EECE-5644

INTRO TO MACHINE LEARNING AND PATTERN RECOGNITION

ASSIGNMENT 2

SAI PRASASTH KOUNDINYA GANDRAKOTA
NUID: 002772719

QUESTION 1

PART 1

We use theoretical optimal classification rule to obtain

$$\frac{P(X|L=1)}{P(X|L=0)} \underset{(D=0)}{\overset{(D=1)}{\gtrless}} \frac{0.6 \times (\lambda_{10} - \lambda_{00})}{0.4 \times (\lambda_{01} - \lambda_{11})}$$

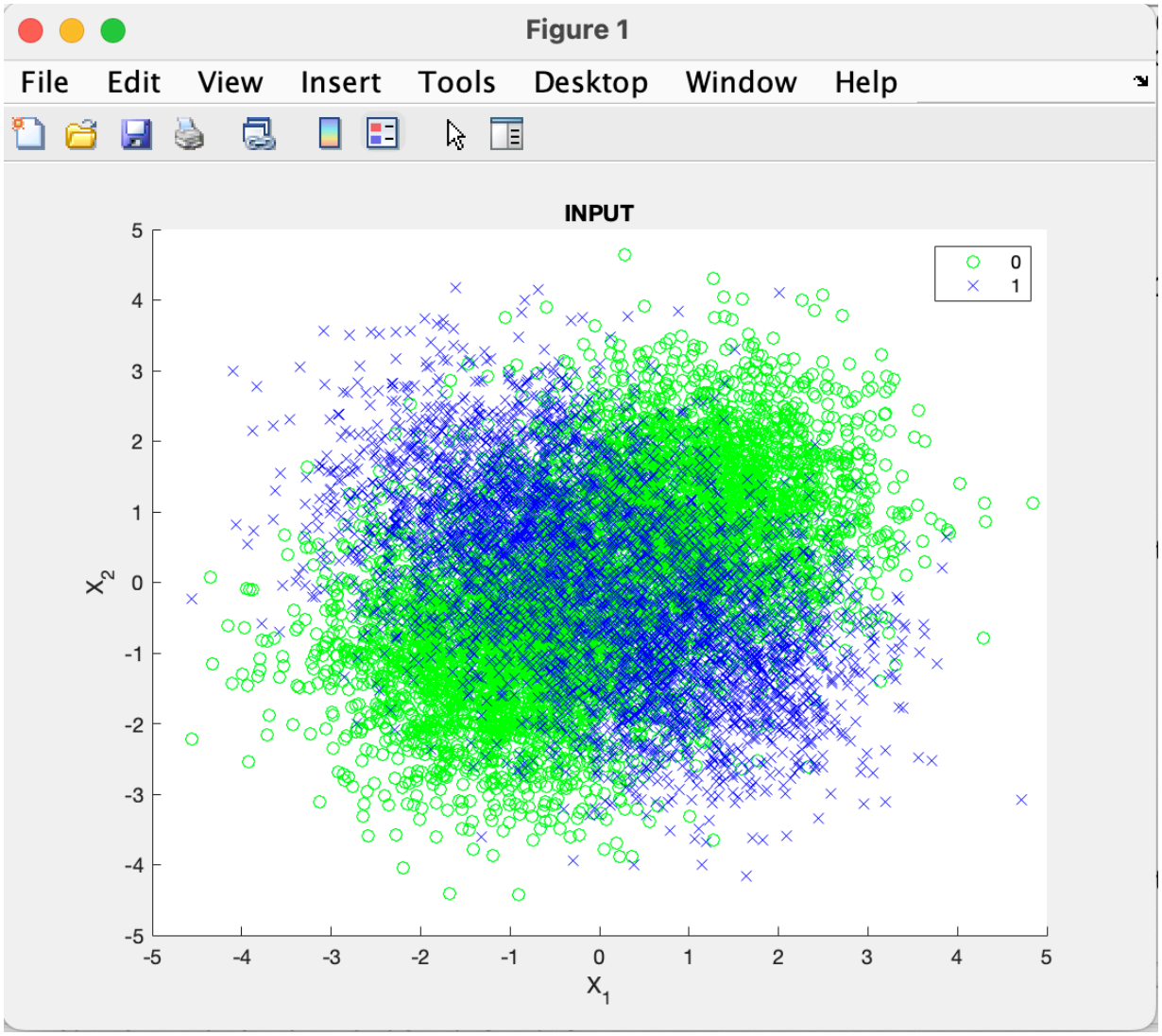
$\lambda_{10} = \lambda_{01} = 1$ and $\lambda_{00} = \lambda_{11} = 0$ as it is 1 point penalty for incorrect decisions.

$$\frac{P(X|L=1)}{P(X|L=0)} \underset{<}{\overset{>}{}} 1.5$$

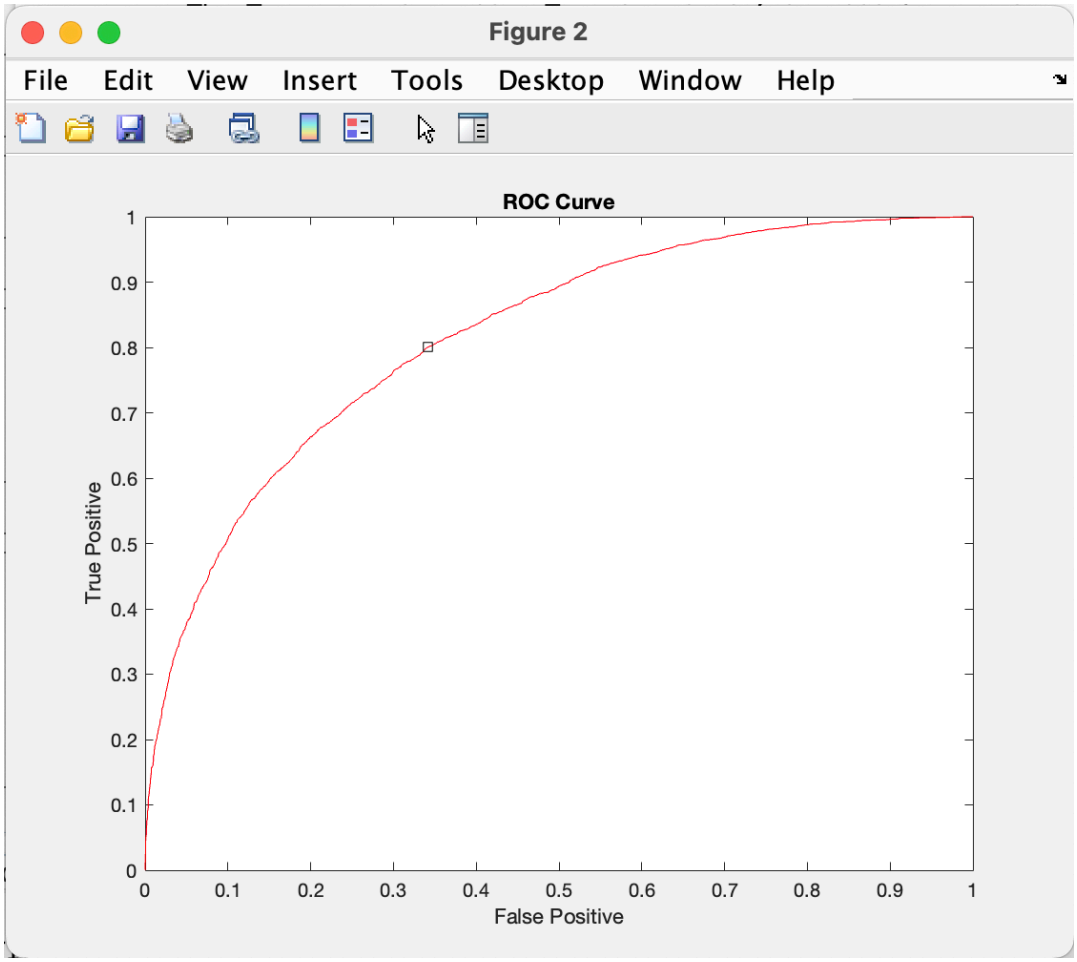
The above classifier was trained and validated on ~~the~~ the $D^{10k}_{validate}$ dataset and the minimum error was calculated along with the ROC curve being plotted.

PART 1

INPUT DATA



ROC CURVE



OUTPUT

Command Window

```
Ideal Threshold =
    1.5000

Ideal Minimum Error =
    0.3005

Practical Threshold =
    0.8027

Practical Minimum Error =
    0.2561
```

PART 2

(A) LOGISTIC LINEAR REGRESSION

In this method we take the input vector $X = [X_1\ X_2]$ and create a basis vector $[1\ X^T]^T$ to match the $\theta = [\theta_0\ \theta_1\ \theta_2]^T$ vector.

$$z = \theta^T b(x)$$

Prediction function $h(x,\theta) = \dfrac{1}{1 - e^{-\theta^T b(x)}}$

which is a sigmoid function where $h(x,\theta) = 1$ if $h(x) \geq 0.5$ and $h(x,\theta) = 0$ if $h(x) < 0.5$

The parameter θ_{ML} is the same as weight vector w_{ML}

$$w_{ML} = \min_w -\log P(Y|X,w)$$

where $P(Y|X,w) = \begin{cases} h(x,w) & ; \text{if } Y = 1 \\ 1 - h(x,w) & ; \text{if } Y = 0 \end{cases}$

