1. Question 1

In general we cannot use $J(w)=w^T S_B w$ as a cost function in Fisher Linear Discriminant analysis because original cost function in Fisher Linear Discriminant Analysis $J(w)=w^T S_B w / w^T S_W w$ derived from maximization of absolute distance value of mean of projected feature vector of each class divided by sum of scatter of each class. Use of $J(w)=w^T S_B w$ can be only possible the case where S_w is scalar multiple of identity matrix. The reporter investigated numerically using random variable varying covariance of distribution each class. Author generated multivariate normal random variable and multivariate t random variable using MATLAB 'mvnrnd' and 'mvtrnd' function. Specific of random variable is as follow. Dimension: 2 and 3; number of samples: 4000 per each class; distance between mean of each class is 1 regardless of dimension of random variable; degree of freedom of t statistic is 3. Large number of sample number was used so that approximate projected t random variable can be approximated as normal random variable taking advantage of central limit theorem; covariance or correlation of each class is shown in Appendix. That is, in classifying projected samples via Bayesian Decision Rule, author fitted projected t random variable into normal random variable estimating variance and mean of projected random variable of each class.

Numerical results of each classification method (using $J(w)=w^T S_B w / w^T S_W w$ versus using $J(w)=w^T S_B w$ as a criterion function) represented in the form of error rate percentage in the following.

Dimension 2 Case	$J(w)=w^T S_B w / w^T S_W w$	$J(w)=w^T S_B w$	Dimension 3 Case	$J(w)=w^T S_B w / w^T S_W w$	$J(w)=w^T S_B w$
1	3.3000	3.2500	1	3.1125	3.1125
2	2.8125	2.8250	2	2.8625	2.8375
3	3.6375	4.0875	3	2.2500	3.3000
4	5.4750	5.8375	4	3.3625	5.6000

Multivariate normal distribution

Dimension 2 Case	$J(w)=w^T S_B w / w^T S_W w$	$J(w)=w^T S_B w$	Dimension 3 Case	$J(w)=w^T S_B w / w^T S_W w$	$J(w)=w^T S_B w$
1	33.4375	33.9625	1	32.8375	32.7750
2	33.1625	32.8000	2	35.0000	34.9000
3	39.1750	39.1625	3	31.5875	33.6375
4	38.3375	37.9250	4	30.8875	32.6500

Multivariate t distribution

Result of case 3 and 4 of both dimension for normal random variable, and case 4 of dimension 2 and case 3 and 4 of dimension 3 for t distribution shows that when sum of covariance of each class is much different from scalar multiple of identity matrix, much there is difference of error rate between using $J(w)=w^T S_B w / w^T S_W w$ and using $J(w)=w^T S_B w$ as a criterion function. Experiment also show that as the dimension increase the difference of error rate becomes larger since it is more likely that off diagonal component of sum of covariance of each class exists.

2. Question 2

Data Generation

Synthetic data was generated using MATLAB 'mvnrnd' routine. Specific of random variable (total 18 combinations) is in the following. Dimension: 1, 2; number of sample per each class and training/test sample: 10, 100, and 1000; distance between mean of each class: 2, 1.26, and 1; variance/covariance: 0.1 and [0.1 0; 0 0.1]; As opposed to question 1, mean was varied so that distribution makes linearly separable and linearly nonseparable to be useful in assessing support vector machine.

Designing classifier

(a) Neural Network approach

Author used built in MATLAB neural network toolbox, specifically, 'newff' as a feedforward process, 'train' as a backprogation process, and 'sim' as a assess performance over test samples, respectively. In the neural network approach, there is one parameter to decide the number of hidden layer unit. In this experiment, author varied the number of hidden layer unit—2*dimension, 3*dimension, and 5*dimension of feature vector. Number of output layer unit is fixed number 2 because there are two classes to classify. Measure of performance is percentage of error rate over test samples as before. Below is error rate.

Error rate (%)	Dimension (D)=1				Dimension (D)=2		
Distance between means=2	N	H=2*D	3*D	5*D	H=2*D	3*D	5*D
	10	100.0000	0	65.0000	45.0000	0	0
	100	3.5000	0	49.5000	50.5000	0	0
	1000	50.4500	0	49.4000	0.6500	0	0
1.26	10	50.0000	0	65.0000	55.0000	0	5.0000
	100	1.5000	2.5000	0	50.5000	1.0000	0.5000
	1000	50.4500	0.4500	1.3500	16.7500	0.5500	0.4000
1	10	50.0000	5.0000	35.0000	55.0000	0	5.0000
	100	49.5000	8.5000	5.0000	50.5000	6.5000	5.5000
	1000	50.4500	5.8500	49.4000	48.7500	5.9000	5.7000

In above table, N represents the training sample number (and test sample number) of each class and H is the number of hidden layer unit. Above table tells us that error rate depends on the number of hidden layer units compared to dimensionality of feature vector. This parameter is free parameter and can be one caveat of neural network approach. From this experiment authors claims that the number of hidden layer would be 3 folds of dimension of feature space if stable performance is required. Especially, small number of hidden layer should be avoided in order to produce stable classifier. We can also observes that when dimension of feature vector goes to 2, performance is less sensitive to hidden layer unit number beyond certain point, say, three times of feature vector dimension. Comparison error rate when the number of training sample is 100 and 1000 given three fold hidden

layer unit number indicates that the more training sample the better performance of classifier. In case of 10 training samples, high performance was expected because test sample itself is quite small and is excluded in generalization of performance related to the number of training samples.

