Question 1

To exploit the possibility the cost function can be reduced by using only the difference in mean values of each class I made one example by generating two classes in two-dimensional feature space. The results are shown in Fig. 1-1 and Fig. 1-2. The Fig 1-1 shows the result when we use w0 obtained by minimizing the cost function J which uses both mean and variance of each class as in D.H.S. text. The projected data are well separated by two classes. From the figure we can clearly observe the projected data are well clustered around the mean of each class. The reason is that w0 is chosen so that the variance of projected data in each class is minimized, as well as the difference in mean value of each class. Whereas, the Fig. 1-2, obtained by minimizing the cost function J which has only the difference in mean of each class, shows that the projected data is not well separated. Although the difference in the mean of each class is smaller than the result shown in Fig. 1-1, the data is widely separate in the projected line. The conclusion is that we need to use both the variance and mean in constructing the cost function J.

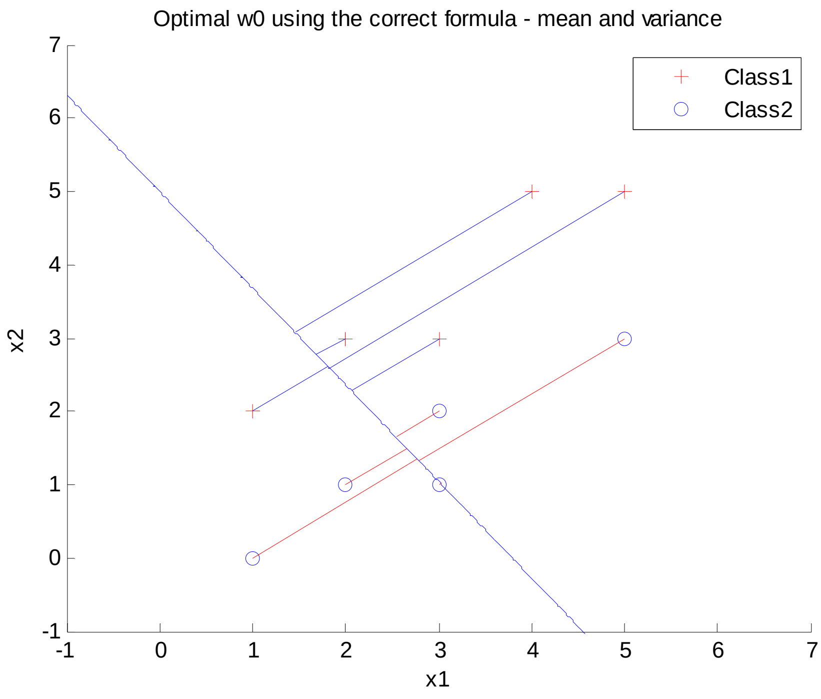


Fig. 1-1 The Optimal w0 is found by using J which has mean and variance value

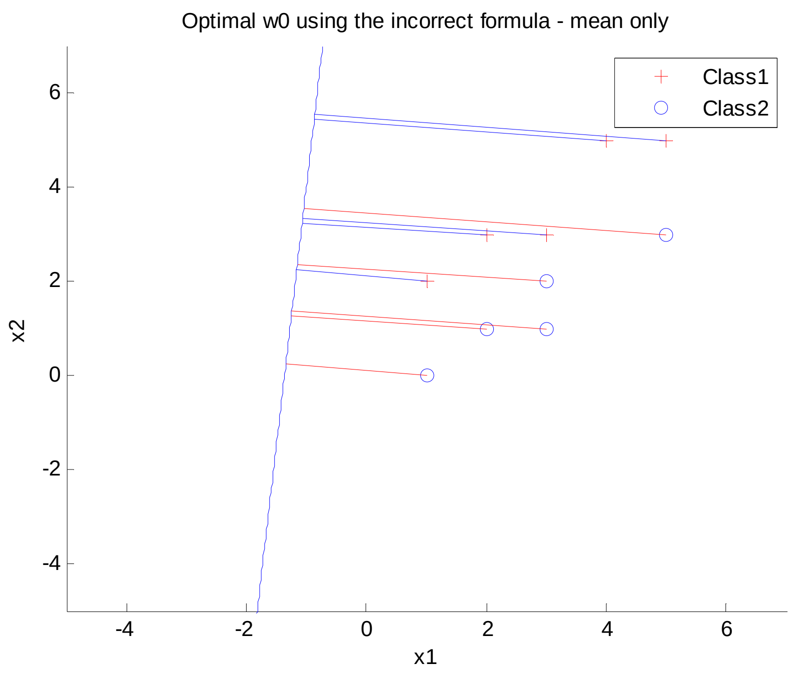


Fig. 1-2 The Optimal w0 is found by using J which has only mean value

<u>Question 2</u>

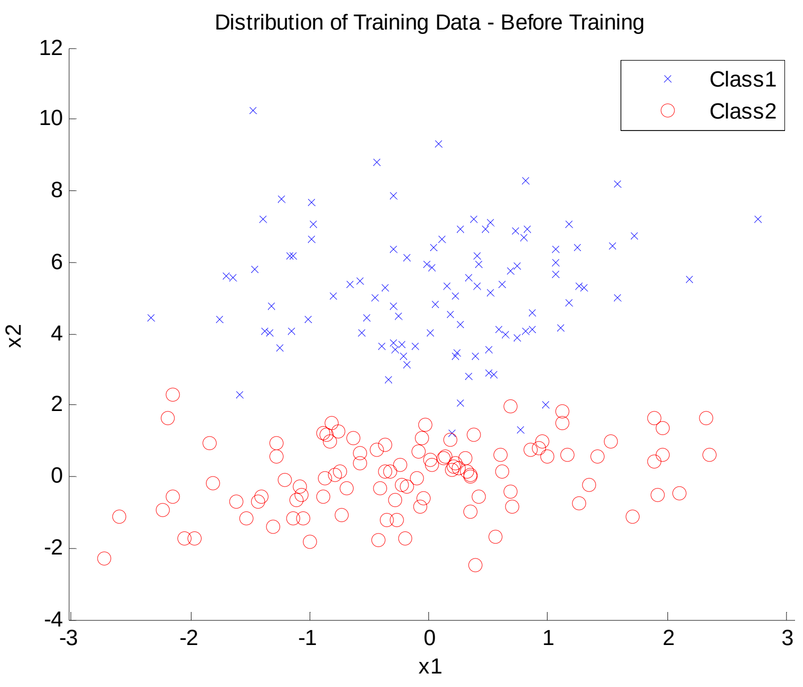
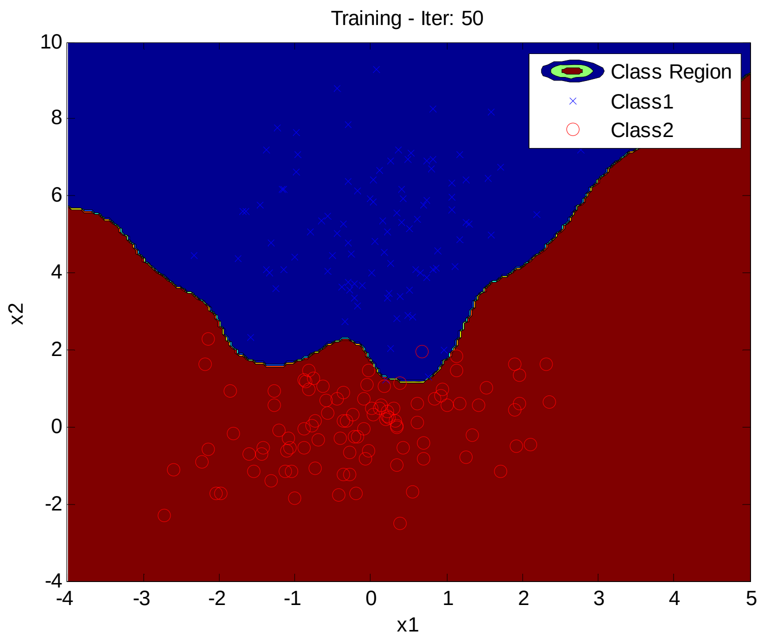
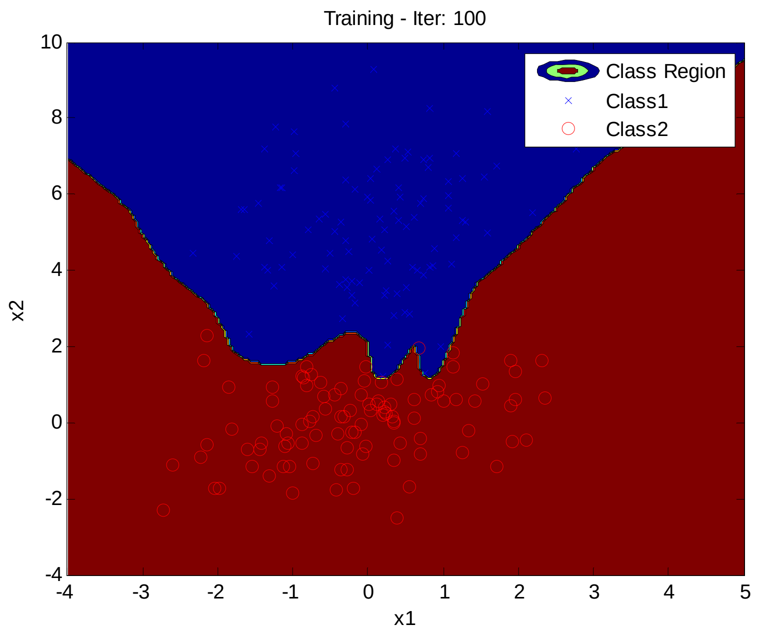
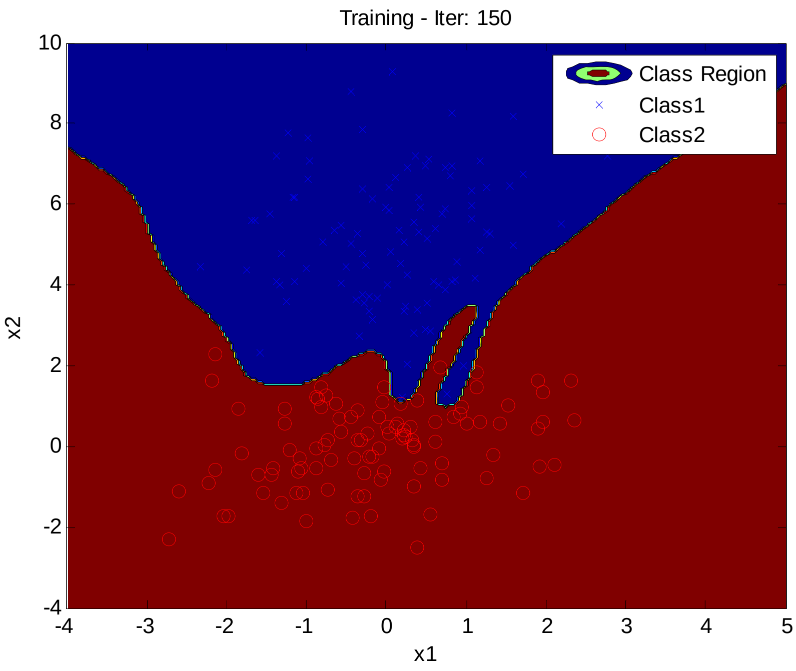


