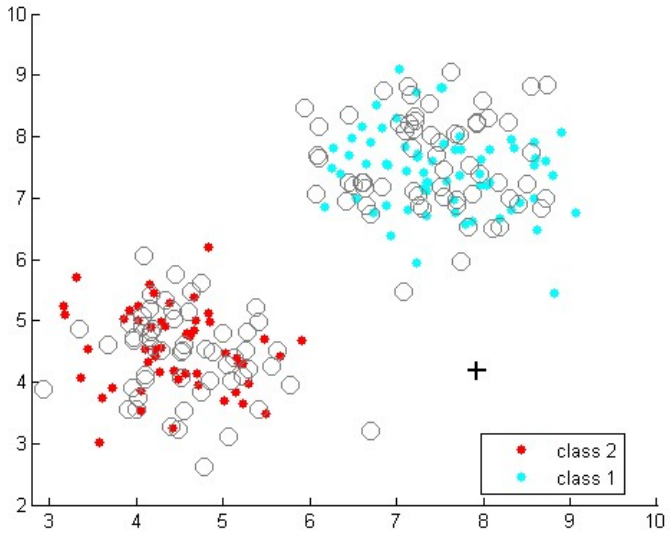
Machine Learning

EE 7345

Yang Du

Q1: Use the *k*-nearest neighbor classifier ($k = 4$) to classify each of the testing vectors in the file *testX1.mat*.

First of all, I use X_1 to generate classifier and then test the data. I set k=4 in KNN method.

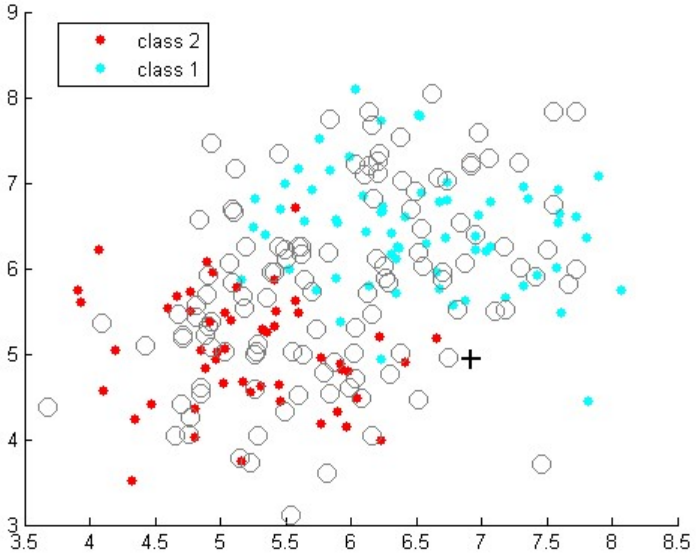


According to the picture above, I use function gscatter() to make classifier and then we define and draw new test vector. Based on the training data we defined the we can find that new data divide in to two clusters. By using tabulate() we can calculate the percentage of each part. The percent shows below:

Value	Count	Percent
class 1	63	52.50%
class 2	57	47.50%

This data is from testclasses.mat. I also count the number of class 1 which is 63. So the correct classification is 100%.

If we use *X2.mat, classes2.mat testX2.mat, testclasses2.mat.* The results are:



