{i=1}^{m}\alpha_i\left(1-y_i\left(w^Tx_i+b\right)\right)$$

上式对 w、b 求导有：$w=\sum_{i=1}^{m}\alpha_i y_i x_i$，$\sum_{i=1}^{m}\alpha_i y_i=0$。代入上式：

$$L(\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

之后一般使用 SMO（Sequential Minimal Optimization）方法求 L(α)的最大值。SMO 的基本步骤为：选取一对需要更新的变量，固定这两个变量以外的参数，求解优化问题：在限制条件 $\alpha_i y_i+\alpha_j y_j=c=-\sum_{k=1;k\neq i,j}^{m}\alpha_k y_k$ 下，s. t.

$$\max_{\alpha} \mathrm{L}(\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

优化得到 α 的最优解后，代入 w、b 的公式即可求得超平面。

在求得超平面的过程中，只有那些处于边缘的数据才被使用。这些数据满足 $y_i g\left(x_i\right)=1$，称为支持向量。

对于本题的训练集：

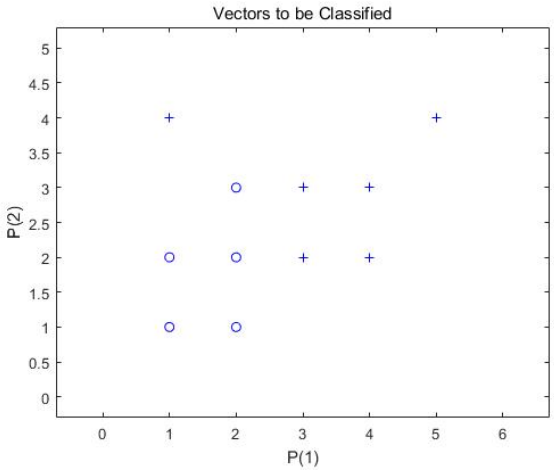


图 1.3.2 训练集分布，圆圈为-1

由训练样本点集分布可知，训练样本为线性不可分样本，故用到线性不可分 SVM 分类器。

线性不可分 SVM 分类器为对线性 SVM 分类器的拓展，核心思想是用一个函数 $\phi(x)$将样本坐标从欧式空间映射到另一个空间（Hilbert 空间），线性不可分数据集在这个新空间变为线性可分的，从而能用线性 SVM 方法求解。在 Hilbert 空间中的内积运算应满足核矩阵 K_M 半正定条件，在这种距离定义下该空间是自洽的。这种内积运算即为核函数。常用的核函数如下图：

常用核函数

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^d$	$d \geqslant 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\sigma}\right)$	$\sigma > 0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)$	tanh 为双曲正切函数, $\beta > 0, \theta < 0$

图 1.3.3 常用核函数

利用核函数内积的性质，可以将原求极值问题转化为 Hilbert 空间中的求极值问题，即满足

$$\max_{\alpha} Q(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i} \bullet \mathbf{x_j})$$

$$\text{s. t.} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \quad 0 \leq \alpha_i \leq C, i = 1, \cdots, n$$

在本实验中通过调用 MATLAB 相关程序包完成 SVM 向量机的分类。

四、实验结果及分析

1. 使用感知器实现对二维样本的分类：

为了找到最好的训练次数，我改变训练次数，得到平均绝对误差 MAE 和均方误差 MSE 随训练次数变化的图像如下：

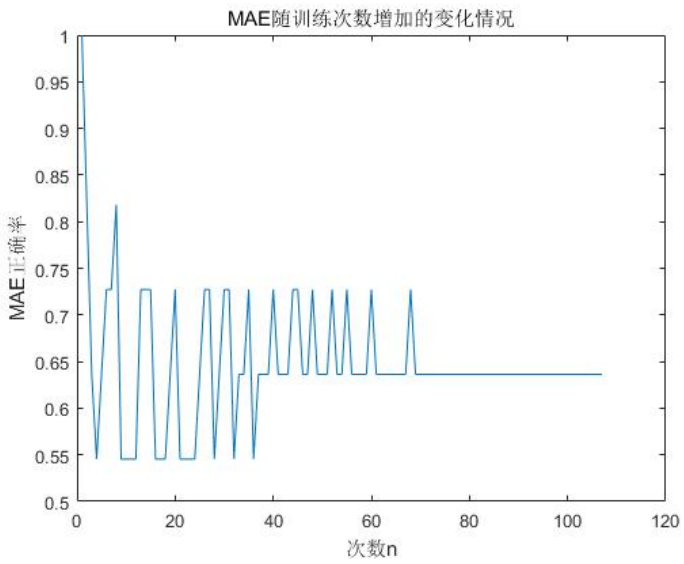


图 1.4.1(a) MAE 随训练次数的变化

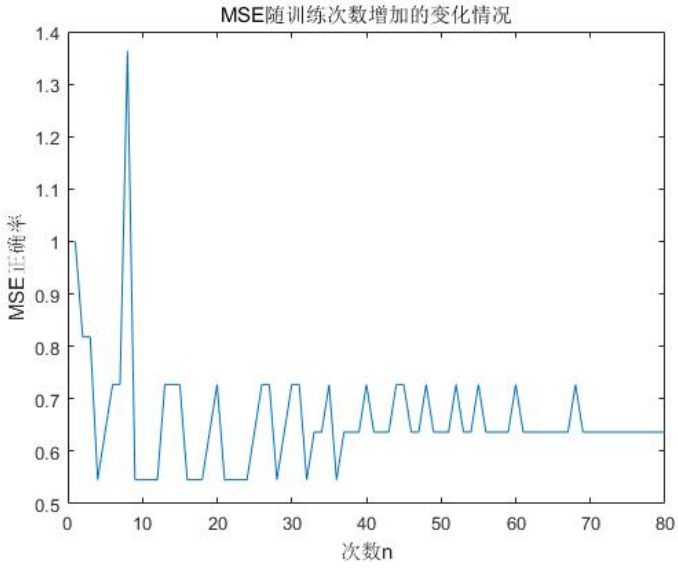


图 1.4.1(b) MSE 随训练次数的变化

由图像可知，刚开始训练误差出现震荡，在 n=4 时第一次达到最小值，在 n=8 时达到峰值，在 n=10 时达到第一段稳定最小误差区间，在 n=70 次时达到稳定正确率。这种振荡也对应着第三节实验原理中对神经元调整算法的分析。将这些步骤下训练的感知器对原样本的分类绘制如下，左边为对训练样本的分类，右边为对测试样本的分类：

