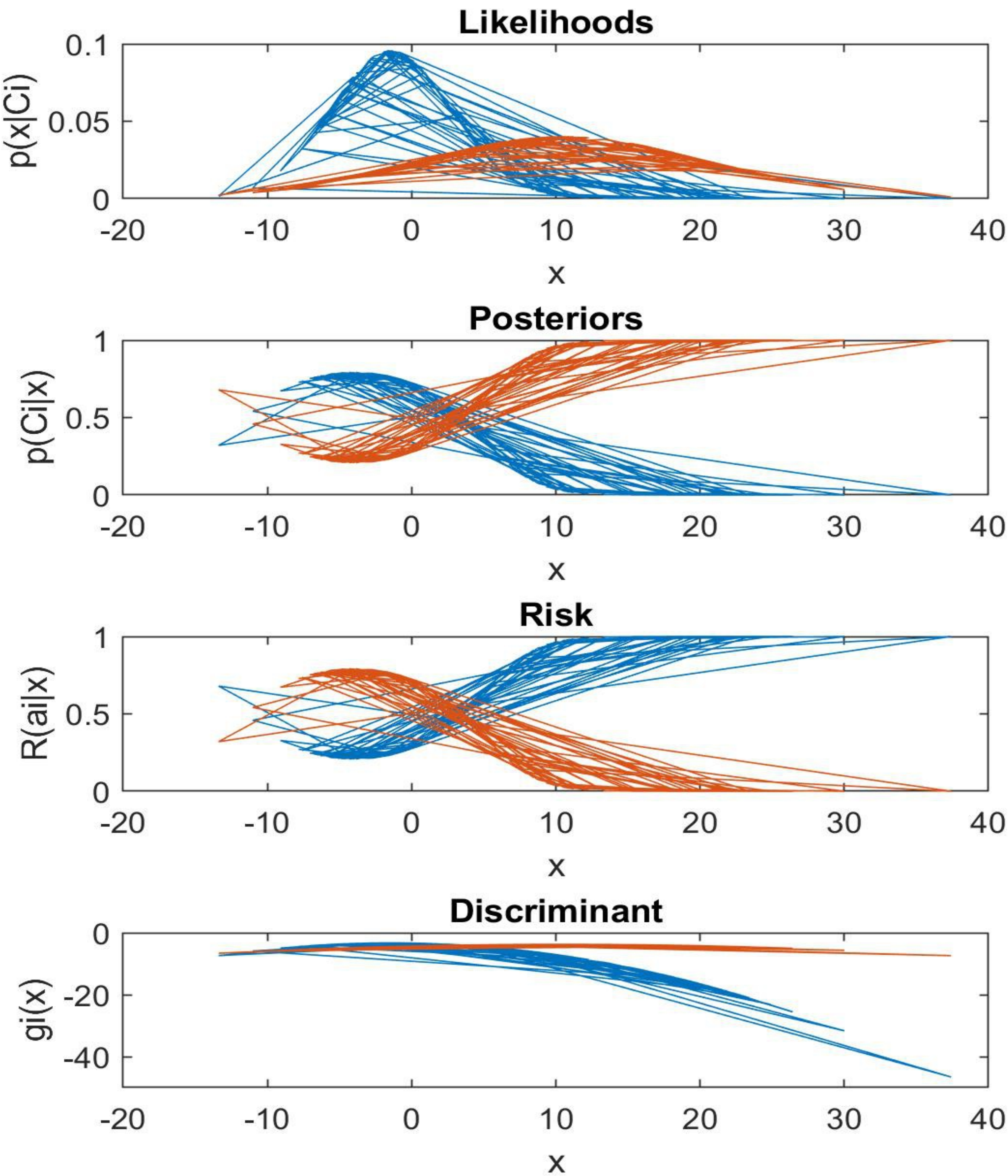
BM 59D Homework#1

1) a) The plots are displayed in the following figure:



The class boundaries are the followings:

likelihood_boundary =   4.5374
posterior_boundary =   3.4654
risk_boundary = 3.4654
discriminant_boundary = 3.4654

As shown in above, the likelihood boundary is different from the other boundaries. Because the posterior, risk and discriminant functions are calculated in the same way. If the priors are equal, all class boundaries have the same value: 4.5374. It is obvious that the prior affects posterior, risk and discriminant.