Fig. 2-1 Training data used for Question 2 and Question 3

a) The distribution of training data is shown in Fig. 2-1. There are two classes, where each class has Gaussian distribution with different mean and covariance matrix. The training data are fed to a Neural Network which has 50 neurons in hidden layer and two neurons in output layer. To train the neural network back propagation algorithm are used and weights are updated per every 200 epoch, where each epoch has five input data.







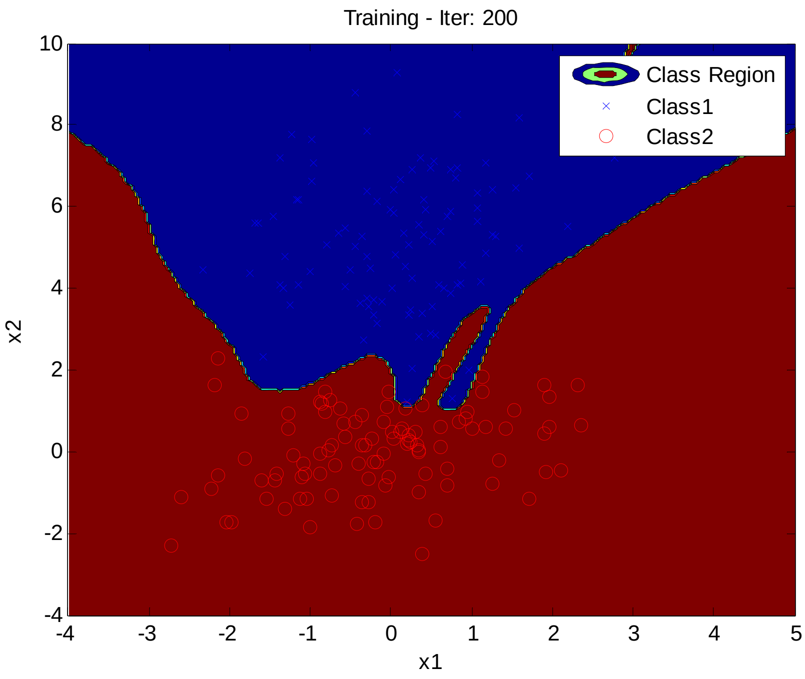


Fig. 2-2 Training results at the epoch = 50,100,150,200

The Fig. 2-2 shows the change of decision boundary at different epochs. The boundary is changing so that the training error is reduced.