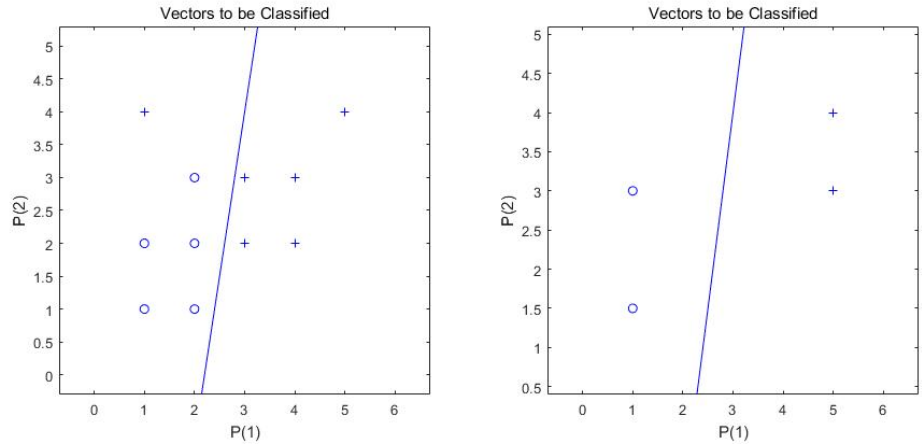


图 1.4.2(a) n=4

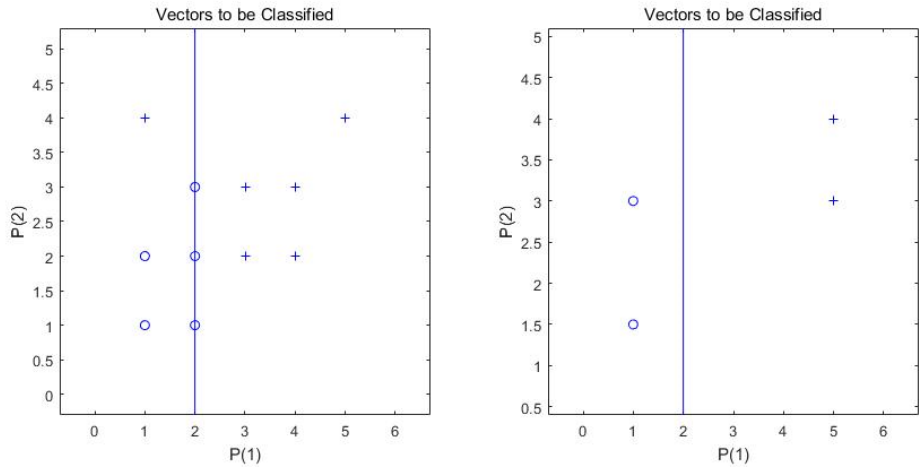


图 1.4.2(b) n=8

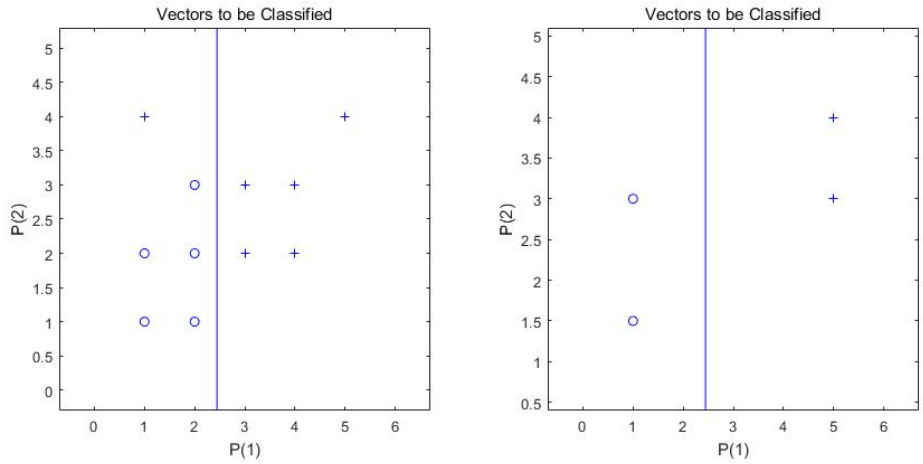


图 1.4.2(c) n=10

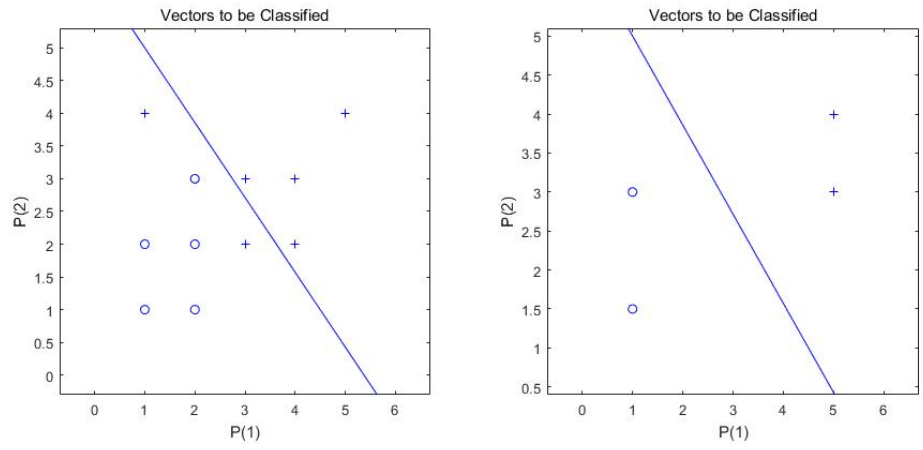


图 1.4.2(d) n=70

由如上实验可以得到如下结论：

（1） 这些分类器对于训练集的分类效果并不相同。因此应该尽可能多地提供测试数据，便于测试分类效果的好坏。

（2） 实验中出现的误差峰值可能是由于分类线与一些样本重合导致的（图 4.2（b））。

（3） 感知器是线性分类器，无法解决非线性问题，因此无法通过训练使得对训

练样本的分类正确率达到 100%。

（4） 当训练次数过多，会出现过拟合的状况，正确率反而会降低（图 4.2（d））。

2.使用线性 SVM 进行样本分类：

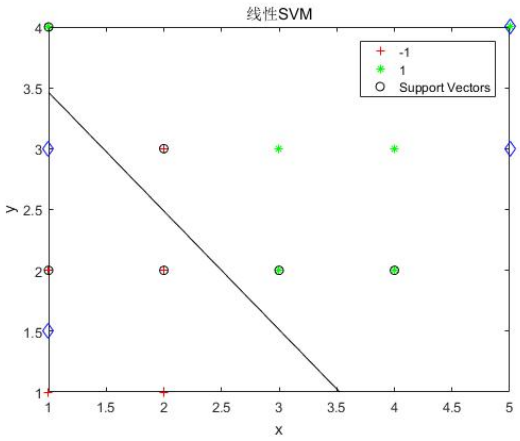
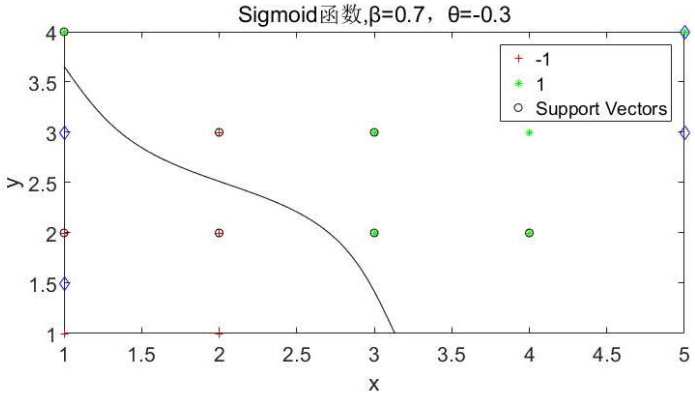