After solving the equation analytically, we obtain the class boundary as average of mean of class 1 and mean of class 2. In this data set, the class boundary is 4.516 (mean_1 =   -1.4519, mean_2 =   10.4839). Since the prior of class 1 is 0.4 and the prior of class 2 is 0.6, this value is close to the class 1 which has less likely mean.

The decision rule is calculated as in the following steps:

R(action_1 | x)
= Loss_1_1*P(Class_1 | x) +1* Loss_1_2*P(Class_2 | x)
= Loss_1_2*(1-P(Class_1 | x))

R(action_2 | x)
= Loss_2_1*P(Class_1 | x) + Loss_2_2*P(Class_2 | x)
= Loss_2_1*P(Class_1 | x)

Choose action_i:
if R(action_i | x) < R(action_k | x)

Choose action 1:
Loss_1_2 / (Loss_1_2 + Loss_2_1) < P(Class_1 | x)

Choose action 2:
Loss_2_1 / (Loss_1_2 + Loss_2_1) < P(Class_2| x)

Then the decision rules are:

½ < P(Class_1 | x)   and ½ < P(Class_2 | x)

As a result, if the two misclassifications are equally costly, the decision threshold would be at ½. The boundary is where the two posteriors are equal when both misclassifications are equally costly (figure 3.2.a in the book).

The confusion matrix of the training set:

	Prediction(+)	Prediction(-)
Actual(+)	46	4
Actual(-)	15	60

The confusion matrix of the validation set:

	Prediction(+)	Prediction(-)
Actual(+)	44	6
Actual(-)	14	61

b) The decision rules are:

1/3 < P(Class_1 | x) and 2/3 < P(Class_2 | x)

When the losses are not symmetric, the boundary shifts toward the class that incurs higher risk when misclassified (figure 3.2.b in the book).

The confusion matrix of the training set:

	Prediction(+)	Prediction(-)
Actual(+)	47	3
Actual(-)	22	53

The confusion matrix of the validation set:

	Prediction(+)	Prediction(-)
Actual(+)	47	3
Actual(-)	17	58

c) Choose action_i:

if R(action_i | x) < Loss_rejection

Choose action 1:

(Loss_1_2 - Loss_Rejection) /Loss_1_2 / < P(Class_1 | x)

Choose action 2:

P(Class_1| x) < Loss_rejection/Loss_2_1

The decision rules are:

Choose action 1 P(Class_1 | x) > 0.6
Choose action 2 P(Class_1 | x) < 0.2
Reject 0.2 < P(Class_1 | x) < 0.6

When there is the option of reject, a region around the boundary is the region of reject.(figure 3.2.c in the book).

The confusion matrix of the training set:

	Prediction(+)	Prediction(-)	Rejection
Actual(+)	41	2	7
Actual(-)	11	45	19

The confusion matrix of the validation set:

	Prediction(+)	Prediction(-)	Rejection
Actual(+)	39	0	11
Actual(-)	11	52	12

d) For the training set:

TP=46, FP=15, FN=4, TN=60

Sensitivity = TP /(TP+FN) = 46/50

Specifity = TN/(TN+FP) = 60/75

PPV = TP /(TP+FP) =46/61

NPV =TN/(TN+FN)=60/64

Accuracy = (TN+TP)/(N+P)=96/125

	0/1 Loss	Asymmetric Loss	Asymmetric Loss & Rej
Sensitivity	46/50	47/50	41/43
Specifity	60/75	53/75	45/56
PPV	46/61	47/69	41/52
NPV	60/64	53/56	45/47
Accuracy	96/125	100/125	86/99

Asymmetric loss and rejection improves all rates, which means that penalizing the misclassifications with the high cost may prevent wrong decisions.