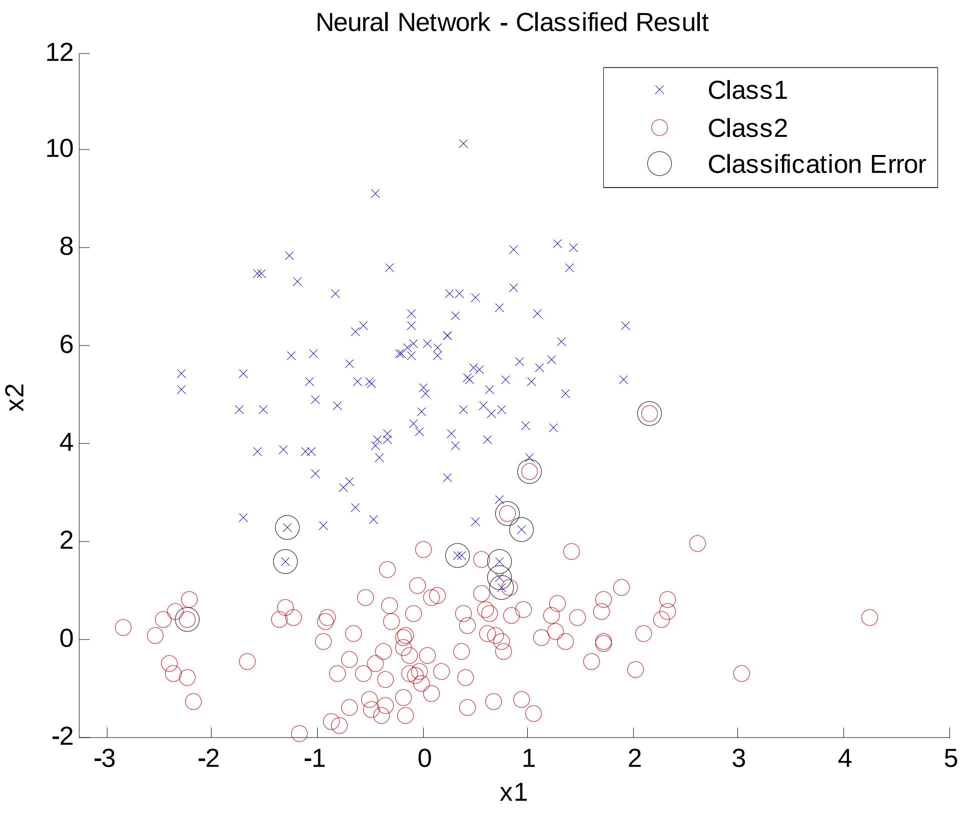


Fig. 2-3 Classification result using Neural Network (Error = 0.055)

The Fig. 2-3 shows the classification result for the test data. The classification error is marked by a large circle and the error rate is 0.055.

b) The training result using SVM are shown in Fig. 2-4. The support vectors are marked by a large circle, which is located near the decision boundary. For the training data set, the SVM is trained very quickly since the set is not complicated. The classification result for the test data is shown in Fig. 2-5. The classification errors occur near decision boundary. The error rate is 0.02, which is smaller than the result using a Neural Network.

- See next page -

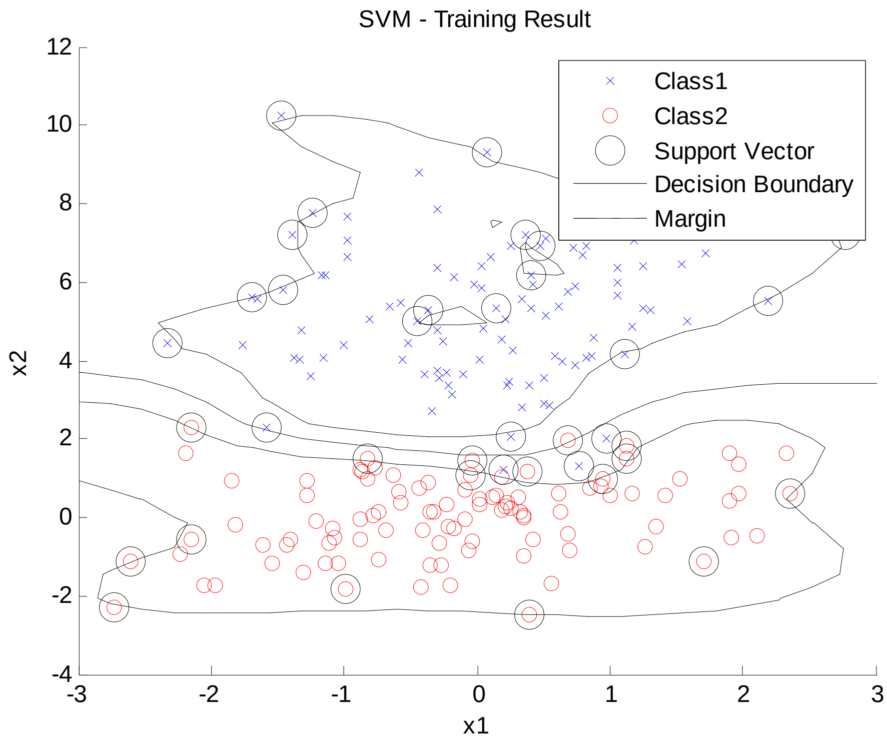


Fig. 2-4 Training result using SVM

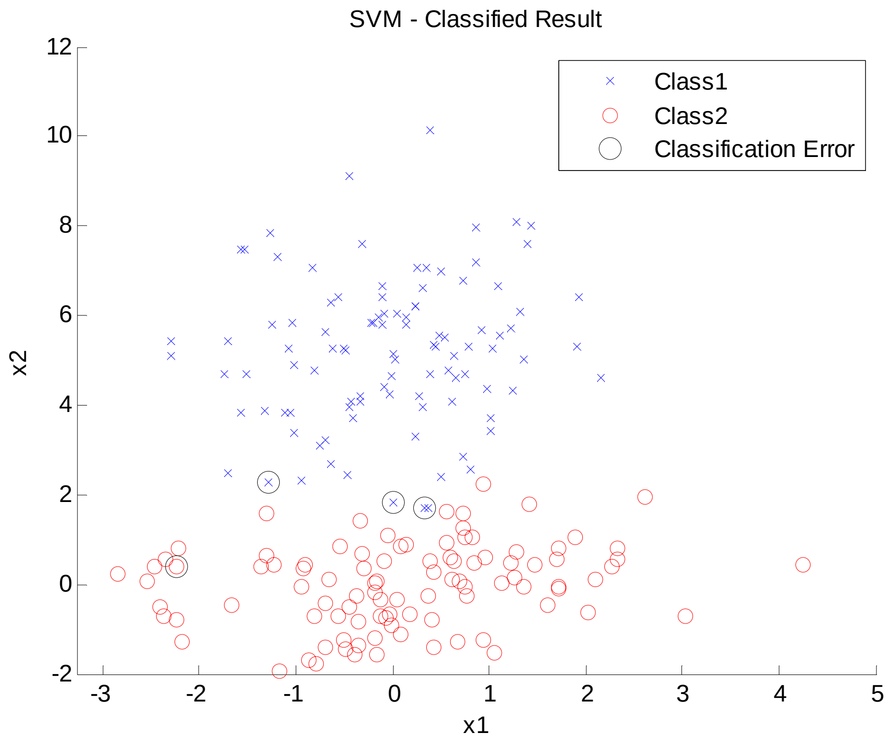


Fig. 2-5 Classification result using SVM (Err = 0.02)

