Zhiqian Zhou

HW5 Question 1

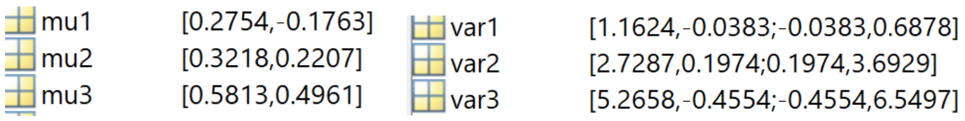
Load Data

```
TrainPts = Data.train;
TestPts = Data.test;

% load and transpose the matrix
train1 = TrainPts{1,1}';
train2 = TrainPts{1,2}';
train3 = TrainPts{1,3}';
test1 = TestPts{1,1}';
test2 = TestPts{1,2}';
test3 = TestPts{1,3}';
```

a) Maximum Likelihood Estimates

```
% a) maximum likelihood estimates
mu1 = mean(train1);
mu2 = mean(train2);
mu3 = mean(train3);
mu = repmat(mu1,size(train1,1),1);
var1 = (train1-mu)'*(train1-mu) ./ size(train1,1);
mu = repmat(mu2,size(train2,1),1);
var2 = (train2-mu)'*(train2-mu) ./ size(train2,1);
mu = repmat(mu3,size(train3,1),1);
var3 = (train3-mu)'*(train3-mu) ./ size(train3,1);
```



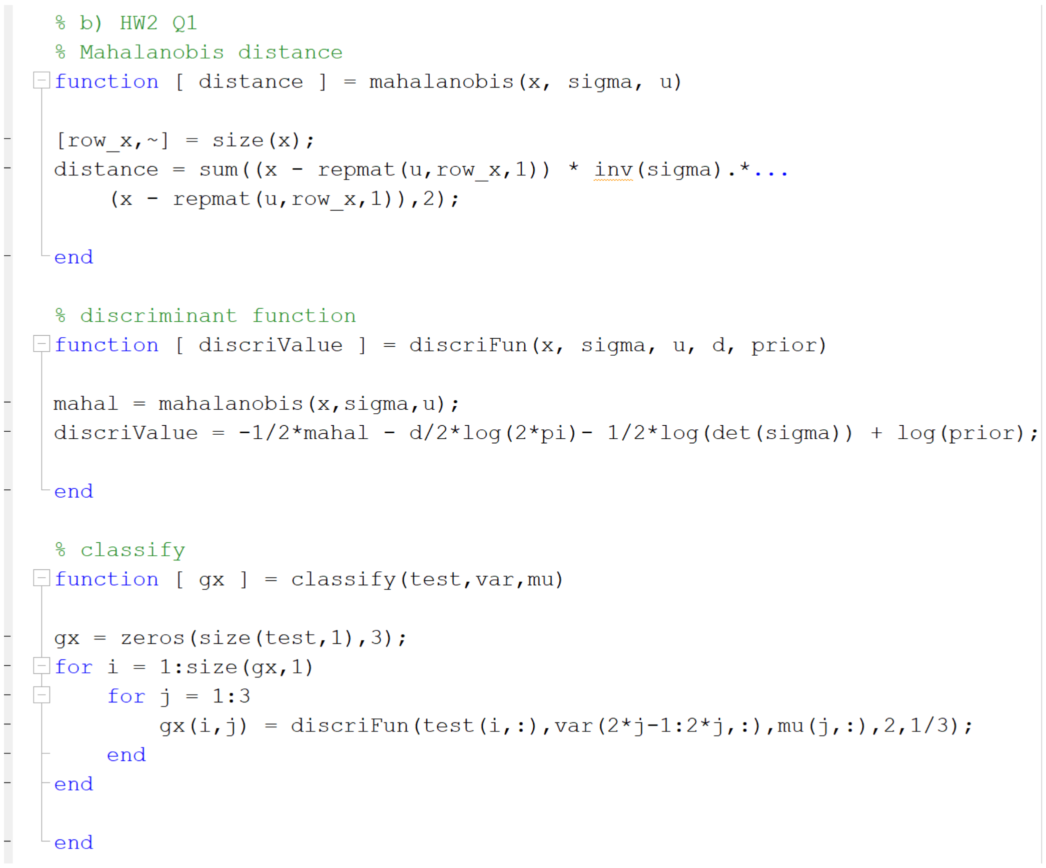
mu1	[0.2754,-0.1763]	var1	[1.1624,-0.0383;-0.0383,0.6878]
mu2	[0.3218,0.2207]	var2	[2.7287,0.1974;0.1974,3.6929]
mu3	[0.5813,0.4961]	var3	[5.2658,-0.4554;-0.4554,6.5497]

b) HW2 Question 1

Write function to compute the Mahalanobis distance, and another function to call the function above and compute the discrimination function with the following generic form.