(b) Support Vector Machine

In implementation of support vector machine, the reporter used code named ‘simpleSVM’ on the web [1]. Running result of ‘simpleSVM’ over the same data used in neural network approach is in the following.

	Dimension (D)=1			Dimension (D)=2		
Distance between means=2	N	Error rate (%)	Linearly Separable ?	N	Error rate (%)	Linearly Separable ?
	10	0	Yes	10	0	Yes
	100	4.0000	No	100	4.0000	No
	1000	0.4000	No	1000	0.4000	No
1.26	10	0	Yes	10	0	Yes
	100	4.0000	No	100	4.0000	No
	1000	0.4000	No	1000	0.4000	No
1	10	5.0000	No	10	5.0000	No
	100	4.0000	No	100	4.0000	No
	1000	0.4000	No	1000	0.4000	No

Above table clearly shows the limitation of support vector machine. Support vector machine classified perfectly test samples when feature can be linearly separable. However, support cannot be applied to nonseparable data in linear fashion. One attraction of support machine from the experiment is the speed of algorithm. Support vector machine is fast. Support vector machine is also insensitive to number of training samples (test samples) in performance aspect.

(c) Comparison between neural network approach and support vector machine

One strong point of neural network is decision surface can be nonlinear in feature vector space. Therefore when feature vector are mingled in feature vector space, neural network can be recommended to use for classification. However, neural network approach depends on the number of hidden layer units and it requires sufficiently large number of training data. On the other hand, support vector machine is virtually parameter free algorithm and can be perfect in test data can be linearly separable. This support vector machine is advertising fast processing speed compared to neural network approach. Therefore, that which algorithm use depends on application—characteristic of feature vector.