2)

Error Matrix =

	0	1	2	3	4
Sample1	7.62732	7.54204	7.52325	7.73008	7.56194
Sample2	-6.17979	-6.54159	-6.49031	-6.35304	-6.66891
Sample3	-20.16665	-25.82632	-24.12145	-28.51871	-28.67248

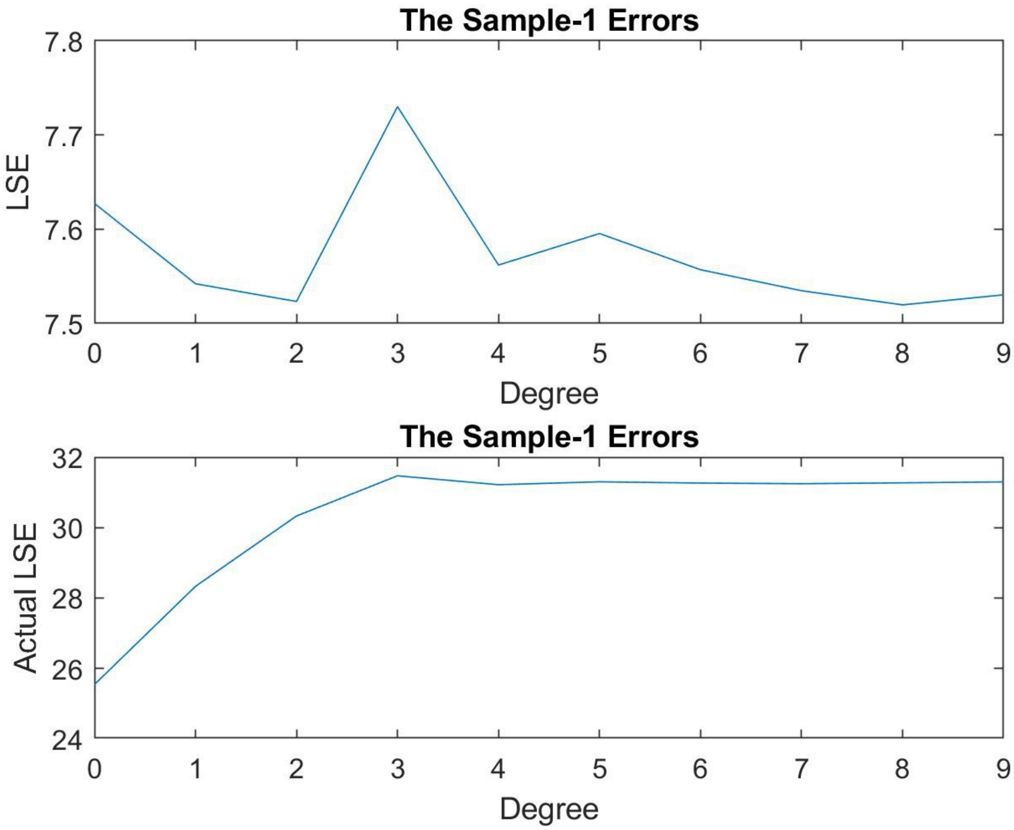
	5	6	7	8	9
Sample1	7.59536	7.55696	7.53468	7.51963	7.53030
Sample2	-6.61645	-6.81161	-6.82358	-6.89299	-7.10329
Sample3	-28.47537	-30.56927	-30.71266	-30.68396	-31.54414