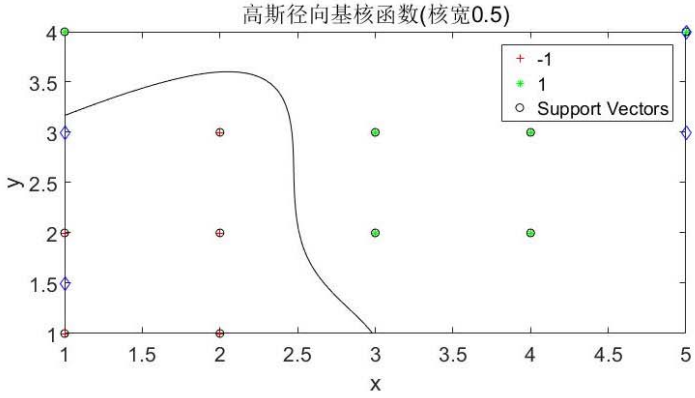
图 1.4.3 线性 SVM 分类结果

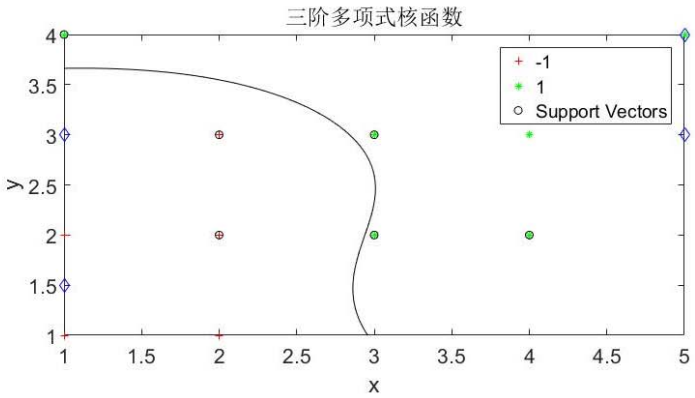
MAE 为 0.1818，错分了一个样本，分类效果与感知器持平。下面改用非线性 SVM 进行分类。

3. 使用 SVM 实现对二维样本的分类：

分别令核函数等于 sigmoid 函数、高斯核函数、多项式核函数和多层感知器核函数，作图如下：







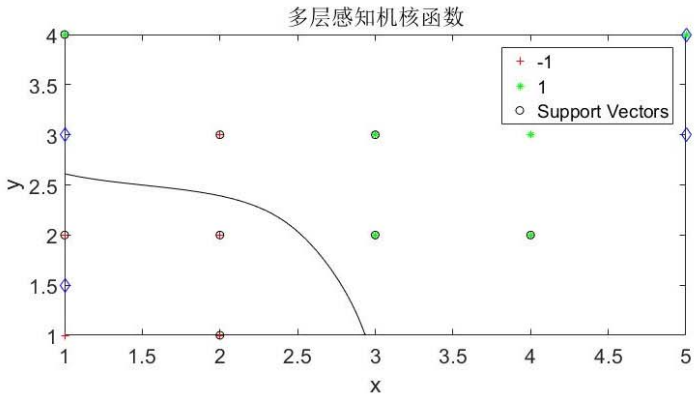


图 1.4.4 几种常见核函数的训练结果

如图，蓝色◇为测试样本。

训练误差 MAE 记录如下：

核函数	Sigmoid	高斯核	三阶多项式	多层感知机
MAE	0.1818	0	0	0.1818

由实验过程可以得到如下结论：

（1） 核函数方法可以比较好的解决线性不可分数据分类问题，仅仅使用一部分数据作为支持向量。

（2） 核函数不同，训练得到的超平面也不同。对于可以将训练集完全分开的几种核函数，难以判断哪个核函数更好。选择核函数时也只能通过试验法，没有通用的核函数查找标准。