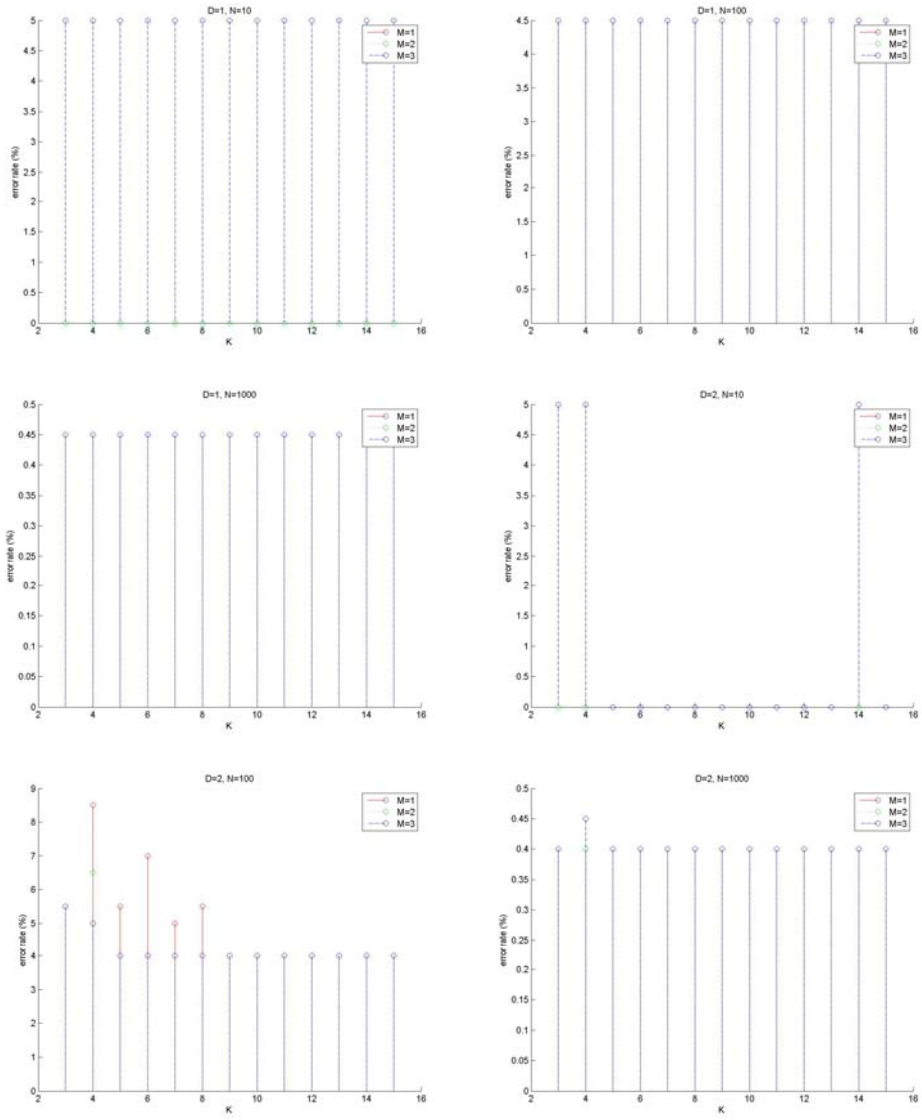
3. Question 3

(a) Parzen window technique

Free parameter of parzen window is the type of window and the window width, i.e., volume. The reporter tried to experiment the error rate varying those parameter. As the type of window, Gaussian and hypercube window was used in MATLAB

(b) K-nearest neighbor technique

In the implementation of K-nearest neighbor technique, author used routine called ‘knn’ on the web [2]. In the K-nearest neighbor technique, the author changed K from 2 to 15 and used L2 norm as a distance metric. He also speculated the error rate of test sample the result varying dimension and the number training sample/ test sample described in below.



In above figure, D represents dimension of feature vector, M the index of distance between the means as before, N number of training sample (test sample) for each class, and K the number of nearest neighbor used in the algorithm

shown from 2 to 15. From this graph, we can see the sample dimension does not heavily affect error rate and K slightly influence error rate. The figure also says that as the larger number training sample, error rate decrease which mean probability estimate is more accurate.

(c) Nearest neighbor technique

In special case of K-nearest neighbor technique, author tries to incorporate Manhattan distance metric modifying existing 'knn' routine and compared. Comparison table of error rate is shown in the following.

Error Rate (%)	Dimension (D)=1			Dimension (D)=2		
Distance between means=2	N	L2 norm	Manhattan	N	L2 norm	Manhattan
	10	0	0	10	0	0
	100	4.5000	8.0000	100	10.5000	10.5000
	1000	0.4500	1.1000	1000	0.9500	1.0500
1.26	10	0	0	10	0	0
	100	4.5000	9.5000	100	9.5000	10.5000
	1000	0.4500	1.2500	1000	1.1500	1.0500
1	10	5	5.0000	10	10.0000	10
	100	4.5000	10.5000	100	8.5000	9
	1000	0.4500	1.0500	1000	0.9500	1

As in K-nearest neighbor, nearest neighbor shows the tendency that the greater number of training samples, the more accurate estimate of density of feature vector. In this pattern of feature vector, the nearest neighbor with L2 norm distance metric outperforms over with Manhattan distance metric. If feature vectors are continuously distributed on grid, Manhattan distance metric may be more accurate distance metric.

(d) Comparison over three nonparametric technique

From error rate comparison, we can find that the nearest neighbor technique can substitute K-nearest neighbor technique in the case where feature is compactly distributed and separated in far distance. However, this is not always the case. If feature vector is scattered the nearest neighbor can misclassify the pattern. That is the nearest neighbor technique is more sensitive to outlier than K-nearest neighbor technique.

APPENDIX

Covariance/ correlation matrix of random variable used for problem 1

(a) Normal distribution

i) Dimension 2

Sigma1{1}=[0.1 0; 0 0.1]; Sigma2{1}=[0.05 0; 0 0.05];

Sigma1{2}=[0.1 0; 0 0.05]; Sigma2{2}=[0.05 0; 0 0.1];

Sigma1{3}=[0.1 0; 0 0.05]; Sigma2{3}=[0.1 0; 0 0.08];

Sigma1{4}=[0.1 0.05; 0.05 0.1]; Sigma2{4}=[0.1 0.03; 0.03 0.5];

ii) Dimension 3

Sigma1{1}=[0.1 0 0; 0 0.1 0; 0 0 0.1]; Sigma2{1}=[0.05 0 0; 0 0.05 0;0 0 0.05];

Sigma1{2}=[0.1 0 0; 0 0.1 0; 0 0 0.05]; Sigma2{2}=[0.05 0 0; 0 0.05 0;0 0 0.1];

Sigma1{3}=[0.1 0.03 0; 0.03 0.1 0.03; 0 0.03 0.05]; Sigma2{3}=[0.05 0.03 0; 0.03 0.1 0.03; 0 0.03 0.05];

Sigma1{4}=[0.1 0.05 0.03; 0.05 0.1 0.05; 0.03 0.05 0.1]; Sigma2{4}=[0.1 0.05 0.02; 0.05 0.1 0.05; 0.02 0.05 0.08];

b) t statistic (correlation matrix)

i) Dimension 2

Sigma1{1}=[0.1 0; 0 0.1]; Sigma2{1}=[0.05 0; 0 0.05];