Using the above equations,

$$w_{ML} = \operatorname{argmin}_w \; -[Y \log h + (1-Y)\log(1-h)]$$

PART 2

(B) QUADRATIC LINEAR REGRESSION

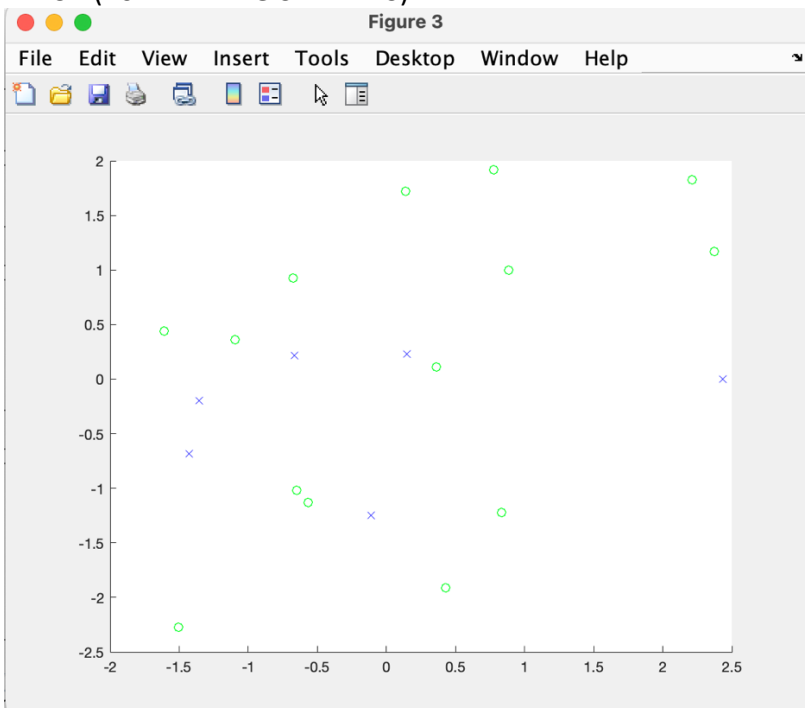
Basis vector, $b(X) = [1 \; X_1 \; X_2 \; X_1^2 \; X_1 X_2 \; X_2^2]$

The rest of the procedure is similar to logistic linear regression.

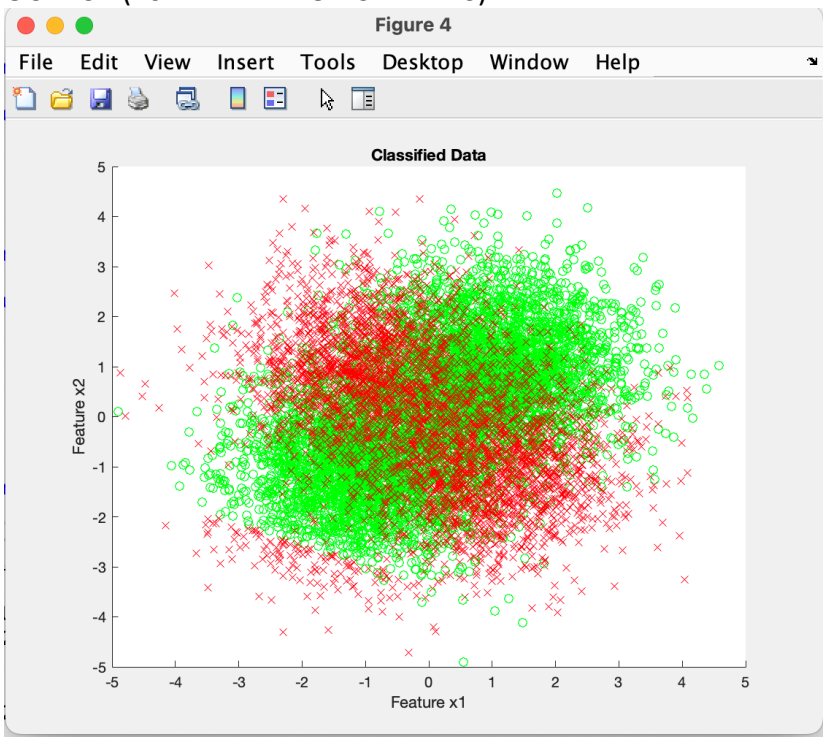
These above methods were trained on the D_{train}^{20}, D_{train}^{200} and D_{train}^{2000} data sets and then applied to the $D_{validate}^{10K}$ dataset.

PART 2 (A)
LOGISTIC LINEAR REGRESSION

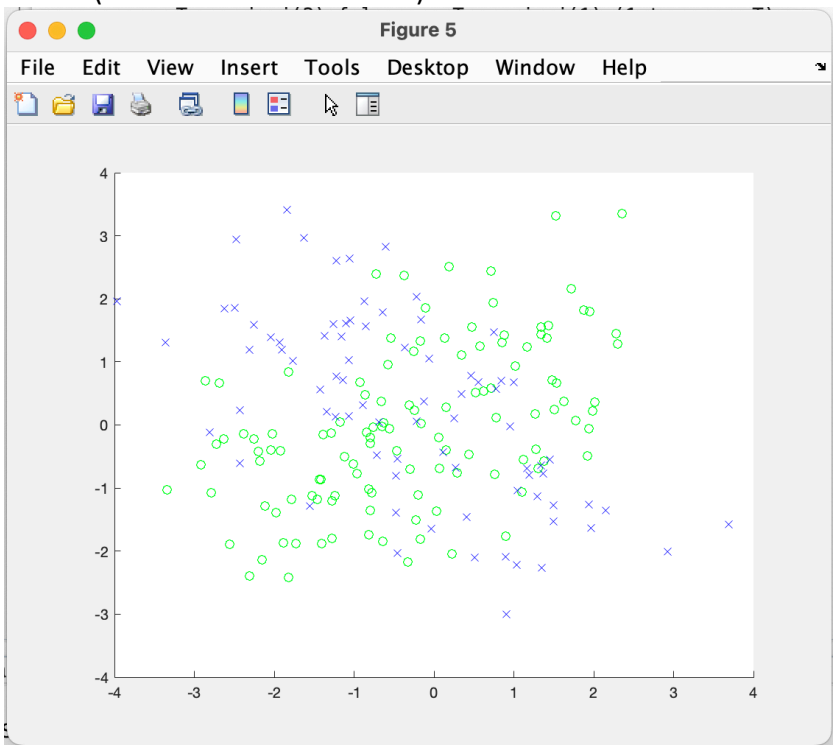
INPUT (20 TRAINING SAMPLES)



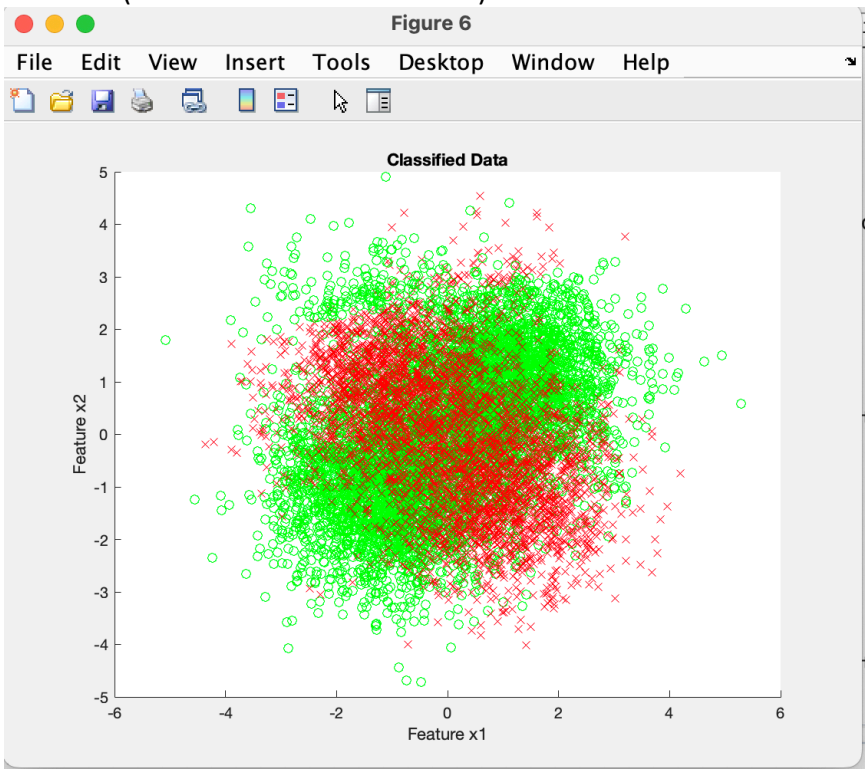
OUTPUT (10K VALIDATION SAMPLES)



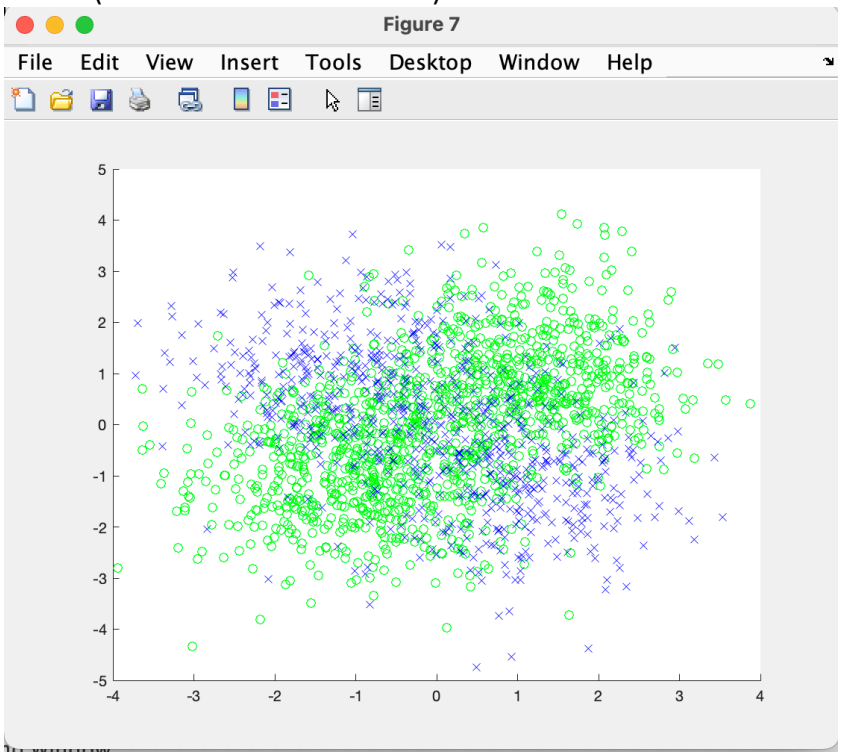
INPUT (200 TRAINING SAMPLES)