（3） 如果使用 Sigmoid 函数作为核函数，需要大量尝试调整参数，最终得到的结构也不够理想。总的来说，对于本题的数据，高斯核函数与多项式核函数的分类结果优于 Sigmoid 函数和多层感知机核函数。

五、实验代码

感知器:

```
clc;clear;
P= [1 2 2 1 3 5 3 4 1 2 4;2 1 2 4 3 4 2 3 1 3 2]; %给定训练样本数据
T= [-1 -1 -1 1 1 1 1 1 -1 -1 1]; %给定样本数据所对应的类别，用1和-1来表示两种类别
%
% for i=1:80
% net=newp([1 5;1 4],1);
% %创建一个有两个输入、样本数据的取值范围已知，并且
% %网络只有一个神经元的感知器神经网络激活函数hardlim（），学习函数learnp（）
% net.trainParam.epochs = i; %设置网络的最大训练次数为20次
% net=train(net,P,T);        %使用训练函数对创建的网络进行训练
% Y=sim(net,P);              %对训练后的网络进行仿真
% E1(i)=mae(Y-T);        %计算网络的平均绝对误差，表示网络错误分类
% E2(i)=mse(Y-T);
% end

%i=find(E2==min(E2));
net=newp([1 5;1 4],1);
%创建一个有两个输入、样本数据的取值范围已知，并且
%网络只有一个神经元的感知器神经网络   激活函数hardlim（），学习函数learnp（）
net.trainParam.epochs = 70; %设置网络的最大训练次数为70次
net=train(net,P,T);        %使用训练函数对创建的网络进行训练
Y=sim(net,P);              %对训练后的网络进行仿真
E1=mae(Y-T);        %计算网络的平均绝对误差，表示网络错误分类
E2=mse(Y-T);
Q=[1 1 5 5;1.5 3 3 4]; %检测训练好的神经网络的性能
Y1=sim(net,Q);         %对网络进行仿真，仿真输出即为分类的结果
```

```
figure;              %创建一个新的绘图窗口
subplot(1,2,1);
T(T==-1)=0;
plotpv(P,T);
plotpc(net.iw{1},net.b{1}) %在坐标图中绘制分类线
subplot(1,2,2);
plotpv(Q,Y1);        %在坐标图中绘制测试数据
plotpc(net.iw{1},net.b{1}) %在坐标图中绘制分类线
```

SVM:

```
function k = kfun(u,v,p1,p2)
    %%sigmoid核函数
k = tanh(p1*(u*v')+p2);
end
clc; clear;
x=[1 2 2 1 3 5 3 4 1 2 4];
y=[2 1 2 4 3 4 2 3 1 3 2];
testdata=[1,1.5;1,3;5,3;5,4];

data=zeros(11,2);% (x,y)构成的数据点
data(:,1)=x;
data(:,2)=y;
groups=[-1  -1  -1  1  1  1  1  1  -1  -1  1]';%各个数据点的标签
figure;
subplot(2,2,1);
p1=0.7;p2=-0.3;   %p1越大，弯曲越大;p1>0,p2<0
Struct1 = svmtrain(data,groups,'Kernel_Function',@(u,v)
kfun(u,v,p1,p2), 'showplot',true);%利用自定义核函数进行训练
classes1=svmclassify(Struct1,data,'showplot',false);%data数据分类，并显
示图形
title('Sigmoid函数,β=0.7, θ=-0.3');
hold on;
plot(testdata(:,1),testdata(:,2),'bd','MarkerSize',8); %绘制训练数据
xlabel('x');
ylabel('y');
CorrectRate1=mae(classes1-groups);  %计算训练误差

subplot(2,2,2);
Struct2 = svmtrain(data,groups,'Kernel_Function','rbf',
'RBF_Sigma',0.5,'showplot',true);
classes2=svmclassify(Struct2,data,'showplot',false);
title('高斯径向基核函数(核宽0.5)');
hold on;
plot(testdata(:,1),testdata(:,2),'bd','MarkerSize',8);
```

```
xlabel('x');
ylabel('y');
CorrectRate2=mae(classes2-groups);
subplot(2,2,3);
Struct3 = svmtrain(data,groups,'Kernel_Function','polynomial',
'showplot',true);
classes3=svmclassify(Struct3,data,'showplot',false);
title('三阶多项式核函数');
hold on;
plot(testdata(:,1),testdata(:,2),'bd','MarkerSize',8);
xlabel('x');
ylabel('y');
CorrectRate3=mae(classes3-groups);
subplot(2,2,4);
Struct4 = svmtrain(data,groups,'Kernel_Function','mlp',
'showplot',true);
classes4= svmclassify(Struct4,data,'showplot',false);
title('多层感知机核函数');
hold on;
plot(testdata(:,1),testdata(:,2),'bd','MarkerSize',8);
xlabel('x');
ylabel('y');
CorrectRate4=mae(classes4-groups);
```