Sigma1{2}=[0.1 0; 0 0.05]; Sigma2{2}=[0.05 0; 0 0.1];

Sigma1{3}=[0.1 0; 0 0.05]; Sigma2{3}=[0.1 0; 0 0.08];

Sigma1{4}=[0.1 0.05; 0.05 0.1]; Sigma2{4}=[0.1 0.03; 0.03 0.5];

ii) Dimension 3

Sigma1{1}=[0.1 0 0; 0 0.1 0; 0 0 0.1]; Sigma2{1}=[0.05 0 0; 0 0.05 0;0 0 0.05];

Sigma1{2}=[0.1 0 0; 0 0.1 0; 0 0 0.05]; Sigma2{2}=[0.05 0 0; 0 0.05 0;0 0 0.1];

Sigma1{3}=[0.1 0.03 0; 0.03 0.1 0.03; 0 0.03 0.05]; Sigma2{3}=[0.05 0.03 0; 0.03 0.1 0.03; 0 0.03 0.05];

Sigma1{4}=[0.1 0.05 0.03; 0.05 0.1 0.05; 0.03 0.05 0.1]; Sigma2{4}=[0.1 0.05 0.02; 0.05 0.1 0.05; 0.02 0.05 0.08];

REFERENCE

[1] simpleSVM, http://asi.insa-rouen.fr/enseignants/~gloosli/simpleSVM.html

[2] KNN, http://homepages.cae.wisc.edu/%7Eece539/matlab/

```
%ECE662 Homework 2 Problem 1

clear all
close all

D=[2 3]; %dimension of feature vector
N=8000; % number of total training sample
L=4;

Mu1=cell(1,length(D));
Mu2=cell(1,length(D));
Mu1{1,1}=[-0.5 0];
Mu2{1,1}=[0.5 0];
Mu1{1,2}=[-0.5 0 0];
Mu2{1,2}=[0.5 0 0];

Sigma1=cell(L,length(D));
Sigma2=cell(L,length(D));

Sigma1{1,1}=[0.1 0; 0 0.1]; Sigma2{1,1}=[0.05 0; 0 0.05];
Sigma1{2,1}=[0.1 0; 0 0.05]; Sigma2{2,1}=[0.05 0; 0 0.1];
Sigma1{3,1}=[0.1 0; 0 0.05]; Sigma2{3,1}=[0.1 0; 0 0.08];
Sigma1{4,1}=[0.1 0.05; 0.05 0.1]; Sigma2{4,1}=[0.1 0.03; 0.03 0.5];

Sigma1{1,2}=[0.1 0 0; 0 0.1 0; 0 0 0.1];
Sigma2{1,2}=[0.05 0 0; 0 0.05 0;0 0 0.05];
Sigma1{2,2}=[0.1 0 0; 0 0.1 0; 0 0 0.05];
Sigma2{2,2}=[0.05 0 0; 0 0.05 0;0 0 0.1];

Sigma1{3,2}=[0.1 0.03 0; 0.03 0.1 0.03; 0 0.03 0.05];
Sigma2{3,2}=[0.05 0.03 0; 0.03 0.1 0.03; 0 0.03 0.05];

Sigma1{4,2}=[0.1 0.05 0.03; 0.05 0.1 0.05; 0.03 0.05 0.1];
Sigma2{4,2}=[0.1 0.05 0.02; 0.05 0.1 0.05; 0.02 0.05 0.08];

perc_err=zeros(L,length(D));
perc_erri=zeros(L,length(D));
for d=1:length(D),
   for ll=1:L,
%generate feature vecture

%First think about Gaussian 'mvnrnd' r = mvnrnd(MU,SIGMA,cases)
%t-statistic 'mvtrnd r=mvtrnd(sigma, df, cases)'
   if d==1,
theta=pi/4;
R=[cos(theta) -sin(theta); sin(theta) cos(theta)];
Mu1{1,d}=(R*Mu1{1,d}.').';
Mu2{1,d}=(R*Mu2{1,d}.').';
   end

%Sigma=[.1 0; 0 .1]; %vary sigma1=[], sigma 2, sigma1, sigma2
```

```
%Sigma1=[0.1 0; 0 0.05];
%Sigma2=[0.05 0; 0 0.1];

X1=mvnrnd(Mu1{1,d}, Sigma1{ll,d}, N/2); %Can change into other distri.
X2=mvnrnd(Mu2{1,d}, Sigma2{ll,d}, N/2); %Can change into other distri.

X=cat(1,X1,X2);

%data for Fisher Linear discriminant analysis

m1=mean(X1).';
m2=mean(X2).';
S1=N/2*cov(X1,1);
S2=N/2*cov(X2,1);
Sw=S1+S2;

% Do original fisher linear discriminant analysis
w=(Sw\eye(D(d)))*(m1-m2);
Y=zeros(N,1);
for n=1:N,
Y(n)=((w./norm(w)).' * (X(n,:).')).';
end

%traditional Bayesian decision rule
std1=std(Y(1:(N/2)));
std2=std(Y((N/2 +1):N));
mean1=mean(Y(1:(N/2)));
mean2=mean(Y((N/2+1):N));

G=log(std2/std1)-((Y-mean1).^2)/(2*std1.^2)+((Y-mean2).^2)/(2*std2.^2);

% Do question
wi=m1-m2;
Yi=zeros(N,1);
for n=1:N,
Yi(n)=((wi./norm(wi)).' * (X(n,:).')).';
end

%traditional Bayesian decision rule
std1=std(Yi(1:(N/2)));
std2=std(Yi((N/2 +1):N));
mean1=mean(Yi(1:(N/2)));
mean2=mean(Yi((N/2+1):N));

Gi=log(std2/std1)-((Yi-mean1).^2)/(2*std1.^2)+((Yi-
mean2).^2)/(2*std2.^2);