c) In the simulation the SVM shows better result than Neural Network. However, the performance is highly dependent on the nature of data set. For the data set used in this simulation the SVM shows better generalization than Neural Network. Also, SVM does not suffer from being stuck into a local minimum, since the cost function always has a global minimum. The Neural Network shows different result depending on the initial value of weights, the epoch number, and the number of hidden layers. In terms of he computational burden the complexity of SVM increases significantly as the data are complicated to classify. In this example, the SVM is trained and runs faster than Neural Network since the data set is not so complex to classify.

Question 3

a) The classification result using Parzen window is shown in Fig. 3-1. The choice of h1 may be a big issue. In our experiment, it is set to 1. The classification error rate is 0.02

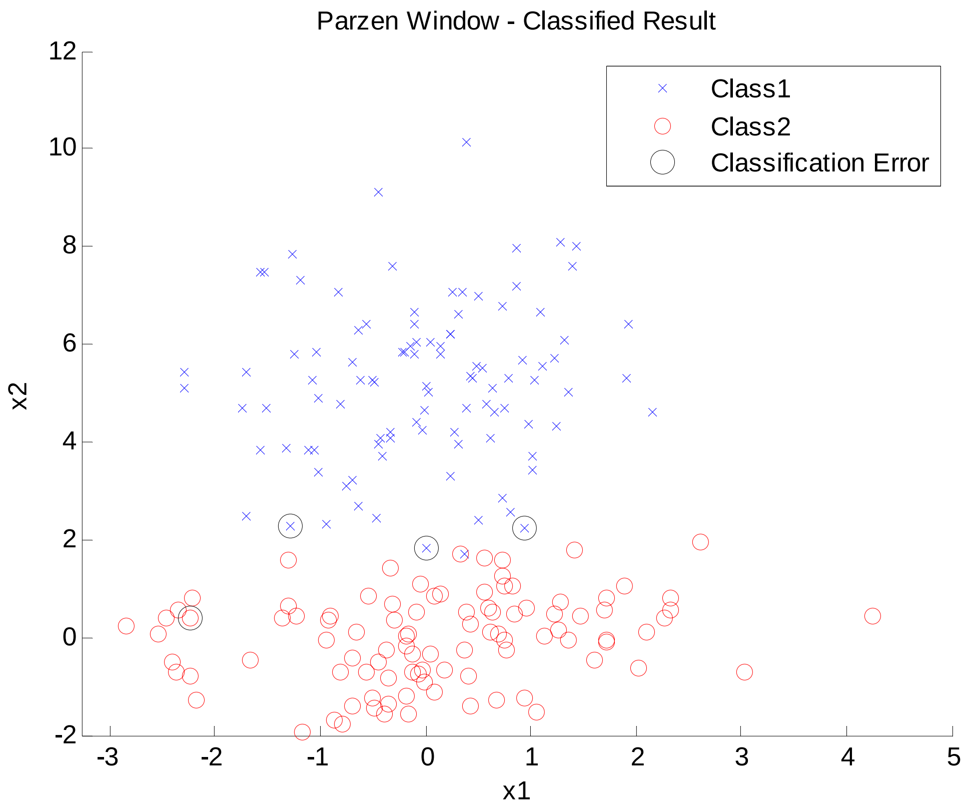


Fig. 3-1 Classification Result using Parzen Window (Err = 0.02)

b) The Fig. 3-2 shows the classification result using KNN. The number of nearest sample is set to 5. The error rate is 0.025.