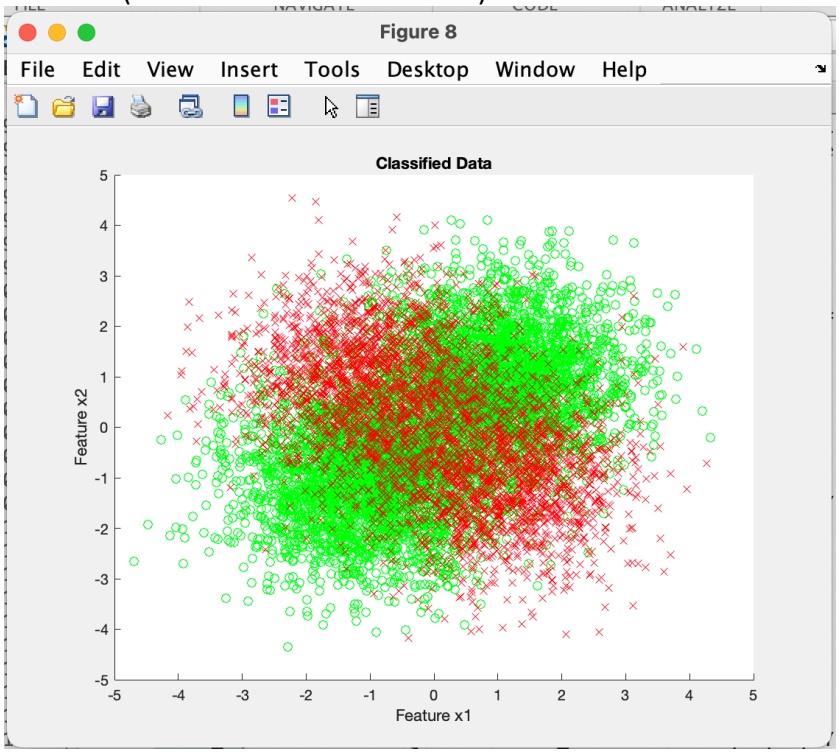
OUTPUT (10K VALIDATION SAMPLES)



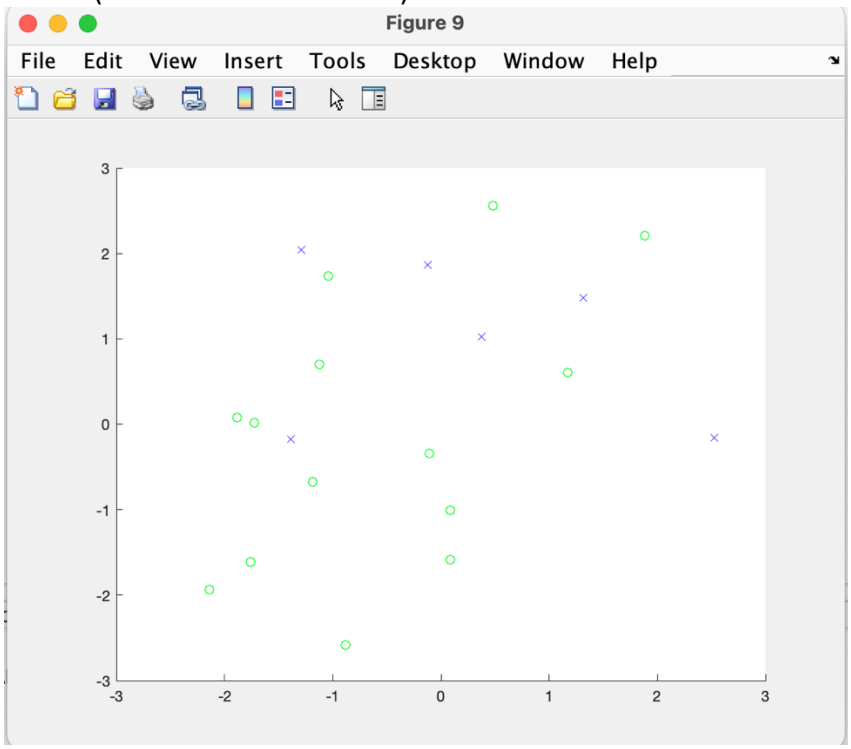
INPUT (2000 TRAINING SAMPLES)



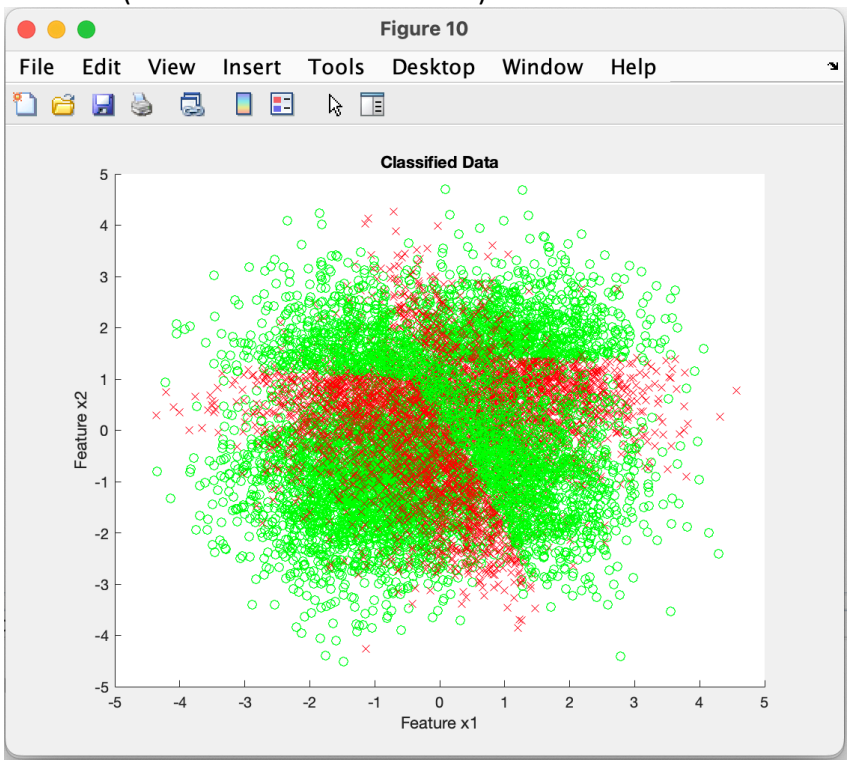
OUTPUT (10K VALIDATION SAMPLES)



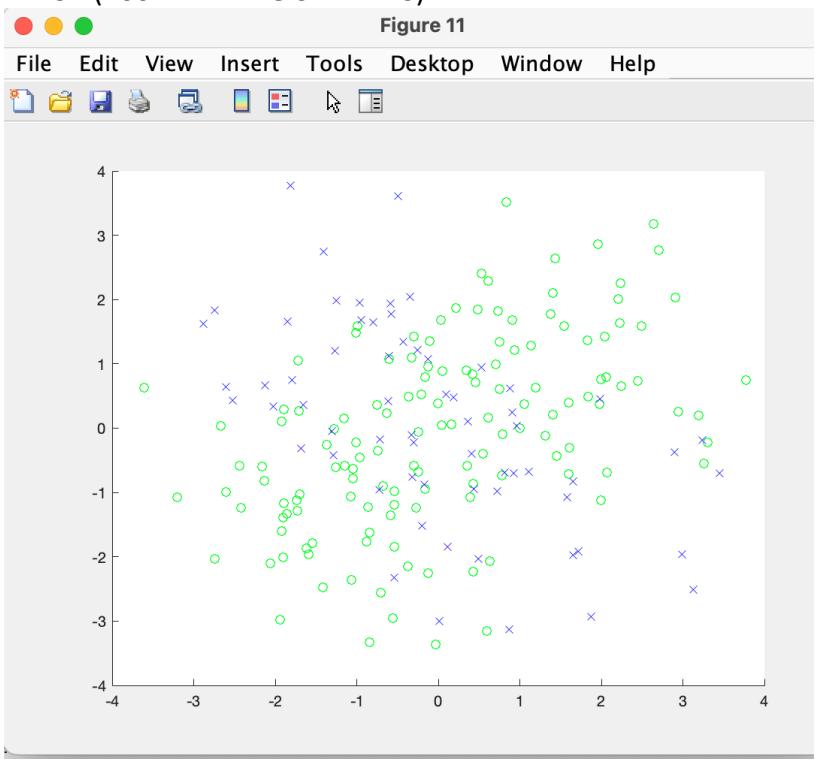
LOGISTIC QUADRATIC REGRESSION
INPUT (20 TRAINING SAMPLES)



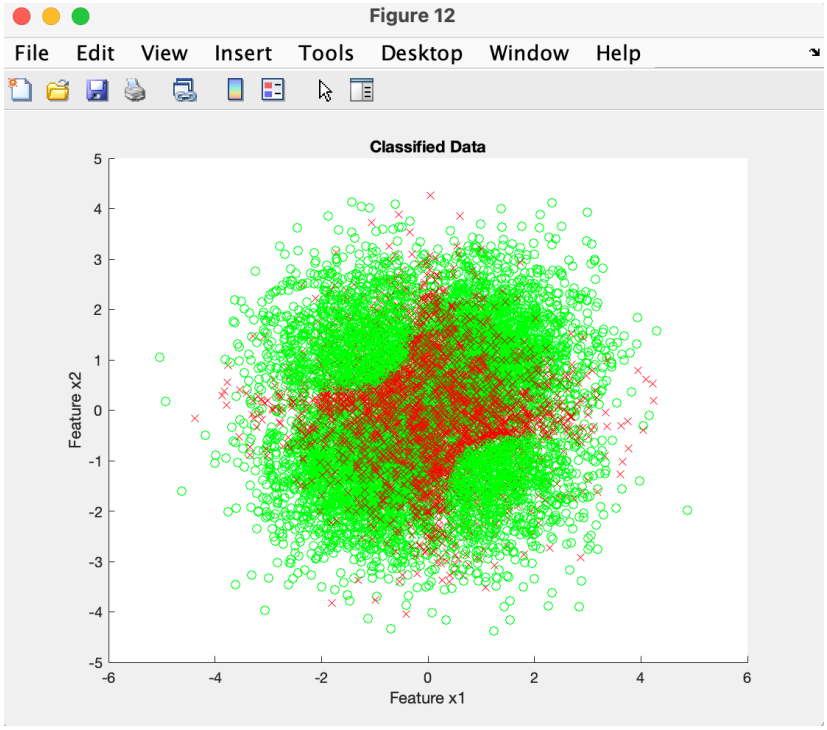
OUTPUT (10K VALIDATION SAMPLES)