% calculate error rate of both cases and compare (percent)

perc_err(ll,d)=(sum([G(1:(N/2))<0 ; G((N/2+1):N)>0])/N)*100; %class1
```

```
should >0 class2 should <0
perc_erri(ll,d)=(sum([Gi(1:(N/2))<0 ; Gi((N/2+1):N)>0])/N)*100;

    end
end
%filename=sprintf('Problem1a.mat');
%save(filename,'perc_err','perc_erri');
```

```
%ECE662 Homework 2 Problem 1

clear all
close all

D=[2 3]; %dimension of feature vector
N=8000; % number of total training sample
L=4;

Mu1=cell(1,length(D));
Mu2=cell(1,length(D));
Mu1{1,1}=[-0.5 0];
Mu2{1,1}=[0.5 0];
Mu1{1,2}=[-0.5 0 0];
Mu2{1,2}=[0.5 0 0];

Sigma1=cell(L,length(D));
Sigma2=cell(L,length(D));

Sigma1{1,1}=[0.1 0; 0 0.1]; Sigma2{1,1}=[0.05 0; 0 0.05];
Sigma1{2,1}=[0.1 0; 0 0.05]; Sigma2{2,1}=[0.05 0; 0 0.1];
Sigma1{3,1}=[0.1 0; 0 0.05]; Sigma2{3,1}=[0.1 0; 0 0.08];
Sigma1{4,1}=[0.1 0.05; 0.05 0.1]; Sigma2{4,1}=[0.1 0.03; 0.03 0.5];

Sigma1{1,2}=[0.1 0 0; 0 0.1 0; 0 0 0.1];
Sigma2{1,2}=[0.05 0 0; 0 0.05 0;0 0 0.05];
Sigma1{2,2}=[0.1 0 0; 0 0.1 0; 0 0 0.05];
Sigma2{2,2}=[0.05 0 0; 0 0.05 0;0 0 0.1];

Sigma1{3,2}=[0.1 0.03 0; 0.03 0.1 0.03; 0 0.03 0.05];
Sigma2{3,2}=[0.05 0.03 0; 0.03 0.1 0.03; 0 0.03 0.05];

Sigma1{4,2}=[0.1 0.05 0.03; 0.05 0.1 0.05; 0.03 0.05 0.1];
Sigma2{4,2}=[0.1 0.05 0.02; 0.05 0.1 0.05; 0.02 0.05 0.08];

perc_err=zeros(L,length(D));
perc_erri=zeros(L,length(D));
for d=1:length(D),
   for ll=1:L,
%generate feature vecture

%First think about Gaussian 'mvnrnd' r = mvnrnd(MU,SIGMA,cases)
%t-statistic 'mvtrnd r=mvtrnd(sigma, df, cases)'

%Sigma=[.1 0; 0 .1]; %vary sigma1=[], sigma 2, sigma1, sigma2
%Sigma1=[0.1 0; 0 0.05];
%Sigma2=[0.05 0; 0 0.1];

X1=mvtrnd(Sigma1{ll,d},3, N/2);
temp =Mu1{1,d}; %mvtrnd
for dd=1:D(d),
```

```
X1(:,dd)=X1(:,dd)+temp(dd);
end

X2=mvtrnd(Sigma2{ll,d},3,N/2);
temp=Mu2{1,d}; %Can change into other distri.
for dd=1:D(d),
   X2(:,dd)=X2(:,dd)+temp(dd);
end




X=cat(1,X1,X2);

%data for Fisher Linear discriminant analysis

m1=mean(X1).';
m2=mean(X2).';
S1=N/2*cov(X1,1);
S2=N/2*cov(X2,1);
Sw=S1+S2;

% Do original fisher linear discriminant analysis
w=(Sw\eye(D(d)))*(m1-m2);
Y=zeros(N,1);
for n=1:N,
Y(n)=((w./norm(w)).' * (X(n,:).')).';
end

%traditional Bayesian decision rule
std1=std(Y(1:(N/2)));
std2=std(Y((N/2 +1):N));
mean1=mean(Y(1:(N/2)));
mean2=mean(Y((N/2+1):N));

G=log(std2/std1)-((Y-mean1).^2)/(2*std1.^2)+((Y-mean2).^2)/(2*std2.^2);

% Do question
wi=m1-m2;
Yi=zeros(N,1);
for n=1:N,
Yi(n)=((wi./norm(wi)).' * (X(n,:).')).';
end

%traditional Bayesian decision rule
std1=std(Yi(1:(N/2)));
std2=std(Yi((N/2 +1):N));
mean1=mean(Yi(1:(N/2)));
mean2=mean(Yi((N/2+1):N));

Gi=log(std2/std1)-((Yi-mean1).^2)/(2*std1.^2)+((Yi-
```

```
mean2).^2)/(2*std2.^2);


% calculate error rate of both cases and compare (percent)

perc_err(ll,d)=(sum([G(1:(N/2))<0 ; G((N/2+1):N)>0])/N)*100; %class1
should >0 class2 should <0
perc_erri(ll,d)=(sum([Gi(1:(N/2))<0 ; Gi((N/2+1):N)>0])/N)*100;

    end
end
filename=sprintf('Problem1b.mat');
save(filename,'perc_err','perc_erri');
```

```
%ECE662 Homework Problem 2 (a)

clear all
close all
%create data
D=[1 2]; %Dimension of feature space
SamplNum=[40 400 4000]; %Total number of sample