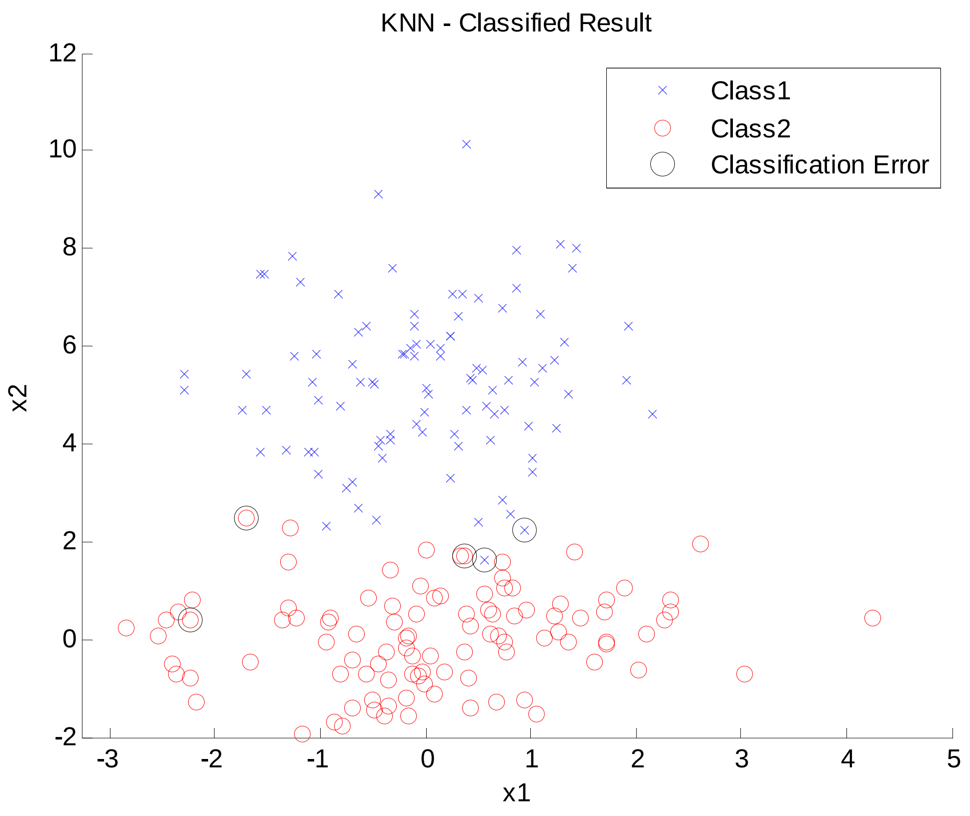


Fig. 3-2 Classification Result using KNN (Err = 0.025)

c) The software is downloaded from following site.

http://stuff.mit.edu/afs/sipb.mit.edu/user/arolfe/matlab.

The classification result using NN is shown in Fig. 3-3. The error rate is 0.045.

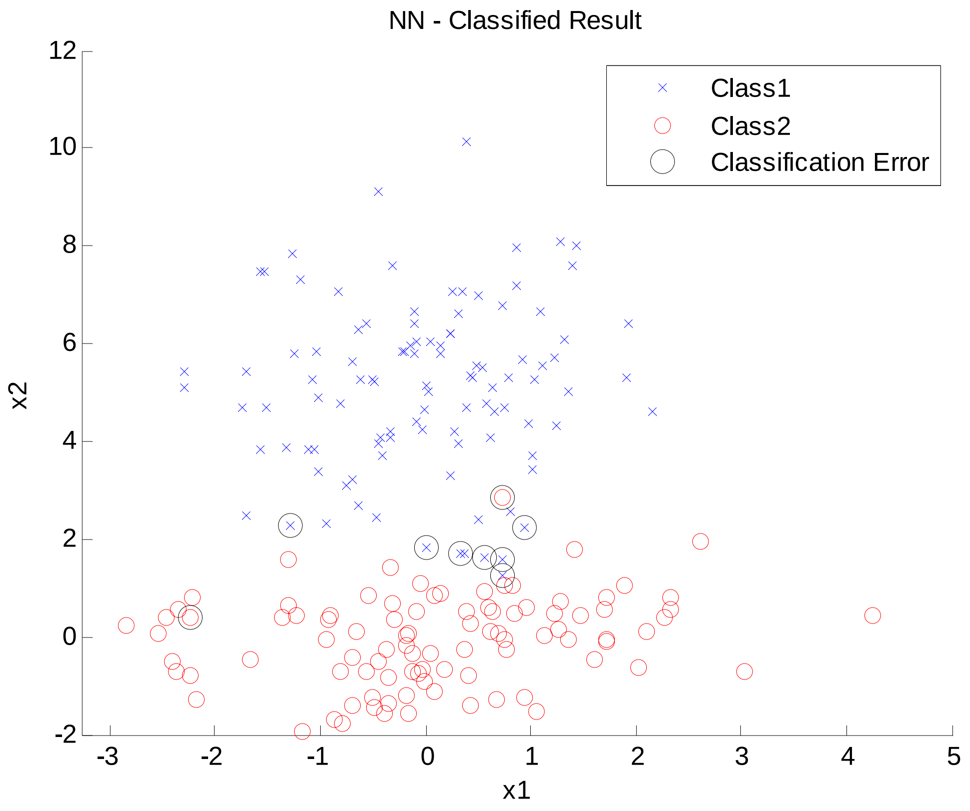


Fig. 3-3 Classification Result using NN (Err – 0.045)

d) The performance of Parzen technique, KNN, and NN is highly dependent on the particular data set and the number of training samples. In our experiment, Parzen works better than the other methods, but it is not a general rule. In region, where the data is sparse, KNN gives smooth estimate of probability density function, which may or may not result in errors for true test data. We can see from Fig. 3-2 there is a classification error where samples are sparse, but Parzen and NN cause no error at the same position. Also, the performance of Parzen technique is highly dependent on the value of h1, which determines the smoothness of estimated probability density function. In contrast to parametric method, these three methods can be used with arbitrary distributions and without knowing the parametric form of the underlying densities.

