Michael Hodges

All files can be found here
https://github.com/Michael-Hodges/EECE5644_Machine_Learning.git
or in the appendix

Problem 1

Design a classifier that achieves minimum probability of error for a three-class problem where the class priors are respectively $P(L = 1) = 0.15$, $P(L = 2) = 0.35$, $P(L = 3) = 0.5$ and the class-conditional data distributions are all Gaussians for two-dimensional data vectors:

$\mathcal{N}([-1; 0], [1, -0.4; -0.4, 0.5]), \mathcal{N}([1; 0], [0.5, 0; 0, 0.2]), \mathcal{N}([0; 1], [0.1, 0; 0, 0.1])$.

Generate 10000 samples according to this data distribution, keep track of the true class labels for each sample. Apply your optimal classifier designed as described above to this dataset and obtain decision labels for each sample. Report the following:

- actual number of samples that were generated from each class;
- the confusion matrix for your classifier consisting of number of samples decided as class $r \in 1, 2, 3$ when their true labels were class $c \in 1, 2, 3$, using r, c as row/column indices;
- the total number of samples misclassified by your classifier;
- an estimate of the probability of error your classifier will achieves, based on these samples;
- a visualization of the data as a 2-dimensional scatterplot, with true labels and decision labels indicated using two separate visulization cues, such as marker shape and marker color;
- a clear but brief description of the results presented as described above.

Note: See the attached generateData Exam1Question1.m Matlab script for data generation.

We will use a 0-1 loss to minimize the total error. The risk functions we will be using is as follows:

1. $R(\alpha_1|x) = \lambda_{11}P(w_1|x) + \lambda_{12}P(w_2|x) + \lambda_{13}P(w_3|x)$
2. $R(\alpha_1|x) = \lambda_{21}P(w_1|x) + \lambda_{22}P(w_2|x) + \lambda_{23}P(w_3|x)$
3. $R(\alpha_1|x) = \lambda_{31}P(w_1|x) + \lambda_{32}P(w_2|x) + \lambda_{33}P(w_3|x)$

Our decision rule is as follows:

1. Choose class 1 if: $R(\alpha_1|x) < R(\alpha_2|x)$ and if $R(\alpha_1|x) < R(\alpha_3|x)$
2. Choose class 2 if: $R(\alpha_2|x) < R(\alpha_1|x)$ and if $R(\alpha_2|x) < R(\alpha_3|x)$
3. Choose class 3 if: $R(\alpha_3|x) < R(\alpha_1|x)$ and if $R(\alpha_3|x) < R(\alpha_2|x)$

Using 0-1 loss to minimize total error it is easy to see that the rules simplify to as follows:

1. Choose class 1 if: $P(w_1|x) > P(w_2|x)$ and if: $P(w_1|x) > P(w_3|x)$
2. Choose class 2 if: $P(w_2|x) > P(w_1|x)$ and if: $P(w_2|x) > P(w_3|x)$
3. Choose class 3 if: $P(w_3|x) > P(w_1|x)$ and if: $P(w_3|x) > P(w_2|x)$

We will use Bayes' rule to generate our data as we do not know the posteriors so we will use the priors and likelihoods (.e.g. $P(x|w_1)P(w_1)$).

The results are shown below for one instance of this test:
Actual number of samples per class:

Class 1: 1530

Class 2: 3484

Class 3: 4986

Confusion Matrix for Rows as decision and columns as true Labels:

$$\begin{bmatrix} 1238 & 118 & 23 \\ 199 & 3150 & 107 \\ 93 & 216 & 4856 \end{bmatrix}$$