mu1=cell(3,2);
mu2=cell(3,2);

mu1{1,1}=-2; mu2{1,1}=2;
mu1{2,1}=-0.8; mu2{2,1}=0.8;
mu1{3,1}=-0.5; mu2{3,1}=0.5;

mu1{1,2}=[-2 0]; mu2{1,2}=[2 0];
mu1{2,2}=[-0.8 0]; mu2{2,2}=[0.8 0];
mu1{3,2}=[-0.5 0]; mu2{3,2}=[0.5 0];

Sigma=cell(1,2);
Sigma{1,1}=0.1;
Sigma{1,2}=[0.1 0; 0 0.1];

hidden_mul=[2 3 5];

train_index=cell(length(D),length(SamplNum),3,length(hidden_mul));
vald_index=cell(length(D),length(SamplNum),3,length(hidden_mul));
test_index=cell(length(D),length(SamplNum),3,length(hidden_mul));
perc_err=zeros(length(D),length(SamplNum),3,length(hidden_mul));

for d=1:length(D),
    for n=1:length(SamplNum),
        for m=1:3,
            for hh=1:length(hidden_mul),
                disp(['d=' num2str(d) ' n=' num2str(n) ' m=' num2str(m) '
hh=' num2str(hh)]);
            filename1=sprintf('./MAT/Gaussian_D%sN%sM%s.mat',...
                num2str(D(d)),num2str(SamplNum(n)),num2str(m));
            load(filename1, 'X'); %import data
%------------------------------------------------------------
temp1=cat(1,ones(SamplNum(n)/2,1),zeros(SamplNum(n)/2,1));
temp2=cat(1,zeros(SamplNum(n)/2,1),ones(SamplNum(n)/2,1));
class=cat(2,temp1, temp2);
clear temp1 temp2

%-------------------------------------------------------------------
X=X.'; %transpose for neural network
class=class.';
%-----------------
[X,ps] = mapminmax(X);         % Normalize inputs
%--------------------------------
```

```
rand('seed', 491218382)
num_hidden=hh*D;
nout = size(class,1);                    % Number of outputs = 2
net = newff(minmax(X),[num_hidden nout]); % Create a new feed forward
network
%--------------------------
[trS, cvS, tstS] = dividevec(X, class, 0.1, 0.5);
%--------------training
net = train(net, trS.P, trS.T, [], [], cvS, tstS);
%-------------------test
out = sim(net, tstS.P);          % Get response from trained network
%------------------calculate error rate
[y_out,I_out] = max(out);
[y_t,I_t] = max(tstS.T);

diff = [I_t - 2*I_out]; %#ok<NBRAK>

f_f = length(find(diff==-2));    % Female crabs classified as Female
f_m = length(find(diff==-3));    % Female crabs classified as Male
m_m = length(find(diff==-1));    % Male crabs classified as Male
m_f = length(find(diff==0));     % Male crabs classified as Female

N = size(tstS.P,2);             % Number of testing samples
fprintf('Total testing samples: %d\n', N);

cm = [f_f f_m; m_f m_m];         % classification matrix

cm_p = (cm ./ N) .* 100;        % classification matrix in percentages

fprintf('Percentage Correct classification   : %f%%\n',
100*(cm(1,1)+cm(2,2))/N);
fprintf('Percentage Incorrect classification : %f%%\n',
100*(cm(1,2)+cm(2,1))/N);

train_index{d,n,m,hh}=trS.indices;
vald_index{d,n,m,hh}=cvS.indices;
test_index{d,n,m,hh}=tstS.indices;
perc_err(d,n,m,hh)=100*(cm(1,2)+cm(2,1))/N;



          end
       end
    end
end
%filename2=sprintf('Problem2a.mat');
%save(filename2,'train_index','vald_index','test_index','perc_err');
```

```
%ECE662 Homework2 Problem 2b

addpath('./SVM/simpleSVM/');

global svModel %#ok<NUSED>

filename=sprintf('Problem2a.mat');
load(filename,'train_index','vald_index','test_index');