HW2 - 1

clear memory

```
clear all;
close all;
```

initialize parameters

```
n_points = 1000;
delta = 0.01;
```

generate two classes

```
class1 = [1 2; 2 3; 3 3; 4 5; 5 5];
class2 = [1 0; 2 1; 3 1; 3 2; 5 3];
[num,dim] = size(class1);
mu1 = mean(class1);
mu2 = mean(class2);
sig1 = cov(class1);
sig2 = cov(class2);
```

1) J - mean and variance

```
sw = sig1+sig2;
w0 = inv(sw)*(mu1-mu2)';

x_axis = -2+[0:n_points-1]*delta;
y_axis = w0(2)/w0(1)*x_axis+5 ;

for i = 1:num
   for j = 1:n_points
      d(j,i) = sum((class1(i,:)-[x_axis(j) y_axis(j)]).^2);
   end
end
```

```
[class1_value class1_index] = min(d);


for i = 1:num
   for j = 1:n_points
      d(j,i) = sum((class2(i,:)-[x_axis(j) y_axis(j)]).^2);
   end
end
[class2_value class2_index] = min(d);


figure(1)
hold on
axis([-1 7 -1 7]);
plot(class1(:,1),class1(:,2),'r+',class2(:,1),class2(:,2),'bo')
plot(x_axis,y_axis)
for i = 1:num
   A = zeros(2,2);
   A(1,:) = class1(i,:);
   A(2,:) = [x_axis(class1_index(i)) y_axis(class1_index(i))];
   plot(A(:,1),A(:,2),'b-')
end


for i = 1:num
   A = zeros(2,2);
   A(1,:) = class2(i,:);
   A(2,:) = [x_axis(class2_index(i)) y_axis(class2_index(i))];
   plot(A(:,1),A(:,2),'r-')
end
title('Optimal w0 using the correct formula - mean and variance');
legend('Class1','Class2');
xlabel('x1');
ylabel('x2');
hold off
```

2) J - mean only

```
w1 = (mu1-mu2);
y_axis = w1(2)/w1(1)*x_axis + 15 ;
for i = 1:num
   for j = 1:n_points
      d(j,i) = sum((class1(i,:)-[x_axis(j) y_axis(j)]).^2);
   end
end
[class1_value class1_index] = min(d);

for i = 1:num
   for j = 1:n_points
      d(j,i) = sum((class2(i,:)-[x_axis(j) y_axis(j)]).^2);
   end
end
[class2_value class2_index] = min(d);

figure(2)
hold on
axis([-5 7 -5 7]);
plot(class1(:,1),class1(:,2),'r+',class2(:,1),class2(:,2),'bo')
plot(x_axis,y_axis)
for i = 1:num
   A = zeros(2,2);
   A(1,:) = class1(i,:);
   A(2,:) = [x_axis(class1_index(i)) y_axis(class1_index(i))];
   plot(A(:,1),A(:,2),'b-')
end

for i = 1:num
   A = zeros(2,2);
   A(1,:) = class2(i,:);
   A(2,:) = [x_axis(class2_index(i)) y_axis(class2_index(i))];
   plot(A(:,1),A(:,2),'r-')
end
title('Optimal w0 using the incorrect formula - mean only');
```

```
legend('Class1','Class2');

xlabel('x1');

ylabel('x2');

hold off
```

HW2 - 2

clear memory

```
clear all;
close all;
```

add path

```
addpath('./nn');
addpath('./stprtool');
stprpath('./stprtool');
```

generate test and training samples

```
n_samples = 200;
n_train_samples = round(n_samples/2);
n_test_samples = n_samples - n_train_samples;
mu1 = [0 5];
sigma1 = [1 0.3; 0.3, 3];
x1 = mvnrnd(mu1, sigma1, n_samples);
t1 = ones(1,n_samples);
mu2 = [0 0];
sigma2 = [1.5 0.3; 0.3, 1];
x2 = mvnrnd(mu2, sigma2, n_samples);
t2 = ones(1,n_samples) * 2;
x_train = [x1(1:n_train_samples,:)', x2(1:n_train_samples,:)'];
t_train = [t1(1:n_train_samples), t2(1:n_train_samples)];
x_test                   =                   [x1(n_train_samples+1:n_samples,:)',
x2(n_train_samples+1:n_samples,:)'];
t_test = [t1(n_train_samples+1:n_samples), t2(n_train_samples+1:n_samples)];
mu = [mu1;mu2];
sigma = [sigma1;sigma2];
```

```
trn.X = x_train;

trn.nanme = 'train';

trn.y = t_train;

trn.dim = 2;

trn.num_data = n_train_samples;


tst.X = x_test;

tst.nanme = 'test';

tst.y = t_test;

tst.dim = 2;

tst.num_data = n_test_samples;


save data.mat trn tst mu sigma n_samples n_train_samples n_test_samples;
```

classify using nn

```
nn_layer = [50, 2];

num_iter = 200;

net = newff(minmax(x_train), [50, 2], {'logsig', 'logsig'}, 'trainbfg');

net.performFcn = 'mse';

net.trainParam.epochs = 5;

net.trainParam.show = NaN;

net = init(net);

net = nnt_classify(net,trn,mu,sigma,num_iter);

out = sim(net,x_test);

[maxVal,nn_test_idx] = max(out);


figure;

gscatter(x_test(1,:), x_test(2, :), nn_test_idx, 'br', 'xo');

hold on;

nn_test_err_idx = find(nn_test_idx ~=  t_test);

x_err_test = x_test(:,nn_test_err_idx);

plot(x_err_test(1,:),x_err_test(2,:),'ok','MarkerSize',10);

title('Neural Network - Classified Result');

xlabel('x1');

ylabel('x2');

legend('Class1','Class2','Classification Error');
```

```
r = cerror(nn_test_idx,t_test);
info = sprintf('Neural Network Classification Error - %2.5f',r);
disp(info);
```

```
Neural Network Classification Error - 0.05500
```