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1} (x-\mu_i) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

For the return value of classify function, the row of gx represent the gi(x) value for each x, and the column of gx represent its value of discrimination function for different class.



```
% b) HW2 Q1
% Mahalanobis distance
function [ distance ] = mahalanobis(x, sigma, u)

[row_x,~] = size(x);
distance = sum((x - repmat(u,row_x,1)) * inv(sigma).*...
    (x - repmat(u,row_x,1)),2);

end

% discriminant function
function [ discriValue ] = discriFun(x, sigma, u, d, prior)

mahal = mahalanobis(x,sigma,u);
discriValue = -1/2*mahal - d/2*log(2*pi)- 1/2*log(det(sigma)) + log(prior);

end

% classify
function [ gx ] = classify(test,var,mu)

gx = zeros(size(test,1),3);
for i = 1:size(gx,1)
    for j = 1:3
        gx(i,j) = discriFun(test(i,:),var(2*j-1:2*j,:),mu(j,:),2,1/3);
    end
end

end
```

c) Classify the test data

The variables, index1, index2, index3, represent which class the data was classified into.

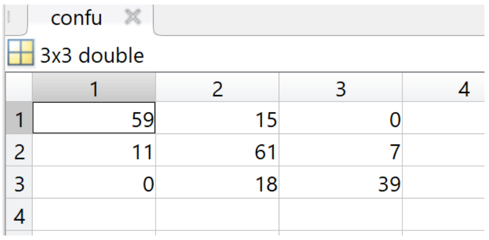
```
% c) classify
mu = [mu1;mu2;mu3];
var = [var1;var2;var3];
gx1 = classify(test1,var,mu);
gx2 = classify(test2,var,mu);
gx3 = classify(test3,var,mu);
[max_g1,index1] = max(gx1,[],2);
[max_g2,index2] = max(gx2,[],2);
[max_g3,index3] = max(gx3,[],2);
```

Compute confusion matrix, each column is actual value and each row is estimate value.

```
% confusion matrix
confu = zeros(3,3);
for i = 1:size(index1,1)
    confu(index1(i,1),1) = confu(index1(i,1),1) + 1;
end

for i = 1:size(index2,1)
    confu(index2(i,1),2) = confu(index2(i,1),2) + 1;
end

for i = 1:size(index3,1)
    confu(index3(i,1),3) = confu(index3(i,1),3) + 1;
end
```



	1	2	3	4
1	59	15	0	
2	11	61	7	
3	0	18	39	
4				

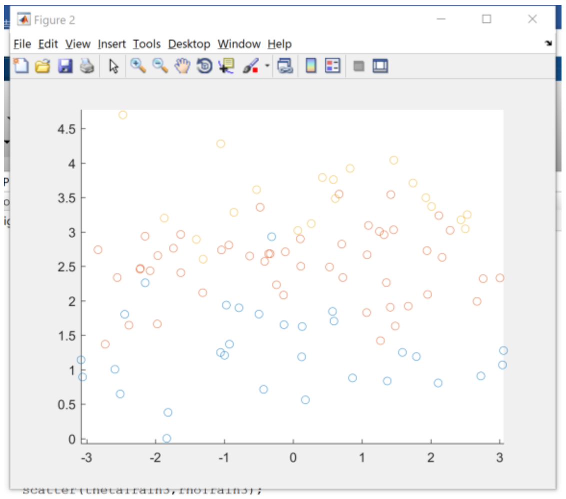
d) Bayesian estimates

Transform data into the polar coordinates and scatter the data.

```
% d) Bayesian estimates
[thetaTrain1, rhoTrain1] = cart2pol(train1(:,1),train1(:,2));
[thetaTrain2, rhoTrain2] = cart2pol(train2(:,1),train2(:,2));
[thetaTrain3, rhoTrain3] = cart2pol(train3(:,1),train3(:,2));
[thetaTest1, rhoTest1] = cart2pol(test1(:,1),test1(:,2));
[thetaTest2, rhoTest2] = cart2pol(test2(:,1),test2(:,2));
[thetaTest3, rhoTest3] = cart2pol(test3(:,1),test3(:,2));

% scatter figure after tranform
figure(2);
scatter(thetaTrain1,rhoTrain1);
hold on;
scatter(thetaTrain2,rhoTrain2);
hold on;
scatter(thetaTrain3,rhoTrain3);
axis equal,
hold off;

rhoMean1 = mean(rhoTrain1);
rhoMean2 = mean(rhoTrain2);
rhoMean3 = mean(rhoTrain3);
```



Ignore the θ and classify based only on r.