D=[1 2]; %Dimension of feature space
SamplNum=[40 400 4000]; %Total number of sample

perc_err=zeros(length(D),length(SamplNum),3);


hh=3;
for d=1:length(D),
    for n=1:length(SamplNum),
        for m=1:3,

            disp(['d=' num2str(d) ' n=' num2str(n) ' m=' num2str(m)]);
          filename1=sprintf('./MAT/Gaussian_D%sN%sM%s.mat',...
            num2str(D(d)),num2str(SamplNum(n)),num2str(m));
          load(filename1, 'X'); %import data


tr_idx=train_index{d,n,m,hh};
vl_idx=vald_index{d,n,m,hh};
ts_idx=test_index{d,n,m,hh};

train_idx=cat(2,tr_idx,vl_idx);

x=zeros(D(d),length(train_idx));
xt=zeros(length(train_idx),1);
y=zeros(D(d),length(ts_idx));
yt=zeros(length(ts_idx),1);

for nn=1:length(train_idx),
    x(:,nn)=X(train_idx(nn),:);
    if train_idx(nn)<=(SamplNum/2),
    xt(nn)=1;
    else xt(nn)=2;
    end
end

for nn=1:length(ts_idx),
    y(:,nn)=X(ts_idx(nn),:);
    if ts_idx(nn)<=(SamplNum/2),
    yt(nn)=1;
    else yt(nn)=2;
```

```
    end
end


donnees = data(x,xt,y,yt);                % stores data
noyau = kernel('rbf',.9);                 % stores kernel
%parametres = param(100,10,'binary','chol'); % stores parameters
parametres = param(100,20,'1vsall','chol'); % stores parameters
trainSVM(donnees, noyau, parametres);      % train the SVM
prediction = testSVM;                  % gives the results on the test set

%calculate error rate
perc_err(d,n,m)=sum(prediction~=yt)/length(yt)*100;

       end
    end
end
%filename=sprintf('Problem2b.mat');
%save(filename,'perc_err');
```

```
%ECE662 Homework 2 Problem 3 b


clear all
close all

%import data
filename=sprintf('Problem2a.mat');
load(filename,'train_index','vald_index','test_index');

D=[1 2]; %Dimension of feature space
SamplNum=[40 400 4000]; %Total number of sample

perc_err=cell(length(D),length(SamplNum),3);


hh=2;
K=15;

for d=2,
    for n=3,
       for m=1,
%for d=1:length(D),
%    for n=1:length(SamplNum),
%       for m=1:3,

     disp(['d=' num2str(d) ' n=' num2str(n) ' m=' num2str(m)]);
          filename1=sprintf('./MAT/Gaussian_D%sN%sM%s.mat',...
              num2str(D(d)),num2str(SamplNum(n)),num2str(m));
          load(filename1, 'X'); %import data


tr_idx=train_index{d,n,m,hh};
vl_idx=vald_index{d,n,m,hh};
ts_idx=test_index{d,n,m,hh};


train_idx=cat(2,tr_idx,vl_idx);
clear tr_idx vl_idx

%make Pr Tr Pt Tt
Pr=zeros(length(train_idx),D(d)); %Pr:pattern of training
Tr=zeros(length(train_idx),2); %Tr: target of training
Pt=zeros(length(ts_idx),D(d)); %Pt: pattern of test
Tt=zeros(length(ts_idx),2); %Tt: target of test

%Class 1: [1 0], Class 2: [0 1]
for nn=1:length(train_idx),
    Pr(nn,:)=X(train_idx(nn),:);
    if train_idx(nn)<=(SamplNum/2),
```

```
    Tr(nn,1)=1;
    Tr(nn,2)=0;
    else
        Tr(nn,1)=0;
        Tr(nn,2)=1;
    end
end

for nn=1:length(ts_idx),
    Pt(nn,:)=X(ts_idx(nn),:);
    if ts_idx(nn)<=(SamplNum/2),
    Tt(nn,1)=1;
    Tt(nn,2)=0;
    else
        Tt(nn,1)=0;
        Tt(nn,2)=1;
    end
end

C_rate=zeros(1,K-1);

for k=1:K,
   [Cmat,C_rate(k)]=knn(Pr,Tr,Pt,Tt,k);
   %disp([int2str(k) '-nearest neighbor classifier, confusion matrix = '])
   %Cmat;