六、收获、体会及建议

经过这次小实验，我将老师讲的知识与实际操作结合，初步掌握了感知器与 SVM 支持向量机的知识，锻炼了 MATLAB 编程能力，为之后进一步学习神经网络打下基础。

实验二　BP网络学习与分类

一、实验目的

学习 BP 神经网络的构建和训练。

二、实验要求

数据集 1：MNIST 手写数字(链接 http://yann.lecun.com/exdb/mnist/) 。训练集 60000 张黑白图(28×28),测试集 10000 张黑白图片（28×28),一共十类。

数据集 2：CIFAR-10 微小图像数据集(链接 http://www.cs.toronto.edu/~kriz/cifar.html).训练集 60000 张彩色图（32×32），一共十类。测试集 10000 张彩色图（32×32），一共十类。

数据集二选一；构建 BP 网络对其进行训练；用训练好的网络对测试集进行分类并对结果进行统计分析。

三、实验原理

本实验选用 MNIST 数据集。

1. MNIST 数据集的读取

MNIST 数据集来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST)。训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50% 是高中学生, 50% 来自人口普查局 (the Census Bureau) 的工作人员。测试集(test set) 也是同样比例的手写数字数据。

其中以 train-images-idx3-ubyte 数据为例，该数据包的内容排布如下图：

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
....			
xxxx	unsigned byte	??	pixel

图 2.3.1 train-images-idx3-ubyte 数据包排布

因此用 matlab 读取时，先正确读取前 12 位进行参数设置，再进行数据的连续读取。我学习网上已有的读取代码，添加注释罗列如下：

```
function images = loadMNISTImages(filename)
%载入MNIST数据集，返回一个矩阵集合，每个矩阵是28*28的，数量和MNIST一样
fp = fopen(filename, 'rb'); %打开文件
assert(fp ~= -1, ['Could not open ', filename, '']); %报错

magic = fread(fp, 1, 'int32', 0, 'ieee-be');
assert(magic == 2051, ['Bad magic number in ', filename, '']);

numImages = fread(fp, 1, 'int32', 0, 'ieee-be'); %读取前边几位信息
numRows = fread(fp, 1, 'int32', 0, 'ieee-be');
numCols = fread(fp, 1, 'int32', 0, 'ieee-be');
```

```
images = fread(fp, inf, 'unsigned char');  %按uint8读入灰阶，0-255
images = reshape(images, numCols, numRows, numImages);  %重构成图像形式
的矩阵
images = permute(images,[2 1 3]);
fclose(fp);

% 将数据重新构建成向量形式，一个图一列向量，便于作为神经网络的输入
images = reshape(images, size(images, 1) * size(images, 2),
size(images, 3));
% 将灰度从整形变为double型，同时起到归一化作用，便于处理
images = double(images) / 255;
end
```

读取的 label 的代码类似，附在第五节。

2. BP 神经网络：

基本原理：BP 神经网络是一种多层的前馈神经网络，主要特点是信号前向传播，误差反向传播。通过输入向量与权向量加权求和，得到一个下层神经元的输入，再加上偏置，经过激活函数计算，作为下层神经元的输出。网络具体结构如图 2.3.1。

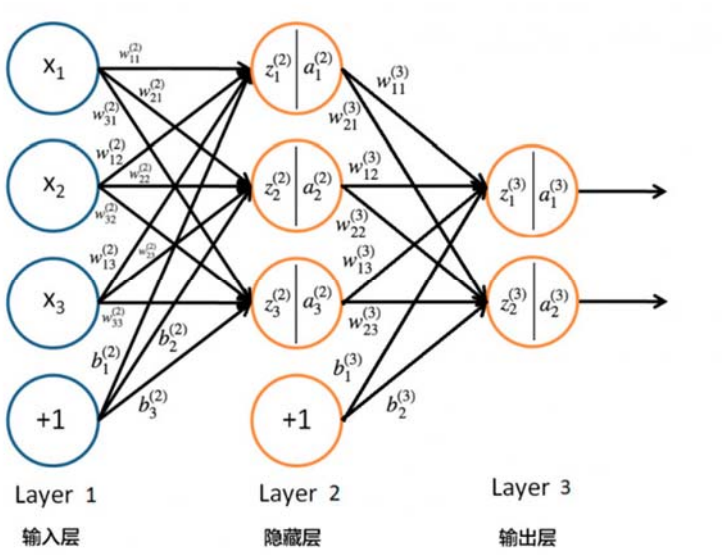


图 2.3.1 三层 BP 神经网络结构

在深度学习出现前，常用三层 BP 神经网络结构。

误差的反向传播过程介绍如下：

首先介绍损失函数 J(w):

$$J(w)=\frac{1}{2}\| t-z\|^2$$

其中 t 为期望输出，z 为实际输出。令 w 表示网络中的权值，由梯度下降法计算 w 的调整值：

$$\Delta w=-\eta\frac{\partial J}{\partial w}$$

其中η为梯度下降的步长。由公式可以看出，步长越大，损失函数下降越快，但容易在极小值附近震荡；步长过小，则会造成收敛过慢。

之后基于链式法则，经过复杂的推导，最终得到隐含层到输出层的权值更新规则：

$$\Delta \mathrm{w}_{kj} = \eta \delta_k y_j = \eta \left(t_k - \mathrm{z}_k \right) f^{'} \left(\mathrm{net}_k \right) \mathrm{y}_j$$

同时得出输入层到隐含层权值的更新规则为：

$$\Delta \mathrm{w}_{ji} = \eta x_i \delta_j = \eta \left[\sum_{k=1}^{c} w_{kj} \delta_k \right] f^{'} \left(net_j \right) x_i$$