Calculate the θ and σ .

```
% estimate for mu
n = size(rhoTrain1,1);
rhoVar1 = (0.25 * 100) / (n*100 + 0.25);
rhoMu1 = (n * rhoMean1 / 0.25) * rhoVar1;
n = size(rhoTrain2,1);
rhoVar2 = (0.25 * 100) / (n*100 + 0.25);
rhoMu2 = (n * rhoMean2 / 0.25) * rhoVar2;
n = size(rhoTrain3,1);
rhoVar3 = (0.25 * 100) / (n*100 + 0.25);
rhoMu3 = (n * rhoMean3 / 0.25) * rhoVar3;
```

rhoMu1	1.2673	rhoVar1	0.0083
rhoMu2	2.5134	rhoVar2	0.0050
rhoMu3	3.4866	rhoVar3	0.0125

The posterior distribution.

```
% posterier function
sym mu
post1 = 1 / sqrt(2*pi*rhoVar1) * exp(-1/2 * power((mu-rhoMu1),2) / rhoVar1);
post2 = 1 / sqrt(2*pi*rhoVar2) * exp(-1/2 * power((mu-rhoMu2),2) / rhoVar2);
post3 = 1 / sqrt(2*pi*rhoVar3) * exp(-1/2 * power((mu-rhoMu3),2) / rhoVar3);
```

Compute density estimate to classify.

```
% density estimate
function [ postrior ] = bayesClassify(test,mu,var)

postrior = zeros(size(test,1),3);
for i = 1:size(test,1)
    for j = 1:3
        postrior(i,j) = 1 / (2*pi*sqrt(var(j,1)*0.25)) *...
            exp(-1/2 * power((test(i,1)-mu(j,1)),2) / (var(j,1)+0.25));
    end
end

end
```

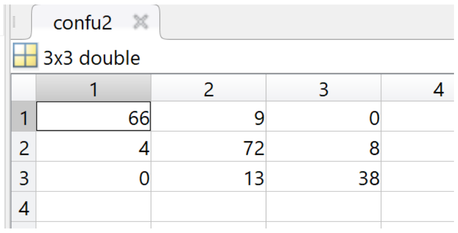
```
% classify
mun = [rhoMu1; rhoMu2; rhoMu3];
varn = [rhoVar1; rhoVar2; rhoVar3];
posterior1 = bayesClassify(rhoTest1,mun,varn);
posterior2 = bayesClassify(rhoTest2,mun,varn);
posterior3 = bayesClassify(rhoTest3,mun,varn);
[max_p1,index1] = max(posterior1,[],2);
[max_p2,index2] = max(posterior2,[],2);
[max_p3,index3] = max(posterior3,[],2);
```

Confusion Matrix, also actual for column and estimate for row.

```
confu2 = zeros(3,3);
for i = 1:size(index1,1)
    confu2(index1(i,1),1) = confu2(index1(i,1),1) + 1;
end

for i = 1:size(index2,1)
    confu2(index2(i,1),2) = confu2(index2(i,1),2) + 1;
end

for i = 1:size(index3,1)
    confu2(index3(i,1),3) = confu2(index3(i,1),3) + 1;
end
```


confu2

3x3 double

	1	2	3	4
1	66	9	0	
2	4	72	8	
3	0	13	38	
4				

Compute and Compare the error rate of two methods

```
% error rate
error = zeros(3,1);
for i = 1:3
    error(i,1) = 1 - confu(i,i)/sum(confu(:,i));
end
```

```
% error rate
error2 = zeros(3,1);
for i = 1:3
    error2(i,1) = 1 - confu2(i,i)/sum(confu2(:,i));
end
```

```
error       [0.1571;0.3511;0.1522]
error2      [0.0571;0.2340;0.1739]
```

From the result above, we can see that the error rate after transformation is relatively lower than the likelihood estimates before the transformation. Besides, after the transformation, the problem becomes a 1-D classification problem. Therefore, proper transformation can both simplify the question and improve the accuracy.

We can see how importance it is to transform and select data in this case. It’s very important for us to learn knowledge from simple cases and then apply them to the problems in reality in the future.