end

%calculate error rate
perc_err{d,n,m}=100-C_rate;
%{
figure(1),clf
stem([1:15],100-C_rate),title('Classification error rate vs. k')
xlabel('k -nearest neighbors')
ylabel('% classification error')
%}
       end
    end
end
%filename=sprintf('Problem3b.mat');
%save(filename,'perc_err');

%{
for d=1:2,
    for n=1:3,
figure
hold on
temp=squeeze(perc_err{d,n,1});
stem(3:15,temp(3:15),'r-')
temp=squeeze(perc_err{d,n,2});
stem(3:15, temp(3:15),'g:')
temp=squeeze(perc_err{d,n,3});
stem(3:15, temp(3:15),'b--')
```

```
legend('M=1','M=2','M=3')
title_text=sprintf('D=%s, N=%s',num2str(d), num2str(10^n));
title(title_text)
xlabel('K');
ylabel('error rate (%)');

filename1=sprintf('Problem3b_D%sN%s.jpg',num2str(d),num2str(n));
saveas(gcf,filename1,'jpg');
   end
end
%}
```

```
%ECE662 Homework 2 Problem 3 c


clear all
close all

%import data
filename=sprintf('Problem2a.mat');
load(filename,'train_index','vald_index','test_index');

D=[1 2]; %Dimension of feature space
SamplNum=[40 400 4000]; %Total number of sample

perc_err=cell(length(D),length(SamplNum),3);


hh=2;
K=15;

%for d=1,
%   for n=1,
%      for m=1,
for d=1:length(D),
   for n=1:length(SamplNum),
      for m=1:3,

     disp(['d=' num2str(d) ' n=' num2str(n) ' m=' num2str(m)]);
          filename1=sprintf('./MAT/Gaussian_D%sN%sM%s.mat',...
             num2str(D(d)),num2str(SamplNum(n)),num2str(m));
          load(filename1, 'X'); %import data


tr_idx=train_index{d,n,m,hh};
vl_idx=vald_index{d,n,m,hh};
ts_idx=test_index{d,n,m,hh};


train_idx=cat(2,tr_idx,vl_idx);
clear tr_idx vl_idx

%make Pr Tr Pt Tt
Pr=zeros(length(train_idx),D(d)); %Pr:pattern of training
Tr=zeros(length(train_idx),2); %Tr: target of training
Pt=zeros(length(ts_idx),D(d)); %Pt: pattern of test
Tt=zeros(length(ts_idx),2); %Tt: target of test

%Class 1: [1 0], Class 2: [0 1]
for nn=1:length(train_idx),
   Pr(nn,:)=X(train_idx(nn),:);
   if train_idx(nn)<=(SamplNum/2),
```

```
    Tr(nn,1)=1;
    Tr(nn,2)=0;
    else
        Tr(nn,1)=0;
        Tr(nn,2)=1;
    end
end

for nn=1:length(ts_idx),
    Pt(nn,:)=X(ts_idx(nn),:);
    if ts_idx(nn)<=(SamplNum/2),
    Tt(nn,1)=1;
    Tt(nn,2)=0;
    else
        Tt(nn,1)=0;
        Tt(nn,2)=1;
    end
end



for k=1,
   [Cmat,C_rate]=knn_manhat(Pr,Tr,Pt,Tt,k);
   %disp([int2str(k) '-nearest neighbor classifier, confusion matrix = '])
   %Cmat;
end

%calculate error rate
perc_err{d,n,m}=100-C_rate;
%{
figure(1),clf
stem([1:15],100-C_rate),title('Classification error rate vs. k')
xlabel('k -nearest neighbors')
ylabel('% classification error')
%}
        end
    end
end

perc_err_Man=zeros(d,n,m);
for d=1:length(D),
    for n=1:length(SamplNum),
        for m=1:3,
            temp=squeeze(perc_err{d,n,m});
            perc_err_Man(d,n,m)=temp;
        end
    end
end


load('Problem3b.mat');
perc_err_L2=zeros(length(D),length(SamplNum),3);
```

```
for d=1:length(D),
    for n=1:length(SamplNum),
        for m=1:3,
            temp=squeeze(perc_err{d,n,m});
            perc_err_L2(d,n,m)=temp(1);
        end
    end
end
%filename=sprintf('Problem3c.mat');
%save(filename,'perc_err_L2','perc_err_Man');
```