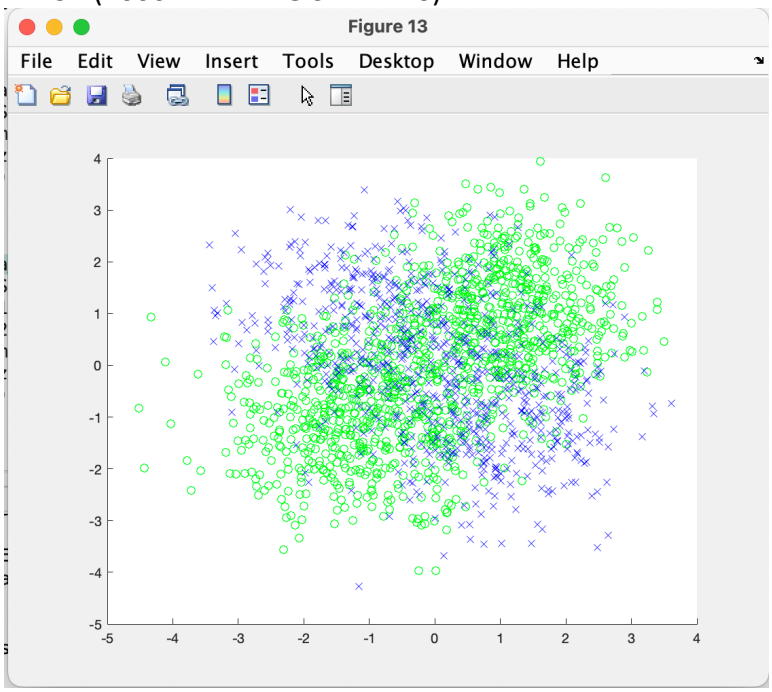
INPUT (200 TRAINING SAMPLES)



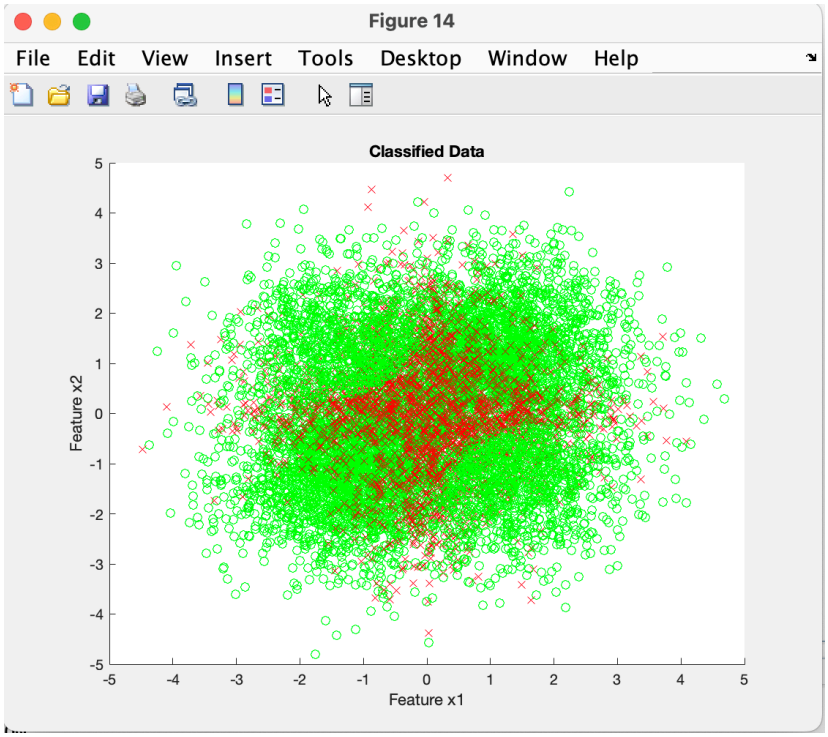
OUTPUT (10K VALIDATION SAMPLES)



INPUT (2000 TRAINING SAMPLES)



OUTPUT (10K VALIDATION SAMPLES)



ERROR

```
LOGISTIC LINEAR REGRESSION
Error for 20 training samples =
   40.8200

Error for 200 training samples =
   39.0200

Error for 2000 training samples =
   39.8300


LOGISTIC QUADRATIC REGRESSION
Error for 20 training samples =
   40.0300

Error for 200 training samples =
   26.3800

Error for 2000 training samples =
   25.0200
```

DISCUSSION

- Logistic Quadratic Model has a better learning rate than Logistic Linear Model
- The boundary formed is elliptical, as the classes are not linearly separable
- As the training samples increase, the accuracy of prediction increases

QUESTION 2

ML Estimator

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmin}} \; \frac{-1}{N} \sum_{i=1}^{N} \frac{\log(P(x_i, y_i|\theta))}{P(y_i|x_i,\theta)P(x_i|\theta)}$$

if we assume x is independent of θ

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmin}} \; \frac{-1}{N} \sum_{i=1}^{N} \log g(y_i|f(x_i,\theta), \varepsilon)$$

$$= \underset{\theta}{\operatorname{argmin}} \; \frac{-1}{2N\sigma^2} \sum_{i=1}^{N} (y - f(x_i,\theta))^T (y - f(x_i - \theta))$$

$$\varepsilon = \sigma^2 . I \quad ; f(x,\theta) = Z^T . \theta$$

$$\theta_{ML} = \underset{\theta}{\operatorname{argmin}} \; \frac{1}{N} \sum_{i=1}^{N} (y_i - z_i^T\theta)^T (y_i - z_i^T\theta)$$

Partially differentiating w.r.t θ^T and equating to 0, we get

$$\theta = \left(\frac{1}{N}\sum_{i=1}^{N} y_i z_i\right)\left(\frac{1}{N}\sum_{i=1}^{N} z_i z_i^T\right)^{-1}$$

MAP Estimator

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmin}} \; -\log(P(\theta|X,Y))$$

From Bayes theorem we know,

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmin}} \; -\log \frac{P(Y|X,\theta)P(\theta)}{P(Y|X)}$$

Since $P(Y|X)$ and θ are independent,

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmin}} \; -\log(P(Y|X,\theta)) - \log(P(\theta))$$

$$P(\theta) = \mathcal{N}(0, \gamma I)$$

Diff. w.r.t θ on both sides.

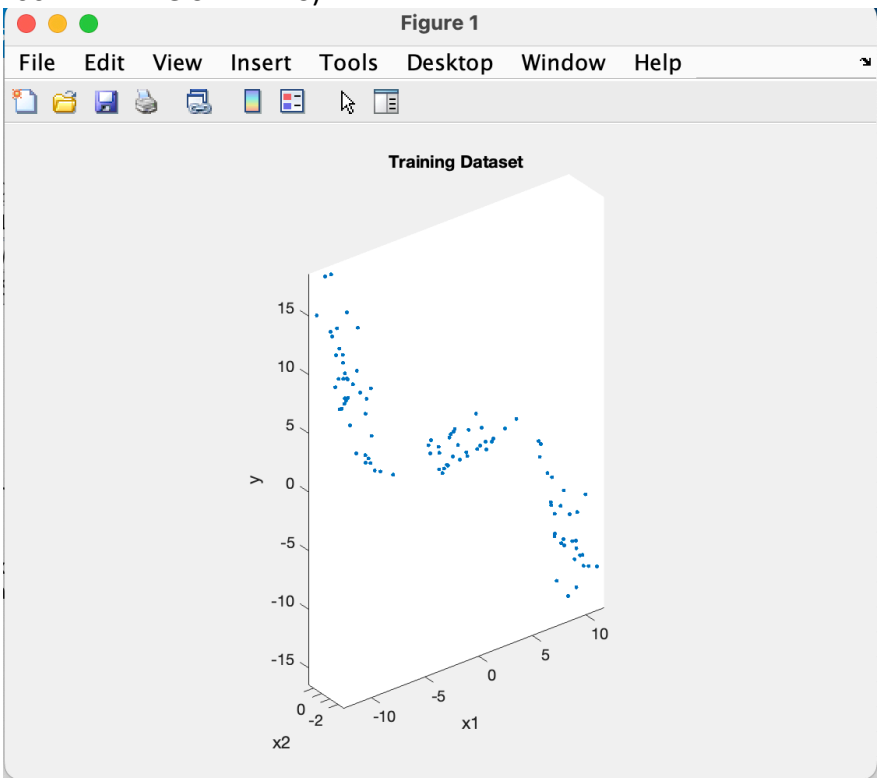
$$\frac{1}{2N\sigma^2} \sum_{i=1}^{N} [2(y_i - z_i\theta^T\theta)z_i] - \frac{1}{\gamma}\theta = 0$$

$$\theta = \left(\frac{1}{N}\sum_{i=1}^{N} z_i^T z_i + \frac{\sigma^2}{\gamma} I\right)^{-1} \left(\frac{1}{N}\sum_{i=1}^{N} y_i z_i\right)$$