具体迭代步骤如下：

（1） 随机产生权值 w1，w2，初始化所用变量，设置神经层数及单元数。

（2） 进行前向求解，获得实际输出值，选取激活函数。一般激活函数常用 sigmoid 函数。

（3） 进行 backward propagation 算法，依据上述公式对权值进行调整

（4） 循环 2、3 步，直到损失（常用均方误差 MSE）达到要求。

3. 关于计算速度

在神经网络的训练中，计算速度非常重要。我一开始只会用 CPU 计算，当我把 60000 个样本输进去时，matlab 卡在了第一次迭代中。于是我将 60000 个数据分成 600 组，每组 100 个，单独训练五次。但即使这样，每训练一组也需要五六分钟，共需要几个小时。后边我开始用 GPU 进行计算，速度提高特别多。但 MATLAB R2016a 无法将 single 数据运用于 GPU，还没有发挥 GPU 最大的运算速度。我选择训练函数为'trainscg'，也由于这个函数可以交由 GPU 运算，而默认函数无法做到。

四、实验结果及分析

1. 关于神经网络的输出层，我一开始只设置了一个输出，将 0-9 通过除以十换算到 0-1 之间经历训练，发现结果在 60%左右。之后参考网上的思路将输出节点改为 10 个，结果提升了很多。结果列表如下：

输出节点数 1			输出节点数 10		
隐层节点数	训练次数	正确率	隐层节点数	训练次数	正确率
20	500	56.6%	25	300	91.71%
25	500	64.37%	25	400	92.94%
25	600	66.11%	25	500	93.88%
25	700	63.62%	25	600	92.31%

经过思考，造成这种的差异的原因是：将数字变为 10 个输出，每个输出对应一个数字，符合的输出取 1 其他取 0，这样实际上是放大了样本之间的差异，加速了网络的训练；而将数字变为一个 0-1 的输出，则缩小了样本之间的差异。因此，相似隐层结构和迭代次数下，十个输出节点要远远好于一个节点的效果。这提醒我们，在进行神经网络训练时，要注意拉大样本间的差异，便于快速训练。

2. 将 25 隐含层节点，10 个输出层节点的训练数据作图如下：

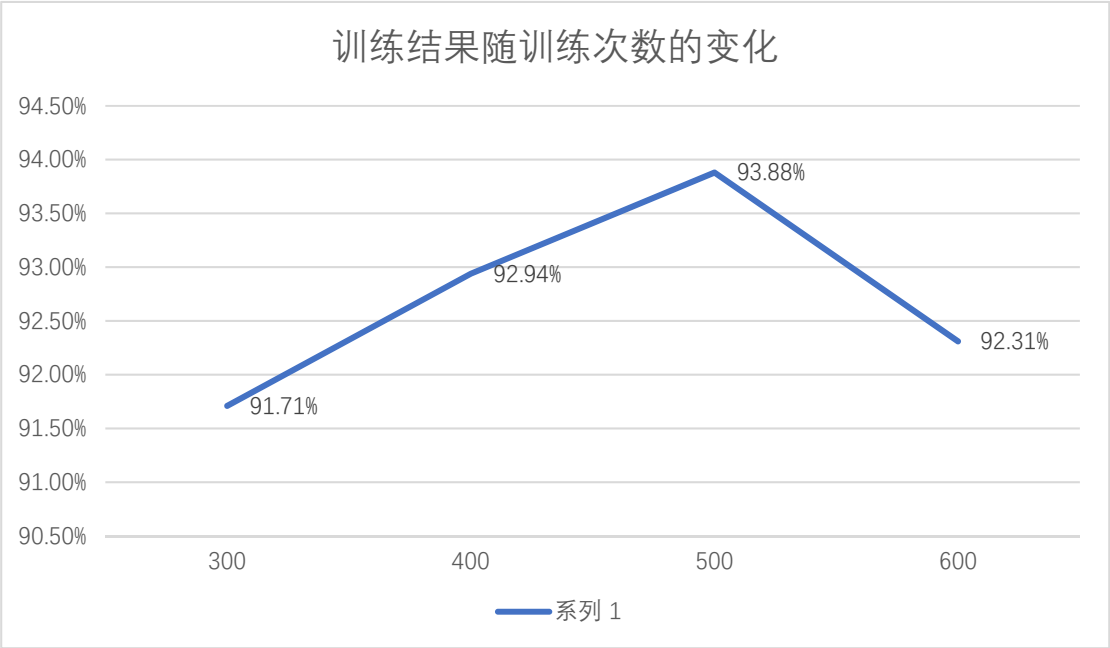


图 2.3.2 10 个输出节点，25 隐含层节点下训练结果随训练次数的变化

由图可知，神经网络的训练存在过拟合情况。如果对训练数据的迭代次数过多，反而可能造成分类准确性的降低。针对具体问题进行网络训练时，应该更改训练次数进行测试，找到最佳训练次数。

3. 针对同样的三层网络，十个输出节点，改变隐含层节点数，训练网络结果如下：

表一

隐层节点数	训练次数	正确率
25	500	93.88%
100	500	94.14%
150	600	95.53%
200	800	96%+

表二

隐层节点数	训练次数	正确率
100	500	94.14%
100	600	95.04%
100	700	96.15%

注：由于计算机计算能力的限制，无法将每个网络结构的最佳正确率训练出来。但加大网络节点，由表一，可以看出层数不变时，正确率随节点数增加呈现上升趋势。原因是节点增多以后，可以更好地模拟复杂映射关系，但相应的训练最佳效果所需的训练次数也要上升。例如，对于单层 25 节点的网络训练到 500 次达到峰值 93.88%，但对于单层 100 次的网络，由表二，显然 700 次还未达到峰值。故可以说明训练最佳结果所需的训练次数随节点增加而上升。