Value	Count	Percent
class 1	63	52.50%
class 2	57	47.50%

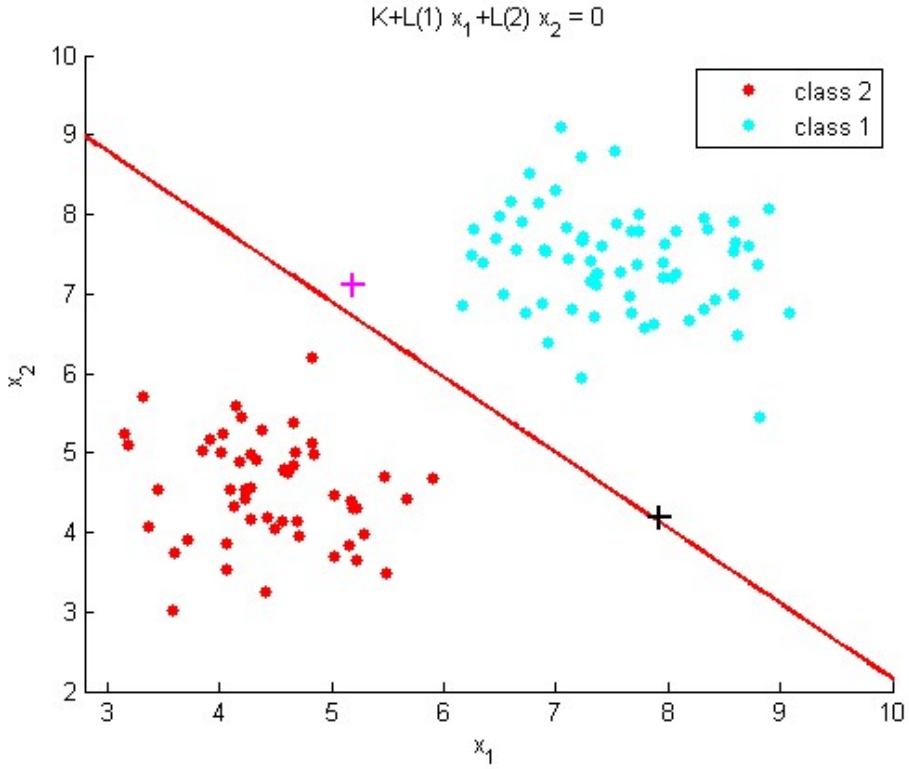
Matlab code

```
clear all;
clc;
load('X_1.mat');


load('classes_1.mat')
%draw training data
A=X;
gscatter(A(:,1),A(:,2),classes)
set(legend,'location','best')

%define and draw new test vector
load('Copy_of_tX_1.mat');
t=X;
testvec =X;
line(testvec(1),testvec(2),'marker','+','color','k','markersize',10,'
linewidth',2)
%do k-NN search
[n,d] = knnsearch(X,testvec,'k',4);
```

```
line(X(n,1),X(n,2),'color',[.5 .5 .5],'marker','o','linestyle','none'
,'markersize',10)
load('testclasses_1.mat');

%tabulate results


load('classes_1.mat')
tabulate(classes)
```

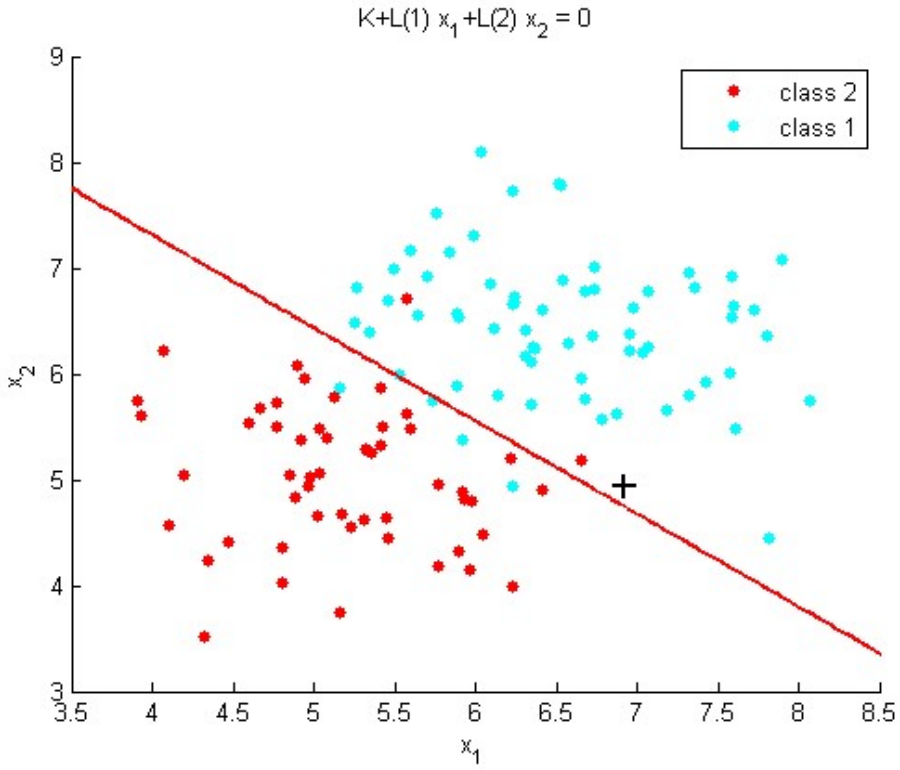
Q2: Train a linear discriminate analysis-based classifier using the Experiment 1 data and determine the percent correct classification. In addition, draw a scatter plot showing the decision boundary for the training data

I change the code in the slides and form the diagram by using discriminate method.



By comparing the test data classification and the classification in classes_1.mat. The correction percentage is 100%.