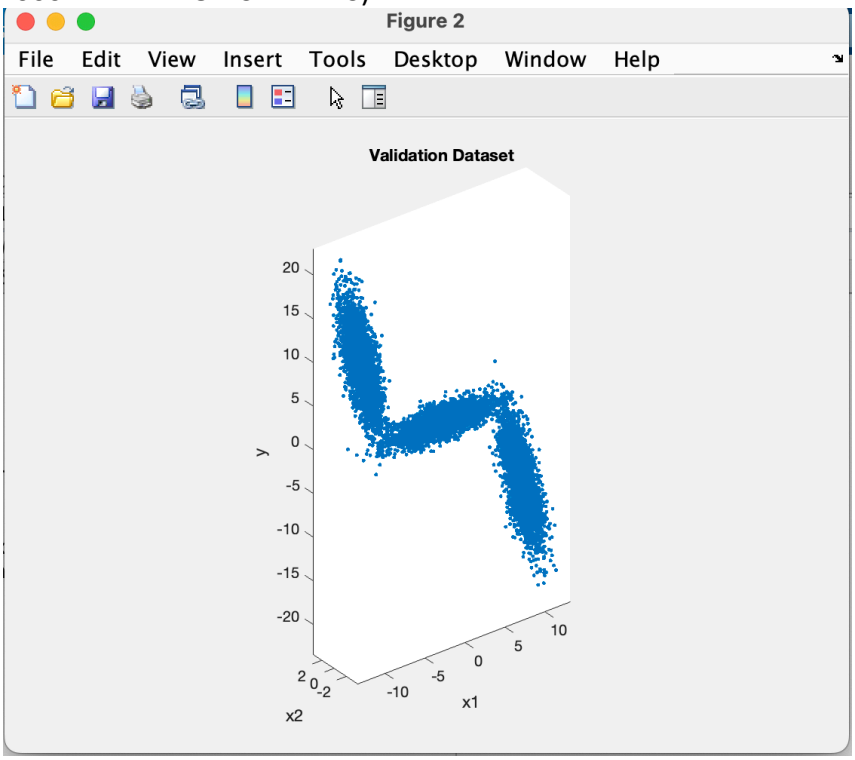
$$\text{Mean Squared Error} = \frac{1}{N}\sum_{i=1}^{N} (Y_{est} - Y_{true})^2$$

Both models were trained using 100 samples and then validated using 10000 samples.

INPUT(100 TRAINING SAMPLES)



INPUT(1000 VALIDATION SAMPLES)



ML ESTIMATOR

Command Window

```
ML ESTIMATOR
Mean Squared Error =
    4.6164
```

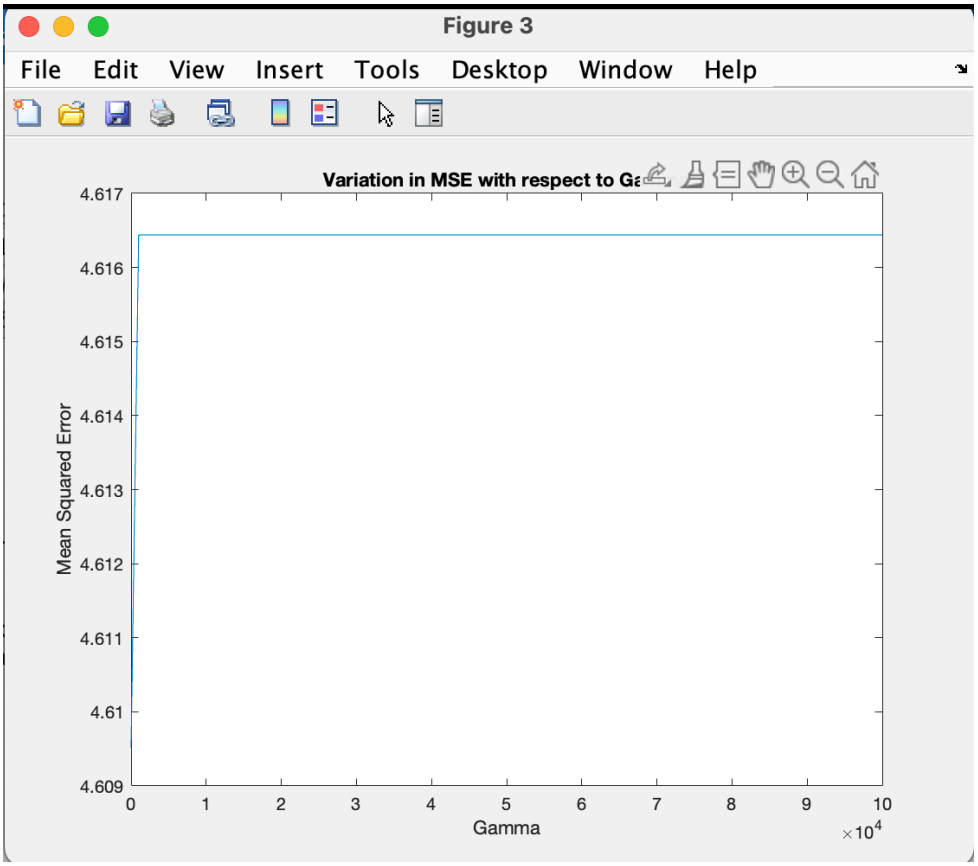
MAP ESTIMATOR

```
MAP ESTIMATOR
Minimum Gamma =
    0.0100

Mean Squared Error =
    4.6095

Maximum Gamma =
      100000

Mean Squared Error =
    4.6164
```



DISCUSSION

- MSE value increases monotonically as gamma increases
- Prior probability helps in improving performance of the model
- When gamma tends to infinity, both the ML and MAP estimators are approximately equal

QUESTION 3

To Find: $[x, y]^T$ with highest probability given prior distribution and range from each reference coordinate.

$$\begin{bmatrix} x_{MAP} \\ y_{MAP} \end{bmatrix} = \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{\operatorname{argmax}} \; P\left(\begin{bmatrix} x \\ y \end{bmatrix} | (r_1 \dots r_k)\right)$$

$$= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{\operatorname{argmax}} \; \log\left((2\pi\sigma_x\sigma_y)^{-1}\right) + \log\left(e^{-1/2 * [x\,y]\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix}}\right) + \sum_{i=1}^{k} \log P\left(\begin{bmatrix} x \\ y \end{bmatrix} | r_i\right)$$

$$= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{\operatorname{argmax}} \; \frac{-1}{2} [x\;y]\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{k} - \frac{(r_i - d_i)^2}{2\sigma_i^2}$$

where $d_i = \left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right\|$

$$= \underset{\begin{bmatrix} x \\ y \end{bmatrix}}{\operatorname{argmin}} \; [x\;y]\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix} + \sum_{i=1}^{k} \frac{(r_i - d_i)^2}{\sigma_i^2}$$

OUTPUT

K = 1

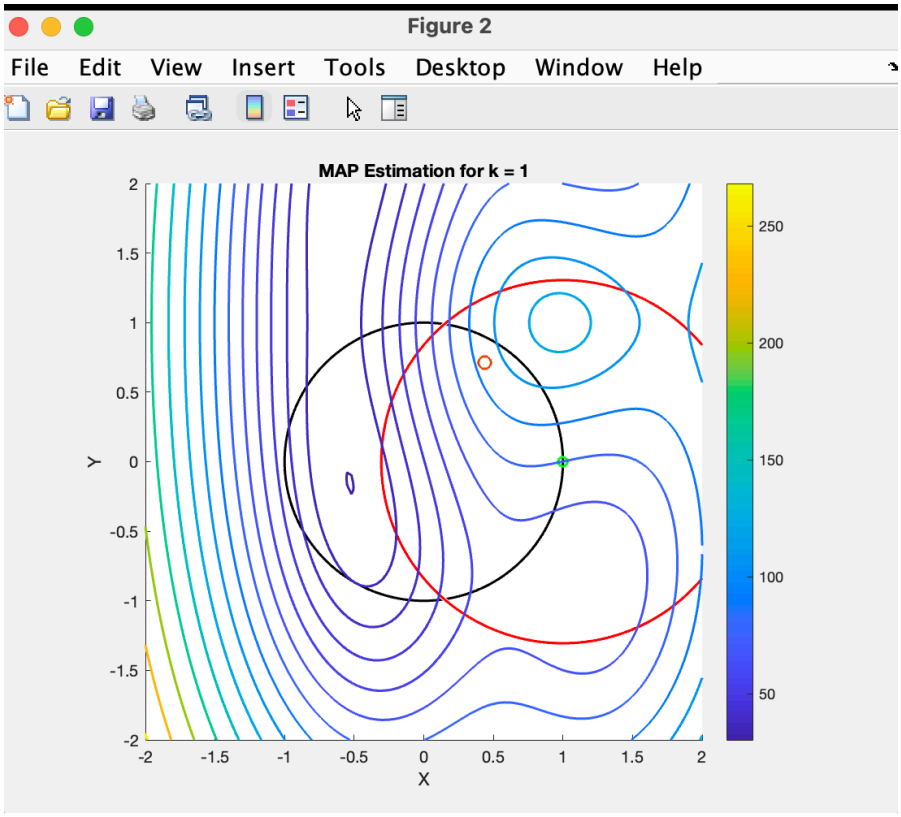


Figure 2
File Edit View Insert Tools Desktop Window Help
MAP Estimation for k = 1
X
Y