classify using svm

```
options.ker = 'rbf';      % use RBF kernel
options.arg = 1;          % kernel argument
options.C = 10;           % regularization constant
model = smo(trn,options);
figure;
ppatterns(trn);
psvm(model);
title('SVM - Training Result');
xlabel('x1');
ylabel('x2');
legend('Class1','Class2','Support Vector','Decision Boundary','Margin');

svm_test_idx = svmclass(tst.X,model);

figure;
gscatter(x_test(1,:), x_test(2, :), svm_test_idx, 'br', 'xo');
hold on;
svm_test_err_idx = find(svm_test_idx ~=  t_test);
x_err_test = x_test(:,svm_test_err_idx);
plot(x_err_test(1,:),x_err_test(2,:),'ok','MarkerSize',10);
title('SVM - Classified Result');
xlabel('x1');
ylabel('x2');
legend('Class1','Class2','Classification Error');
r = cerror(svm_test_idx,tst.y);
info = sprintf('SVM Classification Error - %2.5f',r);
disp(info);
```

```
SVM Classification Error - 0.02500
```

HW2 - 3

clear memory

```
clear all;
close all;
```

load test data

```
load data.mat
x_train = trn.X;
t_train = trn.y;
x_test = tst.X;
t_test = tst.y;
```

add path

```
addpath('./nonparametic');
addpath('./stprtool');
stprpath('./stprtool');
```

initialize parameters

```
h   = 1;                    % The width of parzen window for PARZEN WINDOW
k   = 5;                    % The number of nearest neighbors for KNN
N   = n_samples ;             % The number of total training and test data(Class1-N,
Class2-N)
NT  = n_train_samples ;       % The number of training data(Class1_NT, Class2-NT)
```

classify using Parzen window

```
parzen_test_idx = parzen(h,x_train,t_train,x_test);
r = cerror(parzen_test_idx,t_test);
info = sprintf('Parzen Window Classification Error - %2.5f',r);
```

```
    disp(info);

    figure;
    gscatter(x_test(1,:), x_test(2, :), parzen_test_idx, 'br', 'xo');
    hold on;
    parzen_test_err_idx = find(parzen_test_idx ~=  t_test);
    x_err_test = x_test(:,parzen_test_err_idx);
    plot(x_err_test(1,:),x_err_test(2,:),'ok','MarkerSize',10);
    title('Parzen Window - Classified Result');
    xlabel('x1');
    ylabel('x2');
    legend('Class1','Class2','Classification Error');
```

```
Parzen Window Classification Error - 0.02500
```

classifiy using KNN

```
    knn_test_idx = knn(k,x_train,t_train,x_test);
    r = cerror(knn_test_idx,t_test);
    info = sprintf('KNN Classification Error - %2.5f',r);
    disp(info);

    figure;
    gscatter(x_test(1,:), x_test(2, :), knn_test_idx, 'br', 'xo');
    hold on;
    knn_test_err_idx = find(knn_test_idx ~=  t_test);
    x_err_test = x_test(:,knn_test_err_idx);
    plot(x_err_test(1,:),x_err_test(2,:),'ok','MarkerSize',10);
    title('KNN - Classified Result');
    xlabel('x1');
    ylabel('x2');
    legend('Class1','Class2','Classification Error');
```

```
KNN Classification Error - 0.03500
```

classify using nn

```
Range = x_train';     %Range = [data_num dimension]
Range_min = min(Range);
x1_min = Range_min(1);
y1_min = Range_min(2);

Range_max = max(Range);
x1_max = Range_max(1);
y1_max = Range_max(2);

Range = x_test';      %Range = [data_num dimension]
Range_min = min(Range);
x2_min = Range_min(1);
y2_min = Range_min(2);

Range_max = max(Range);
x2_max = Range_max(1);
y2_max = Range_max(2);

x_min = min(x1_min,x2_min);
x_max = max(x1_max,x2_max);
y_min = min(y1_min,y2_min);
y_max = min(y1_max,y2_max);
region5 = max(NT,(N-NT));

region = [x_min x_max y_min y_max];
region = [region region5];
Nclasses = 2;

D = nn(x_train,t_train, region);
figure;
x_axis = linspace(x_min,x_max);
y_axis = linspace(y_min,y_max);
contourf(x_axis,y_axis,D);
```

```
title('NN - Decison Boundary');
xlabel('x1');
ylabel('x2');


[train_err,  test_err,nn_test_idx]  =  calculate_error  (D,  x_train,  t_train,
x_test,t_test, region, Nclasses);
r = cerror(nn_test_idx,t_test);
info = sprintf('NN Classification Error - %2.5f',r);
disp(info);

figure;
gscatter(x_test(1,:), x_test(2, :), nn_test_idx, 'br', 'xo');
hold on;
nn_test_err_idx = find(nn_test_idx ~=  t_test);
x_err_test = x_test(:,nn_test_err_idx);
plot(x_err_test(1,:),x_err_test(2,:),'ok','MarkerSize',10);
title('NN - Classified Result');
xlabel('x1');
ylabel('x2');
legend('Class1','Class2','Classification Error');
```

```
NN Classification Error - 0.04000
```