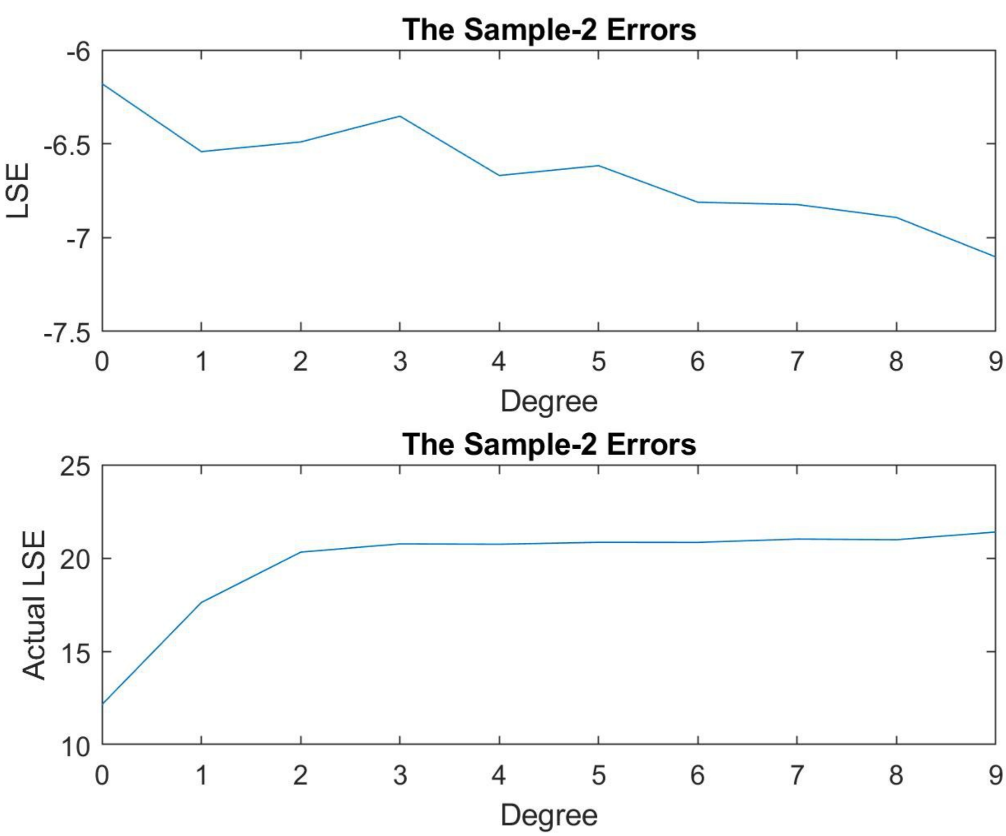
Actual Error Matrix =

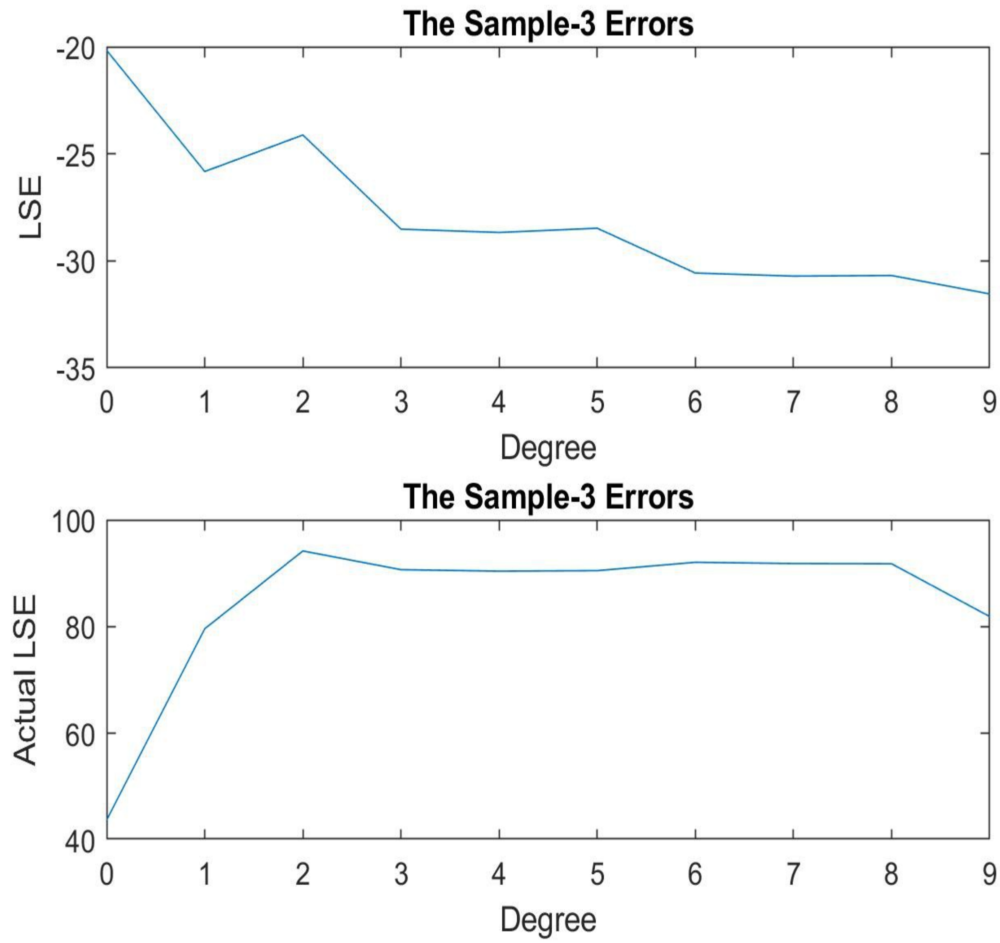
	0	1	2	3	4
Sample1	25.54734	28.33278	30.32989	31.47083	31.21795
Sample2	12.18239	17.63314	20.32667	20.77025	20.75128
Sample3	43.74375	79.60106	94.17995	90.66894	90.39507