K = 2

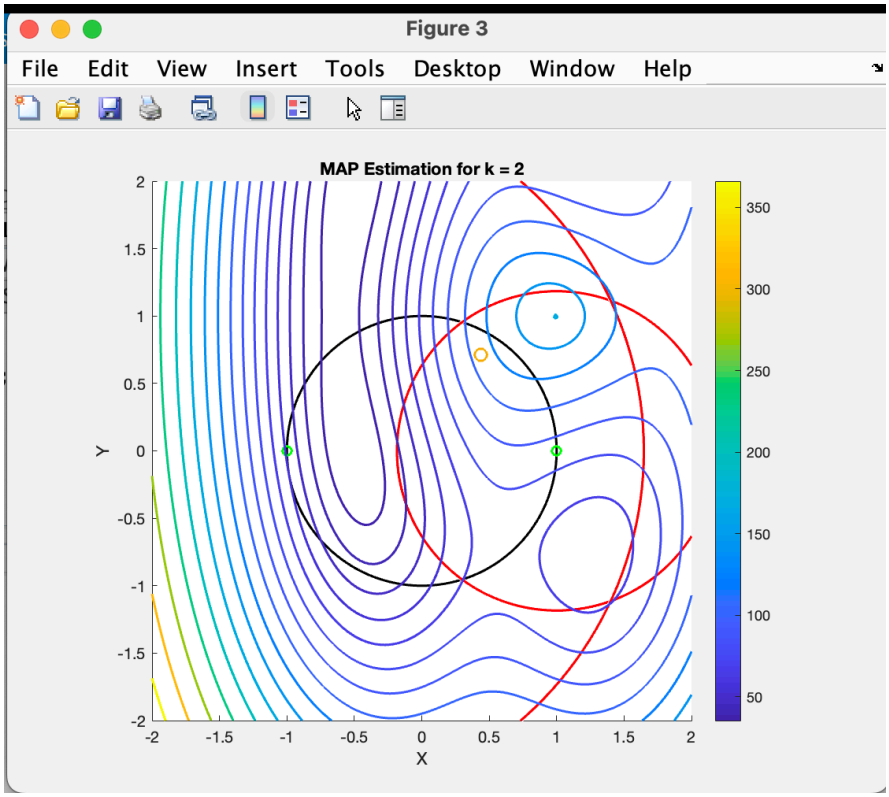


Figure 3
File Edit View Insert Tools Desktop Window Help
MAP Estimation for k = 2
X
Y

K = 3

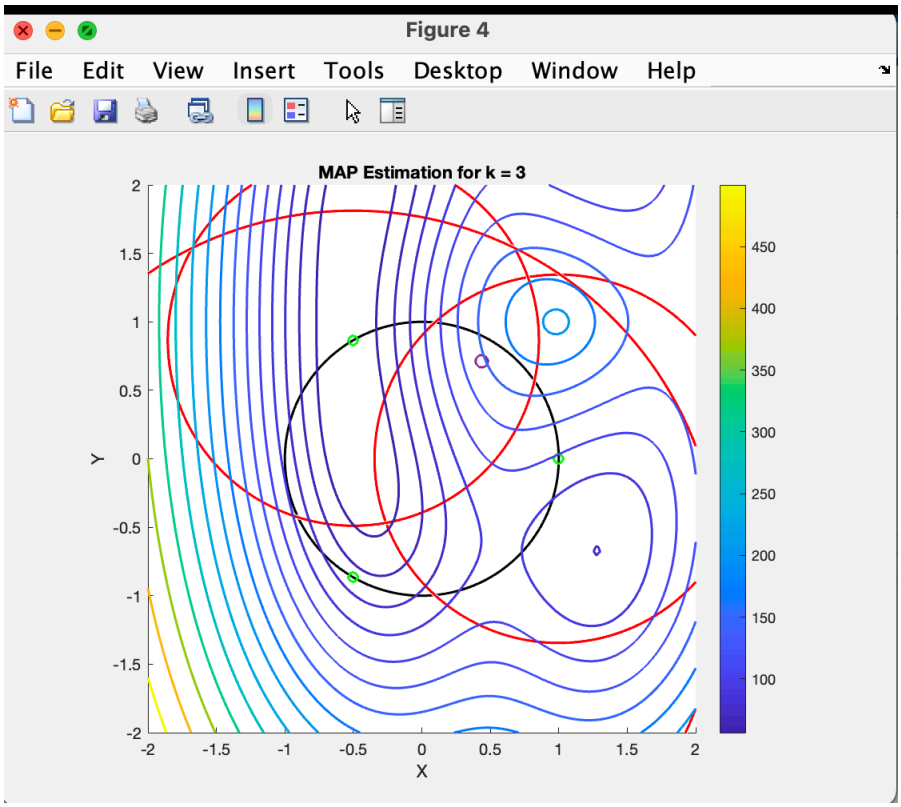


Figure 4
File Edit View Insert Tools Desktop Window Help
MAP Estimation for k = 3
X
Y

K = 4

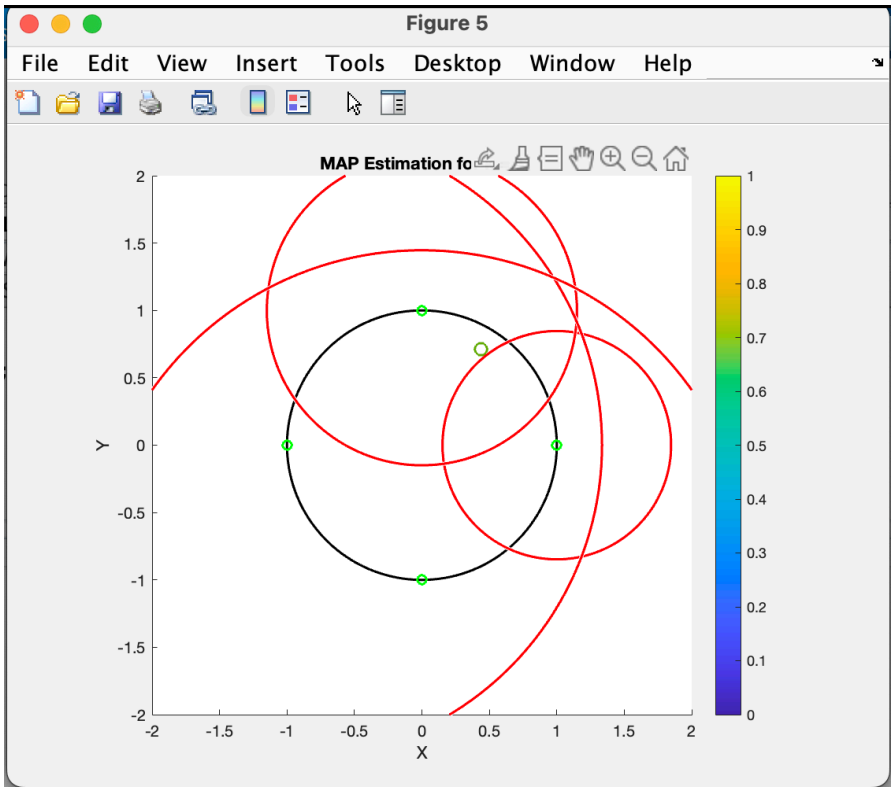


Figure 5
File Edit View Insert Tools Desktop Window Help
MAP Estimation fo
X
Y

DISCUSSION

- Estimator accuracy increases as K increases
- Contour graphs depict the estimator's confidence by reducing the region of locations with high likelihood.
- The accuracy of the estimator can be gauged from the contour graph by measuring the distance between the actual location and the point with the smallest contour, typically located around the middle of the innermost contour.
- With an increase in K, the estimator's confidence increases.
- It is challenging to discern the reduction in the area of high probability locations for small K values, but the phenomenon becomes more evident by tracking a single contour level as K rises.

QUESTION 4

In order to find minimum risk we have to minimize expected loss.

Loss function

$L(ij) = \lambda_S$, if $i \neq j$

$L(ij) = 0$, if $i = j$

$L(ic+1) = \lambda_R$

Expected loss,

$$E(L_i|X) = \sum_{j=1}^{c} P(w_j|X) L(ij) + \left(1 - \sum_{j=1}^{c} P(w_j|X)\right) L(ic+1)$$

$$= \sum_{j=1}^{c} \left(L(ij) - L(ic+1)\right) \cdot P(w_j|X) + \lambda_R$$

In order to minimize $E(L_i|X)$ we consider 3 cases:

→ $\lambda_R = 0$: choose i such that $P(w_i|x)$ is greatest.

→ $\lambda_R > 0$: set threshold T such that any $P(w_i|x) < 1 - T$ and choose i such that $P(w_i|X)$ is greatest with the above condition.

→ $\lambda_R > \lambda_S$: set small threshold; and choose i such that $P(w_i|X)$ is maximum.

QUESTION 5

Given $Z = [Z_1, Z_2, \cdots N]^T$ where

$$Z_K = \begin{cases} 1 & ; \text{if state} = K \\ 0 & ; \text{otherwise} \end{cases}$$

Estimate $\theta = [\theta_1, \theta_2 \cdots, \theta_K]^T$

$$P(Z_K = 1) = \theta_K \quad \text{for } K = 1 \text{ to } n$$

$$Z_n \sim \text{Categorical}(\theta) \quad \text{for } n = 1 \text{ to } N$$

There are 2 conditions on θ since it's a probability,

→ $\theta >= 0$

→ $\sum \theta_i = 1$

PART A

ML Estimator

- Lagrangian optimization method is used to identify θ
- Likelihood is maximized

$$\theta_{ML} = \arg\max_{\theta} \left[\log P(Y|\theta) - \lambda(\theta^T \cdot 1 - 1) \right]$$

$$L(\theta, \lambda) = \arg\min_{\theta} -\frac{1}{N} \sum_{i=1}^{N} \log P(Z_i|\theta) + \lambda(\theta^T 1 - 1)$$

Diff. $L(\theta, \lambda)$ w.r.t to all θ_K and equating to 0.

$$\lambda(\theta_1) = \frac{N_1}{N} \quad ; \quad \lambda - \frac{N_1}{N(\theta_1)} = 0$$

$$\lambda(\theta_K) = \frac{N_K}{N} \quad ; \quad \lambda - \frac{N_K}{N(\theta_K)} = 0$$

$$\lambda(\theta_1 + \theta_2 + \cdots + \theta_K) = \frac{(N_1 + N_2 + \cdots N_K)}{N} = \frac{N}{N}$$

$$\lambda = 1$$

PART B

MAP Estimator

$$\theta_{MAP} = \arg\max_{\theta} \log(P(\theta|Y))$$

$$= \arg\max_{\theta} \log\left(\frac{P(Y|\theta) \cdot P(\theta)}{P(Y)}\right)$$

$P(Y) \rightarrow$ constant

$$\theta_{MAP} = \arg\max_{\theta} \log(P(Y|\theta)) + \log(P(\theta))$$

$$= \arg\max_{\theta} \left[\frac{1}{N}\sum_{i=1}^{N} \log(P(Z_i|\theta)) + \log \prod_{i=1}^{K} (\theta_K)^{\alpha_K - 1} - \ln B(\alpha)\right.$$