4. 为探究网络结构对训练结果的影响，我还分别构建了以下网络模型：

（1）双层隐含层，每层 25 节点，训练 600 次，正确率 93.53，结果和单层 25 节点（93.88%）相差不多。

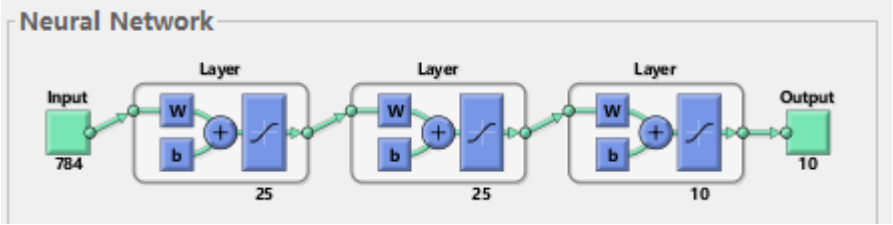


图 2.3.3 双层网络结构图

（2）双层隐含层，第一层 100 节点，第二层 50 节点，训练 600 次，正确率 96.41%。比单层 100 节点训练 600 次的 95.04%略高。

（3）三层隐含层，第一层 100 节点，第二层 50 节点，第三次 25 节点，训练 600 次，正确率 97.18%。

由这几个模型可知，两次相同节点数的网络对结果影响不大，但逐级递减的网络结构会对训练结果有一定改善。我认为原因在于，逐级递减的层数，一定程度避免了多节点（如 100）影响少节点（如 10）时，隐含层节点相互影响的抵消，相当于放大了每一层对下一层的影响。层数对结果的影响还需要进一步探究。

5. 对训练程度的近似估计

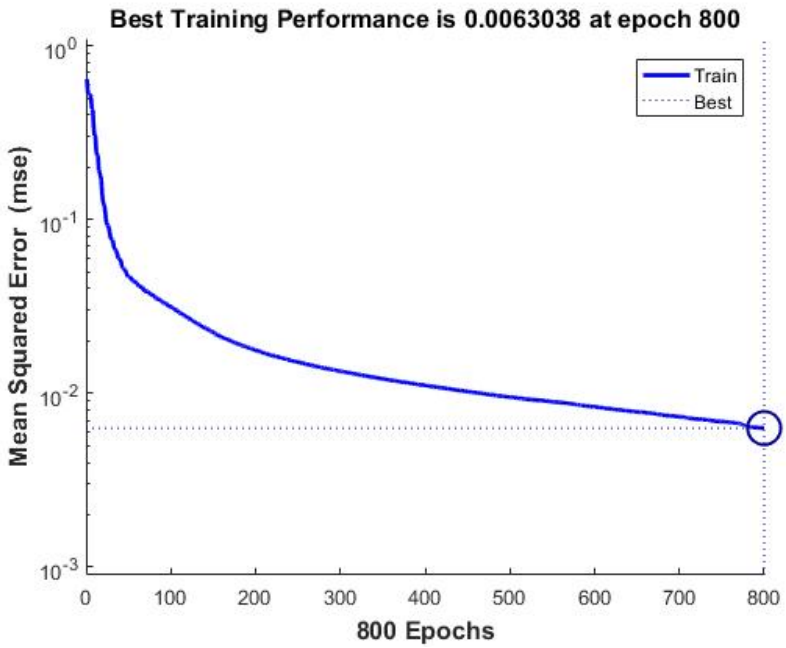


图 2.3.4 三层 BP 网络，200 隐含层节点，均方误差随迭代次数变化

从图 2.3.5 可以看出，选择 MSE 为损失函数后，随迭代次数增加，损失函数逐渐减小。一开始下降较快，最后下降较慢，说明 800 次已经训练较为完善，甚至可能有过拟合的情况。如果训练 300 次，在图中可以看到 300 次对应点的斜率仍较大，具有欠拟合的可能。因此可以通过这个图近似估计训练程度。

6. 实际检测

我将手写的数字“0513”在使用 PS 调整大小后，输入训练好的网络进行训练，训练结果如下：

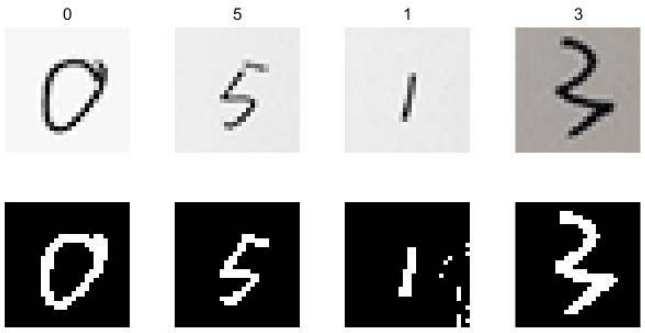


图 2.3.5 对手写数字进行检测

图中第一行是原始图像，第二行是经过图像处理之后得到的图像，也就是进行预测的图像，最上面的标签为网络对图像数字进行的判断。

在实际应用中，对图像的处理非常重要，且对最后判断的正确率影响很大。我进行图像处理的步骤记录如下：

（1） 将彩色图转为黑白图。
（2） 转为 double 型图像。
（3） 使用 imadjust 函数进行对比度调整
（4） 找到调整后图像的像素均值 a
（5） 将大于均值 a 的像素赋 0，小于 a 的像素赋 1，以达到进一步提高对比度并反转图像的目的。

从手写的四个数字来看，识别效果很好，且具有一定的鲁棒性。

五、实验代码

1.读取标签函数（读取图像函数在第三节末尾）

```
function labels = loadMNISTLabels(filename)
%读入label数据，返回每个图片的标签

fp = fopen(filename, 'rb');
assert(fp ~= -1, ['Could not open ', filename, '']);

magic = fread(fp, 1, 'int32', 0, 'ieee-be');
assert(magic == 2049, ['Bad magic number in ', filename, '']);

numLabels = fread(fp, 1, 'int32', 0, 'ieee-be');

labels = fread(fp, inf, 'unsigned char'); %把内容全读进来，并转为
unsigned char类型

assert(size(labels,1) == numLabels, 'Mismatch in label count');

fclose(fp);
end
```