Total Errors: 756
Probability of error: 7.56

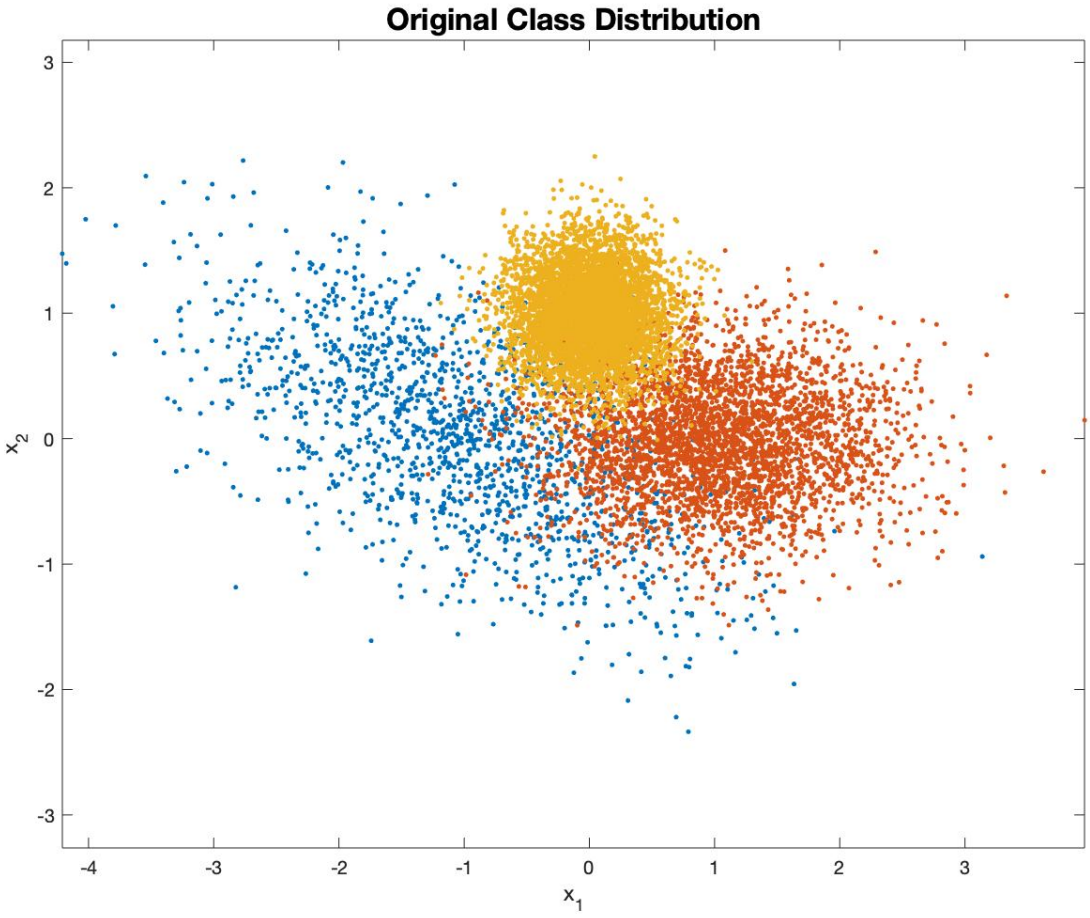

Figure 1: Original Distribution

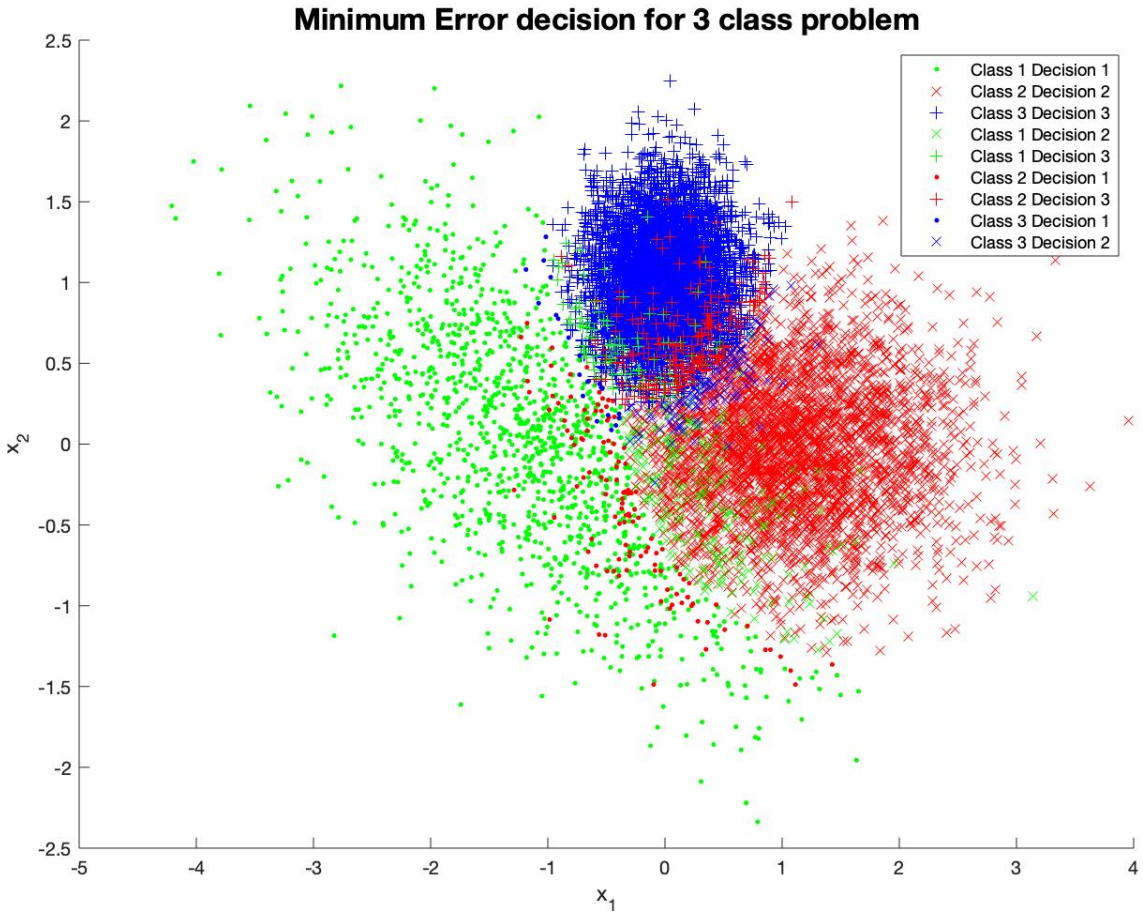

Figure 2: Class Labels and Decisions as chosen by the classifier. True class is shown by color, decision is shown by shape.