$B(\alpha)$ is a constant w.r.t θ

$$L(\theta, \lambda) = \arg\max_{\theta} \frac{1}{N}\sum_{i=1}^{N} \log(\theta_{Z_i}) + (\alpha_K - 1)\sum_{i=1}^{K} \log(\theta_K) - \lambda(\theta^T 1)$$

Diff. $L(\theta, \lambda)$ w.r.t θ_K values and equate to 0

$$\lambda + \frac{(\alpha_1 - 1)}{\theta_1} - \frac{N_1}{N(\theta_1)} = 0 \quad ; \quad \lambda(\theta_1) = \frac{N_1}{N} + (\alpha_1 - 1)$$

$$\lambda + \frac{\alpha_K - 1}{\theta_K} - \frac{N_K}{N(\theta_K)} = 0 \quad ; \quad \lambda(\theta_K) = \frac{N_K}{N}(\alpha_K - 1)$$

$$\lambda = 1 + \alpha^T \cdot 1 - K$$

$$\theta_K = \frac{\frac{N_K}{N} + (\alpha_K - 1)}{1 + \alpha^T \cdot 1 - K}$$

APPENDIX (MATLAB CODE)

QUESTION 1

```
%ML HW 2
%Q1
clc; clear all; close all;
rng(10);


%PART 1
n = 2;
priori = [0.6,0.4];
mean_ij(1,:) = [-1,-1];
mean_ij(2,:) = [1,1];
mean_ij(3,:) = [-1,1];
mean_ij(4,:) = [1,-1];
cov_m = eye(2);
w = [0.5,0.5];

%Generating Samples with true class labels
DV_10K = generate_samples (mean_ij,cov_m,priori,n,w,10000);

figure
scatter(DV_10K(find(DV_10K(:,3)==0),1),DV_10K(find(DV_10K(:,3)==0),2),'o',
'g')
hold on
scatter(DV_10K(find(DV_10K(:,3)==1),1),DV_10K(find(DV_10K(:,3)==1),2),'X',
'b')
xlabel('X_1')
ylabel('X_2')
legend('0','1')
title('INPUT')


[disc,sort_disc,tau] = calc_threshold(DV_10K,mean_ij,cov_m,w);

[dec,true_pos,false_pos,error,min_error,min_error_ind] =
classify_data(DV_10K,tau,disc,priori);

[tau_T,dec_T,true_pos_T,false_pos_T,error_T] =
calc_theoretical_tau(priori,DV_10K,disc);

disp('Ideal Threshold = ');
disp(exp(tau_T));
disp('Ideal Minimum Error = ');
disp(error_T);

disp('Practical Threshold = ');
disp(exp(tau(min_error_ind)));
disp('Practical Minimum Error = ');
disp(min_error);

figure
```

```
plot(false_pos, true_pos, 'r')
hold on
plot(false_pos(min_error_ind), true_pos(min_error_ind), 'square', 'color',
'k')
xlabel('False Positive')
ylabel('True Positive')
title('ROC Curve')

%PART 2A
%Logistic Linear
DT_20 = generate_samples(mean_ij,cov_m,priori,n,w,20);
DT_200 = generate_samples(mean_ij,cov_m,priori,n,w,200);
DT_2K = generate_samples(mean_ij,cov_m,priori,n,w,2000);

[theta_ML,z] = cost_optimization_linear(DT_20,n);
plot_data(DT_20);
DV_10K = generate_samples(mean_ij,cov_m,priori,n,w,10000);
[theta_test,z_test] = cost_optimization_linear(DV_10K,n);
[error_20,dec_test_20] = logistic(theta_ML,z_test,DV_10K);
plot_class_data(DV_10K,dec_test_20');

[theta_ML,z] = cost_optimization_linear(DT_200,n);
plot_data(DT_200);
DV_10K = generate_samples(mean_ij,cov_m,priori,n,w,10000);
[theta_test,z_test] = cost_optimization_linear(DV_10K,n);
[error_200,dec_test_200] = logistic(theta_ML,z_test,DV_10K);
plot_class_data(DV_10K,dec_test_200');

[theta_ML,z] = cost_optimization_linear(DT_2K,n);
plot_data(DT_2K);
DV_10K = generate_samples(mean_ij,cov_m,priori,n,w,10000);
[theta_test,z_test] = cost_optimization_linear(DV_10K,n);
[error_2K,dec_test_2K] = logistic(theta_ML,z_test,DV_10K);
plot_class_data(DV_10K,dec_test_2K');

disp("LOGISTIC LINEAR REGRESSION");
disp('Error for 20 training samples = ');
disp(error_20);
disp('Error for 200 training samples = ');
disp(error_200);
disp('Error for 2000 training samples = ');
disp(error_2K);


%PART 2B
%Logistic Quadratic
DT_20 = generate_samples(mean_ij,cov_m,priori,n,w,20);
DT_200 = generate_samples(mean_ij,cov_m,priori,n,w,200);
DT_2K = generate_samples(mean_ij,cov_m,priori,n,w,2000);

[theta_MQ,z] = cost_optimization_quadratic(DT_20,n);
plot_data(DT_20);
DV_10K = generate_samples(mean_ij,cov_m,priori,n,w,10000);
[theta_test,z_test] = cost_optimization_quadratic(DV_10K,n);
[error_20,dec_test_20] = logistic(theta_MQ,z_test,DV_10K);
```

```
plot_class_data(DV_10K,dec_test_20');

[theta_MQ,z] = cost_optimization_quadratic(DT_200,n);
plot_data(DT_200);
DV_10K = generate_samples(mean_ij,cov_m,priori,n,w,10000);
[theta_test,z_test] = cost_optimization_quadratic(DV_10K,n);
[error_200,dec_test_200] = logistic(theta_MQ,z_test,DV_10K);
plot_class_data(DV_10K,dec_test_200');

[theta_MQ,z] = cost_optimization_quadratic(DT_2K,n);
plot_data(DT_2K);
DV_10K = generate_samples(mean_ij,cov_m,priori,n,w,10000);
[theta_test,z_test] = cost_optimization_quadratic(DV_10K,n);
[error_2K,dec_test_2K] = logistic(theta_MQ,z_test,DV_10K);
plot_class_data(DV_10K,dec_test_2K');

disp("LOGISTIC QUADRATIC REGRESSION");
disp('Error for 20 training samples = ');
disp(error_20);
disp('Error for 200 training samples = ');
disp(error_200);
disp('Error for 2000 training samples = ');
disp(error_2K);


function DS = generate_samples(mean_ij,cov_m,priori,n,w,set_size)
    DS = zeros(set_size,n+1);
    CL = (rand(set_size, 1) >= priori(1));
    CL = double(CL);
    for i = 1:set_size
        if CL(i) == 0
            if rand(1,1) >= w(1)
                sample(i,:) = mvnrnd(mean_ij(1,:),cov_m);
            else
                sample(i,:) = mvnrnd(mean_ij(2,:),cov_m);
            end
        elseif CL(i) == 1
            if rand(1,1) >= w(1)
                sample(i,:) = mvnrnd(mean_ij(3,:),cov_m);
            else
                sample(i,:) = mvnrnd(mean_ij(4,:),cov_m);
            end
        end
    end
    DS = [sample,CL];
end

function plot_data(DS)
    figure
    scatter(DS(find(DS(:,3)==0),1),DS(find(DS(:,3)==0),2),'o', 'g')
    hold on
    scatter(DS(find(DS(:,3)==1),1),DS(find(DS(:,3)==1),2),'X', 'b')
end

function plot_class_data(DS,dec)
```

```
    sample1 = DS(:,1);
    sample2 = DS(:,2);
    label = DS(:,3);
    figure;
    scatter(sample1(label==0 & dec == 0),sample2(label==0 & dec ==
0),'o','g');
    hold on;
    scatter(sample1(label==0 & dec == 1),sample2(label==0 & dec ==
1),'X','r');
    scatter(sample1(label==1 & dec == 0),sample2(label==1 & dec ==
0),'X','r');
    scatter(sample1(label==1 & dec == 1),sample2(label==1 & dec ==
1),'o','g');
    xlabel("Feature x1");
    ylabel("Feature x2");
    title("Classified Data");
end


function [disc,sort_disc,tau] = calc_threshold(DS,mean_ij,cov_m,w)
    PXL0 = w(1)*mvnpdf(DS(:,1:2),mean_ij(1,:),cov_m) +
w(2)*mvnpdf(DS(:,1:2),mean_ij(2,:),cov_m);
    PXL1 = w(1)*mvnpdf(DS(:,1:2),mean_ij(3,:),cov_m) +
w(2)*mvnpdf(DS(:,1:2),mean_ij(4,:),cov_m);
    disc = log(PXL1) - log(PXL0);
    sort_disc = sort(disc);
    tau = (sort_disc(1:end-1) + sort_disc(2:end)) / 2;
end

function [dec,true_pos,false_pos,error,min_error,min_error_ind] =
classify_data(DS,tau,disc,priori)
    for i = 1:length(tau)
        dec = disc >= tau(i);
        true_pos(i) = numel(find((dec==1) & (DS(:,3)==1))) /
numel(find(DS(:,3)==1));
        false_pos(i) = numel(find((dec==1) & (DS(:,3)==0))) /
numel(find(DS(:,3)==0));
        error(i) = priori(2)*false_pos(i) + priori(1)*(1-true_pos(i));
    end
    min_error = min(error);
    min_error_ind = find(error==min(error));
end

function [tau_T,dec_T,true_pos_T,false_pos_T,error_T] =
calc_theoretical_tau(priori,DS,disc)
    tau_T = log(priori(1) / priori(2));
    dec_T = disc >= tau_T;
    true_pos_T = numel(find((dec_T==1) & (DS(:,3)==1))) /
numel(find(DS(:,3)==1));
    false_pos_T = numel(find((dec_T==1) & (DS(:,3)==0))) /
numel(find(DS(:,3)==0));
    error_T = priori(2)*false_pos_T + priori(1)*(1-true_pos_T);
end
```

```
function cost = cost_function(theta,z,m,Y)
    h = 1 ./ (1 + exp(-(theta' * z)));
    cost = (-1 / m) * (sum(Y' * log(h'),'all') + sum((1 - Y)' * log(1 -
h'),'all'));
end

function [theta_ML,z] = cost_optimization_linear(DS,n)
    m = size(DS,1);
    z = [ones(m, 1),DS(:,1:2)]';
    theta_j = zeros(n+1,1);
    Y = DS(:,3);
    theta_ML = fminsearch(@(theta) cost_function(theta, z, m, Y),theta_j);
end

function [theta_MQ,z] = cost_optimization_quadratic(DS,n)
    m = size(DS,1);
    x1 = DS(:,1);
    x2 = DS(:,2);
    z = [ones(m, 1),x1,x2,x1.^2,x1.*x2,x2.^2]';
    theta_j = zeros(6,1);
    Y = DS(:,3);
    theta_MQ = fminsearch(@(theta) cost_function(theta, z, m, Y),theta_j);
end


function [error,dec_test] = logistic(theta_ML,z_test,DS)
    Y = DS(:,3);
    dec_test = (1 ./ (1+ exp(-(theta_ML' * z_test)))) >= 0.5;
    dec_test_T = dec_test';
    true_pos = numel(find((dec_test_T==1) & (Y==1)));
    true_neg = numel(find((dec_test_T==0) & (Y==0)));
    error = (10000 - (true_pos + true_neg))/100;
end
```