2. BP 网络程序

```
clc; clear;
```

```
test_img = loadMNISTImages('MNIST数据集\t10k-images.idx3-ubyte');
test_lab = loadMNISTLabels('MNIST数据集\t10k-labels.idx1-ubyte');
train_img = loadMNISTImages('MNIST数据集\train-images.idx3-ubyte');
train_lab = loadMNISTLabels('MNIST数据集\train-labels.idx1-ubyte');

train_lab2=handlelabel(train_lab);
test_lab2=handlelabel(test_lab);

%三层神经网络
train_num = 60000;
input = 784; %输入层
output = 10; %输出层，10个输出代表十个数字，也转化为了0-1的变量
hid1 = 100; %隐藏层

PR=zeros(input,2); %限定输入值的范围0-1
PR(:,1)=0;PR(:,2)=1;
%net=newff(PR,[hid1,output],{'tansig','tansig'},'trainlm','learngdm',
'mse');
net=newff(PR,[hid1,output],{'tansig','tansig'},'trainscg','learngdm',
'mse');%trainscg可以用GPU训练
% err=1e-5;
% net.trainparam.goal=err;   %期望训练误差
net.trainParam.lr=0.1;   %学习速率
net.trainparam.epochs=700; %最多训练步数
net.trainParam.show=1;   %每隔这么多步显示一次训练结果

gpudev=gpuDevice; %事先声明gpudev变量为gpu设备类
gpudev.AvailableMemory; %实时获得当前gpu的可用内存

[net,tr]=train(net,train_img,train_lab2,'useGPU','yes'); %在2017版以后
转为single型运算速度更快

BP_labma=sim(net,test_img);
BP_lab=zeros(1,10000);
for i=1:10000   %测试
    [~ , BP_lab(i)]=max(BP_labma(:,i));
    BP_lab(i)=BP_lab(i)-1;
end
j=(sum(BP_lab==test_lab'))/10000;
```

3. 将结果矩阵转为数字的函数

```
function labelma = handlelabel(z)
[m , ~]=size(z);
labelma=zeros(10,m);
for i=1:m
```

```
    labelma(z(i)+1,i)= 1;
    end
```

4. 测试手写图片

```
clc;clear;
load('net3');
raw(:,:,:,1) = imread('手写0.bmp');
raw(:,:,:,2) = imread('手写5.bmp');
raw(:,:,:,3) = imread('手写1.bmp');
raw(:,:,:,4) = imread('手写3-2.bmp');
T1=zeros(28,28,4);
b=zeros(28,28);
for i=1:4
T(:,:,i)=rgb2gray(raw(:,:,:,i)); %将彩色图转为黑白图
Tr(:,:,i)=im2double(T(:,:,i));
z=min(min(Tr(:,:,i)));
y=max(max(Tr(:,:,i)));
if z>0 || y<1
Tr(:,:,i)=imadjust(Tr(:,:,i),[z,y],[0,1]);
end
a(i)=sum(sum(Tr(:,:,i)))/(28*28);
b(Tr(:,:,i)>a(i))=0;
b(Tr(:,:,i)<a(i))=1;
Tr(:,:,i)=b;
end

T1=Tr;
[height,width,flag]=size(T1);
T1=reshape(T1,[height*width,4]);
labma=sim(net,T1);
lab=zeros(1,4);
for i=1:4   %测试
    [~ , lab(i)]=max(labma(:,i));
    lab(i)=lab(i)-1;
end
subplot(2,4,1);imshow(raw(:,:,:,1));title(num2str(lab(1)));
subplot(2,4,5);imshow(Tr(:,:,1));

subplot(2,4,2);imshow(raw(:,:,:,2));title(num2str(lab(2)));
subplot(2,4,6);imshow(Tr(:,:,2));

subplot(2,4,3);imshow(raw(:,:,:,3));title(num2str(lab(3)));
subplot(2,4,7);imshow(Tr(:,:,3));

subplot(2,4,4);imshow(raw(:,:,:,4));title(num2str(lab(4)));
```

```
subplot(2,4,8);imshow(Tr(:,:,4));
```

六、收获、体会及建议

在本次实验中，我初步学习了 MATLAB 进行神经网络的搭建过程，理解了神经网络的原理，掌握了 MATLAB 神经网络工具包与 GPU 加速的操作步骤。我深刻体会到计算速度的重要性，明白了算法改进对速度的重要影响。一旦网络训练完毕，使用起来还是非常快的，在一些特定问题上准确率也可以达到非常高，因此神经网络正逐渐成为我们生活不可分割的一部分。

参考文献

1. 张学工编著. *模式识别（第三版）*. 清华大学出版社，2010.
2. ppt 课件：2019 系列实验-张先锐
3. 李阳老师的 ppt 课件
4. Rosenblatt, F., *Principles of Neurodynamics*, Washington, D.C., Spartan Press, 1961
5. CSDN：MATLAB 神经网络编程（一）—— 感知器 https://blog.csdn.net/FIELDOFFIER/article/details/44264715
6. CSDN：SVM matlab 代码说明 https://www.cnblogs.com/welen/p/6422068.html
7. CSDN：MATLAB 读取 mnist 数据库 https://blog.csdn.net/tracer9/article/details/51253604
8. CSDN：建立 BP 神经网络中 train 函数的参数及学习算法参数 https://blog.csdn.net/ckzhb/article/details/60879570matlab
9. matlab 运行神经网络可以用 GPU 吗 http://f.dataguru.cn/thread-874432-1-1.html
10. 深入理解 BP 神经网络 https://www.jianshu.com/p/6ab6f53874f7
11. 【MATLAB】BP 神经网络识别 MNIST 手写数字 https://blog.csdn.net/github_35807147/article/details/80529247