	5	6	7	8	9
Sample1	31.30075	31.26461	31.24626	31.27133	31.29788
Sample2	20.85216	20.84751	21.02758	20.99298	21.40343
Sample3	90.48789	92.05664	91.82071	91.77986	81.87985

As shown in the following plots, while the order of polynomial increases, the least square error of each sample falls down (The error of sample 1 starts to reduce after the polynomial degree 3). However, the error between the fitted polynomial and the actual curve increases. It is needed to adjust the complexity of the model to the complexity of the data to minimize the generalization error.


The Sample-1 Errors
LSE
7.8
7.7
7.6
7.5
0 1 2 3 4 5 6 7 8 9
Degree
The Sample-1 Errors
Actual LSE
32
30
28
26
24
0 1 2 3 4 5 6 7 8 9
Degree


The Sample-2 Errors
LSE
-6
-6.5
-7
-7.5
0 1 2 3 4 5 6 7 8 9
Degree
The Sample-2 Errors
Actual LSE
25
20
15
10
0 1 2 3 4 5 6 7 8 9
Degree



The Sample-3 Errors
LSE
-20
-25
-30
-35
0
1
2
3
4
5
6
7
8
9
Degree
The Sample-3 Errors
Actual LSE
100
80
60
40
0
1
2
3
4
5
6
7
8
9
Degree

Appendicies

main.m

```
data= load('C:\Users\Bernisko\Google Drive\Masters\BM 59D\HW2\BM59D_Hw2_Data.mat')

% CLASSIFICATION
xtr_classification = data.xtr_classification;
xval_classification = data.xval_classification;
display('The Plots and The Confusion Matrix of Training and Validation Set')
classification(xtr_classification,1,1,0,0)
classification(xval_classification,1,1,0,0)
display('The Confusion Matrix of Training and Validation Set With Asymmetric Loss')
classification(xtr_classification,1/2,1,0,0)
classification(xval_classification,1/2,1,0,0)
display('The Confusion Matrix of Training and Validation Set With Asymmetric Loss
and Rejection')
classification(xtr_classification,1/2,1,0.2,0)
classification(xval_classification,1/2,1,0.2,0)



%POLYNOMIAL REGRESSION
x_regression= data.x_regression;
x = x_regression(:,1:3);
y = x_regression(:,4);
variances=[0.5,0.3,0.1];
[sample,dimension]=size(x);
degree=9;
for i=1:dimension
    for j=1:degree
        [model,error] =  polynomial_regression(x(:,i),y,variances(i),j);
        models(:,j,i) = model;
        errors(i,j) = error;
    end
    plot(errors(i,:));
    title(sprintf('The Sample-%d Errors',i));
    xlabel('Degree');
    ylabel('LSE');
end
printmat(errors, 'Error Matrix', 'Sample1 Sample2 Sample3', '1 2 3 4 5 6 7 8 9' )
```