If we change to another data. The result is:



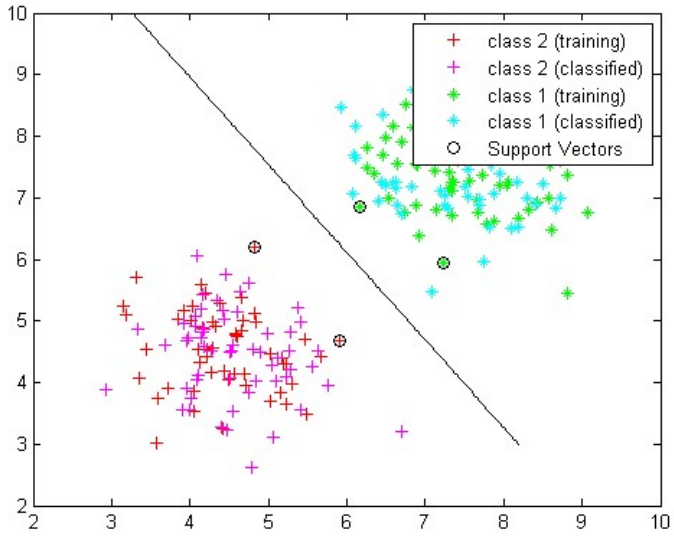
In these data, we find the result is different in our classification and the result in classes_2.mat. There are two classification is different which mean correct classification is 118/120 equals to 98.33%.

Matlab code

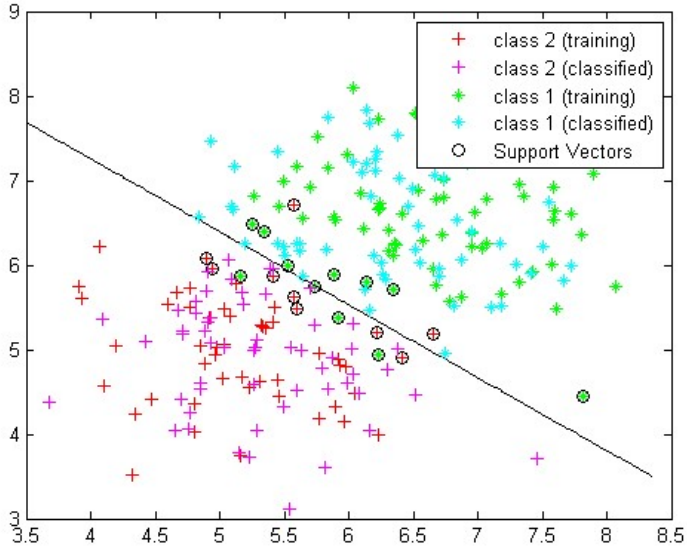
```
clear all;
load('X_2.mat');


load('classes_2.mat');

%get LDA model
A=X;
gscatter(A(:,1),A(:,2),classes)
cls = ClassificationDiscriminant.fit(X,classes);
hold on
K = cls.Coeffs(1,2).Const;
L = cls.Coeffs(1,2).Linear;
f = @(x1,x2) K + L(1)*x1 + L(2)*x2;
h2 = ezplot(f, [0 12 0 12]);
set(h2,'Color','r','linewidth',2)
%define and draw new test vector
load('testX_2.mat');
```

```
%load('testclasses_1.mat')
b=X;
testvec = b;
line(testvec(1),testvec(2),'marker','+','color','k','markersize',10,'
linewidth',2)

testclass = predict(cls, testvec)
```

Q3: Repeat problem 2 using support vector machines. Use a linear decision boundary. In experiment 1:



In experiment 2:



Matlab code:

```
clear all;
load('X_2.mat');
load('classes_2.mat');
%train svm

mysvm =
svmtrain(X,classes,'Kernel_Function','linear','showplot',true,'boxcon
straint',2);
load('testX_2.mat');
load('testclasses_2.mat');
xdata=X;
group=classes;
%svmStruct=svmtrain(xdata,group,'showplot',true)
Group = svmclassify(mysvm,xdata,'showplot',true)
```

Q5: Compare and contrast the various classification methods.

	advantages	disadvantages
SVM	1. high accuracy with kernel function 2. high quality of classification	1. not easy to set parameters 2. big memory consumption
LDA	1.optimal dispersibility 2. no limitation for distribution and variance	
KNN	1.Easy to use, do not need parameters and training 2. suitable for rare events 3. data has multiple labels and multi-modal.	1. heavy calculation work 2. if two samples are unbalanced extremely, the classification of test data may be wrong