QUESTION 2

```
%ML HW 2
%Q2
clc; clear all; close all;

Ntrain = 100;
Nvalidate = 10000;

[trainx,trainy,valx,valy] = hw2q2(Ntrain,Nvalidate);

%ML ESTIMATOR
train_z = [trainx.^3;trainx.^2;trainx;ones(2,Ntrain)];
```

```
train_z = reshape(train_z,[2,4,Ntrain]);

w1 = zeros(2,2);
w2 = zeros(2,4);

for i = 1:Ntrain
    w1 = w1 + (squeeze(train_z(:,:,i))*squeeze(train_z(:,:,i))');
    w2 = w2 + (squeeze(train_z(:,:,i))*trainy(i));
end

theta_ML = inv(w1)*w2;

for i = 1:Nvalidate
    y(i) = estimate(valx(:,i),theta_ML);
end

square_error = sum((valy-y).^2,'all');
mean_square_error = square_error/Nvalidate;

disp('ML ESTIMATOR');
disp('Mean Squared Error = ');
disp(mean_square_error);

%MAP ESTIMATOR
gamma = linspace(0.01, 100000, 100);
variance = 0.001;
noise = normrnd(0,variance);

for i = 1:length(gamma)
    gamma_val = gamma(i);
    w1 = zeros(2);
    w2 = zeros(2, 1);
    for j = 1:Ntrain
        w1 = w1 + (squeeze(train_z(:,:,j))*squeeze(train_z(:,:,j))') +
((variance/gamma_val)*eye(2));
        w2 = w2 + (squeeze(train_z(:,:,j))*trainy(j));
    end
    theta_MAP = inv(w1)*w2;
    for j = 1:Nvalidate
        y(j) = estimate(valx(:,j),theta_MAP);
    end
    square_error = sum((valy-y).^2,'all');
    mean_square_error = square_error/Nvalidate;
    MSE(i) = mean_square_error;
end
Min_MSE = min(MSE);
Min_MSE_ind = find(MSE==Min_MSE);
Max_MSE = max(MSE);
Max_MSE_ind = find(MSE==Max_MSE);

disp('MAP ESTIMATOR');
disp('Minimum Gamma = ');
disp(gamma(Min_MSE_ind));
disp('Mean Squared Error = ');
disp(Min_MSE);
```

```
disp('Maximum Gamma = ');
disp(gamma(Max_MSE_ind));
disp('Mean Squared Error = ');
disp(Max_MSE);

figure;
plot(gamma, MSE);
xlabel('Gamma');
ylabel('Mean Squared Error');
title('Variation in MSE with respect to Gamma');

function y = estimate(x, w)
    y = (x.^3)' * w(:,1) + (x.^2)' * w(:,2) + (x') * w(:,3) + (ones(1,2) *
w(:,4));
end

function [trainx,trainy,valx,valy] = hw2q2(Ntrain,Nvalidate)
    data = generateData(Ntrain);
    figure(1), plot3(data(1,:),data(2,:),data(3,:),'.'), axis equal,
    xlabel('x1'),ylabel('x2'), zlabel('y'), title('Training Dataset'),
    trainx = data(1:2,:);
    trainy = data(3,:);
    data = generateData(Nvalidate);
    figure(2), plot3(data(1,:),data(2,:),data(3,:),'.'), axis equal,
    xlabel('x1'),ylabel('x2'), zlabel('y'), title('Validation Dataset'),
    valx = data(1:2,:);
    valy = data(3,:);
end

function x = generateData(N)
    gmmParameters.priors = [.3,.4,.3];
    gmmParameters.meanVectors = [-10 0 10;0 0 0;10 0 -10];
    gmmParameters.covMatrices(:,:,1) = [1 0 -3;0 1 0;-3 0 15];
    gmmParameters.covMatrices(:,:,2) = [8 0 0;0 .5 0;0 0 .5];
    gmmParameters.covMatrices(:,:,3) = [1 0 -3;0 1 0;-3 0 15];
    [x,labels] = generateDataFromGMM(N,gmmParameters);
end

function [x,labels] = generateDataFromGMM(N,gmmParameters)
    priors = gmmParameters.priors;
    meanVectors = gmmParameters.meanVectors;
    covMatrices = gmmParameters.covMatrices;
    n = size(gmmParameters.meanVectors,1);
    C = length(priors);
    x = zeros(n,N); labels = zeros(1,N);
    u = rand(1,N); thresholds = [cumsum(priors),1];
    for l = 1:C
    indl = find(u <= thresholds(l)); Nl = length(indl);
    labels(1,indl) = l*ones(1,Nl);
    u(1,indl) = 1.1*ones(1,Nl);
    x(:,indl) = mvnrnd(meanVectors(:,l),covMatrices(:,:,l),Nl)';
    end
end
```

QUESTION 3

```
clear all; close all; clc;

c_L = vecSpace(0.0001,500,100,"geom");

r = sqrt(rand(1,1));
angle = rand(1,1) * 2 * pi;
x = r .* cos(angle);
y = r .* sin(angle);
trueP = [x; y];

var_X = 0.25;
var_Y = 0.25;
var_I = 0.3;

v_set = [1,2,3,4];

for i = 1:length(v_set)
    m = [];
    LM_pos = landmark_pos(i);
    for j = 1:size(LM_pos,1)
        m = [m; measure(trueP,LM_pos(j,:))];
    end
    [X,Y] = meshgrid(linspace(-2,2,128),linspace(-2,2,128));
    grid_coord = cat(3,X,Y);
    cvalues = MAP_estimation(m,grid_coord,LM_pos,var_X,var_Y);
    plotting(X,Y,c_L,trueP,m,grid_coord,cvalues,LM_pos)
end

function landmarks = landmark_pos(k)
    angles = linspace(0,2*pi,k+1);
    angles = angles(1:k);
    landmarks = [cos(angles)',sin(angles)'];
end

function [measurement] = measure(trueP,lm)
    diff_T = norm(trueP - lm);
    while true
        noise = normrnd(0,0.3);
        measurement = diff_T + noise;
        if measurement >= 0
            break;
        end
    end
end
function post = MAP_estimation(measurements,mh_grid,lndmrk,var_X,var_Y)
    priori = bsxfun(@times,mh_grid,reshape(inv([var_X^2,0;
0,var_Y^2]),[1,1,2,2]));
    priori = priori .* permute(mh_grid,[1,2,4,3]);
    priori = squeeze(priori);
    range_sum = 0;
    for i = 1:length(measurements)
        r_i = measurements(i);
        lm = lndmrk(i,:);
```

```
        for j = 1:1
            lm = lndmrk(j,:);
            lm = reshape(lm,[1,1,1,2]); % add an extra dimension
            diff_i = sqrt(sum((mh_grid - lm).^2,4));
            range_sum = range_sum + ((measurements(i) - diff_i).^2 / 0.3^2);
        end
        diff_i = sqrt(sum((mh_grid - lm).^2,3));
        range_sum = range_sum + ((r_i - diff_i).^2) / 0.3^2;
    end
    result = priori + range_sum;
    post = result;
end

function plotting(X,Y,c_L,xy_T,r_meas,gp,cvalues,lm)
    hold on;
    [C,h] =
contour(gp(:,:,1),gp(:,:,2),cvalues(:,:,1),c_L,'LineWidth',1.5,'LineStyle','-
');
    colormap(parula(length(c_L)-1));
    hold off;
    fun = @(xy) sum((sqrt(sum((repmat(xy,1,size(r_meas,2)) -
landmark_pos(size(r_meas,2))).^2)) - repmat(r_meas,2,1)).^2);
    figure
    hold on;
    axis equal
    unit_circle =
viscircles([0,0],1,'Color','k','LineWidth',1.5,'LineStyle','-');
    lm = landmark_pos(length(r_meas));
    lm = squeeze(lm);
    for i = 1:length(r_meas)
        x = lm(i,1);
        y = lm(i,2);
        range_circle =
viscircles([x,y],r_meas(i),'Color','r','LineWidth',1.5,'LineStyle','-');

plot(x,y,'go','MarkerFaceColor','none','MarkerSize',6,'LineWidth',1.5);
    end
    plot(xy_T(1),xy_T(2),'o','MarkerSize',8,'LineWidth',1.5);
    xlabel('X')
    ylabel('Y')
    title(['MAP Estimation for k = ',num2str(length(r_meas))])
    xlim([-2,2])
    ylim([-2,2])
    c = colorbar;
    hold off;
end
```