classification.m

```
function classification(data_set,loss_1_2,loss_2_1,loss_rej, isDisplay)

data = data_set(:,1);
labels= data_set(:,2);
data_size = size(data,1);
class_1 =  data(labels == 1);
class_2 =  data(labels == -1);

% calculate class 1 parameters
[mean_1,var_1] =calc_class_params(class_1);
prior_1 = size(class_1,1)/data_size;
likelihood_1=calc_likelihood(data,mean_1,var_1);

% calculate class 2 parameters
[mean_2,var_2] =calc_class_params(class_2);
prior_2 = size(class_2,1)/data_size;
likelihood_2=calc_likelihood(data,mean_2,var_2);

%calculate evidence and posteriors
evidence =(likelihood_1.*prior_1 + likelihood_2.*prior_2);
posterior_1 = (likelihood_1.* prior_1)./ evidence;
posterior_2 = (likelihood_2.* prior_2)./ evidence;

%calculate risk
risk_1 = loss_1_2*(1-posterior_1);
risk_2 = loss_2_1*(1-posterior_2);

%calculate discriminant
discriminant_1 = log(likelihood_1) + log(prior_1);
discriminant_2 = log(likelihood_2) + log(prior_2);

%calculate confusion matrix
if loss_rej==0
    class_1_risk = loss_1_2 / (loss_1_2 +loss_2_1)
    tp = size(labels(labels(posterior_1 >class_1_risk ) == 1));
    fp = size(labels(labels(posterior_1 >class_1_risk ) == -1))
    fn = size(labels(labels(posterior_1 < class_1_risk) == 1))
    tn = size(labels(labels(posterior_1 < class_1_risk) == -1))
else
    class_1_lower_risk = loss_rej /loss_2_1
    class_1_upper_risk = (loss_1_2 -loss_rej) / loss_1_2
    tp = size(labels(labels(posterior_1 > class_1_upper_risk) == 1))
    fp = size(labels(labels(posterior_1 > class_1_upper_risk ) == -1))
    fn = size(labels(labels(posterior_1 < class_1_lower_risk) == 1))
    tn = size(labels(labels(posterior_1 < class_1_lower_risk) == -1))
    rej_p = size(labels(labels(posterior_1 < class_1_upper_risk & posterior_1 >
class_1_lower_risk) == 1))
    rej_n = size(labels(labels(posterior_1 < class_1_upper_risk & posterior_1 >
class_1_lower_risk) == -1))
end

if isDisplay==1
    % plot likelihoods
    figure
    subplot(4,1,1)
    plot(data,likelihood_1,data,likelihood_2)
    title('Likelihoods')
```

```
    xlabel('x')
    ylabel('p(x|Ci)')
    likelihood_boundary =find_decision_boundary(data,likelihood_1,likelihood_2)


    % plot posteriors
    subplot(4,1,2)
    plot(data,posterior_1,data,posterior_2)
    title('Posteriors')
    xlabel('x')
    ylabel('p(Ci|x)')
    posterior_boundary = find_decision_boundary(data,posterior_1,posterior_2)

    % plot risk
    subplot(4,1,3)
    plot(data,risk_1,data,risk_2)
    title('Risk')
    xlabel('x')
    ylabel('R(ai|x)')
    risk_boundary =find_decision_boundary(data,risk_1,risk_2)

    % plot discriminants
    subplot(4,1,4)
    plot(data,discriminant_1,data,discriminant_2)
    title('Discriminant')
    xlabel('x')
    ylabel('gi(x)')
    discriminant_boundary
=find_decision_boundary(data,discriminant_1,discriminant_2)

end
end
```

calc_class_params.m

```
function [mean var] = calc_class_params(class)
class_size = length(class);
sum=0;
for i=1:length(class)
    sum=sum+class(i);
end
mean = sum/class_size;
sum=0;
for i=1:length(class)
    sum=sum+ (class(i)-mean)^2;
end
var=sum/length(class);
end
```

calc_likelihood.m

```
function [likelihood] = calc_likelihood(class,mean,var)
likelihood = 1/sqrt(2*pi*var) * exp(-0.5*(class - mean).*(class - mean)/var);
end
```

find_decision_boundary.m

```
function idx = find_decision_boundary(class, class_1,class_2)
A = abs(class_1-class_2);
idx = class(find(A == min(A)));
end
```

polynomial_regression.m

```
function [model, lse, actual_curve_lse] = polynomial_regression(x,y,var,degree)

sample = length(x);
actual_curve = x.^3 - x + 1
for i=1:degree+1
    for j=1:degree+1
        for k=1:sample
            A(i,j,k)=x(k).^(i+j-2);
        end
    end
end

for j=1:degree+1
    for k=1:sample
        D(k,j)=x(k).^(j-1);
    end
end

for j=1:degree+1
    for k=1:sample
        D_T(j,k)=x(k).^(j-1);
    end
end

w = inv(D_T*D)*D_T*y;

for j=1:sample
    sum=0.0;
    for k=1:degree+1
        sum = sum + w(k)  * x(j)^k;
    end
    model(j,:)  = sum;
end

lse = sample * log(sqrt(2*pi*(var^2))) + (1/( 2*(var^2))) * mean((y-model).^2);
actual_curve_lse = sample * log(sqrt(2*pi*(var^2))) + (1/( 2*(var^2))) *
mean((actual_curve-model).^2);

end
```