As we can see in the figure there is a clear boundary for the decisions being made between the three classes, and if that is not clear then the plot below shows only the accurately classified items, which has a clearer boundary where the above plot shows all the classification errors.

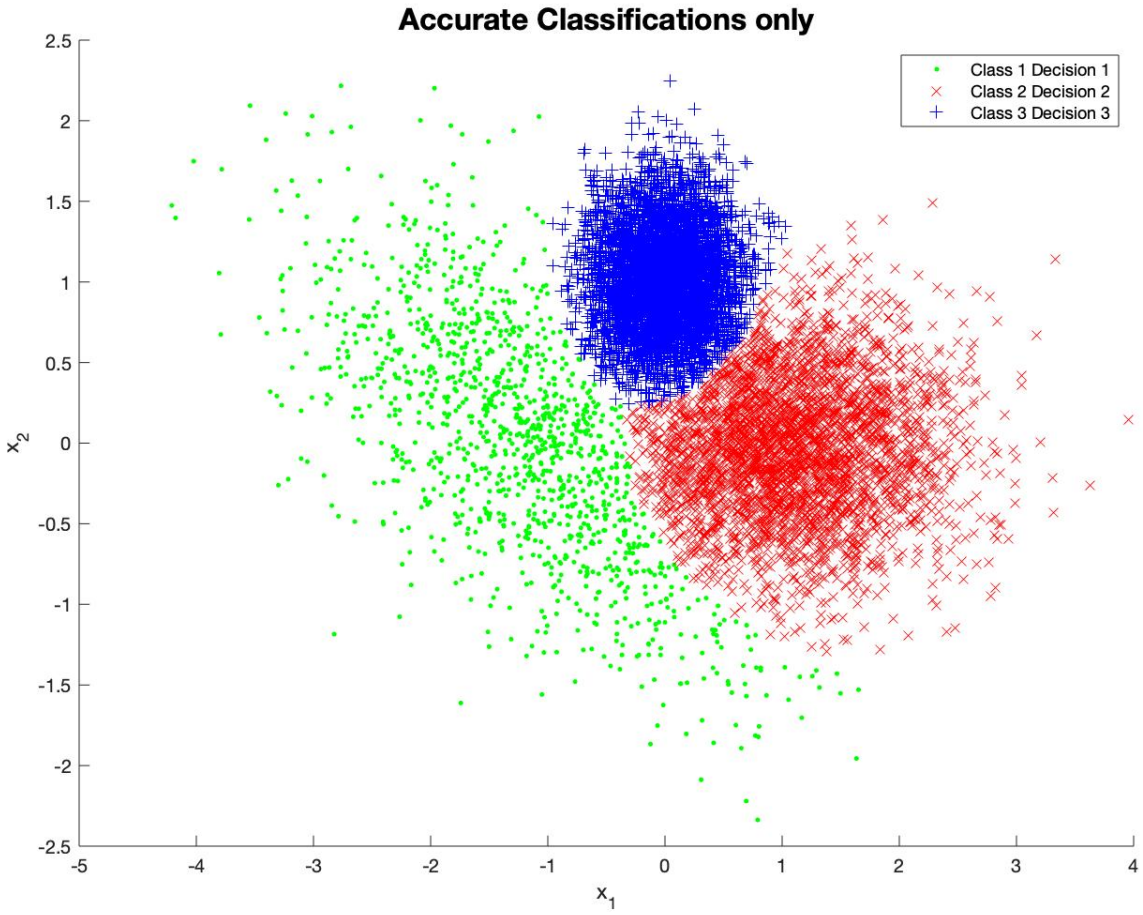

Figure 3: Accurate Classifications only

Problem 2

The problem we want to solve is as follows:

$$\begin{aligned} X_{map}, Y_{Map} &= \underset{X,Y}{\operatorname{argmax}} P([x,y]|r_i) \quad \text{for i} \in 1...k \\ &= \underset{X,Y}{\operatorname{argmax}} P(r_i|[x,y])P([x,y]) \\ &= \underset{X,Y}{\operatorname{argmax}} \sum_{i=1}^{k} \ln P(r_i|[x,y]) + \ln P([x,y]) \end{aligned}$$

We know that our measurement has mean of the true measurement D_Ti and variance of the noise σ_i^2 such that we get $\mathcal{N}(D_{Ti}, \sigma_i^2)$, using this in the equation gets us the following:

$$= \underset{X,Y}{\operatorname{argmax}} \sum_{i=1}^{k} \ln \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{\frac{-(r_i - d_i(x,y))^2}{2\sigma_i^2}} + \ln \frac{1}{(2\pi\sigma_x\sigma_y)} e^{\frac{-1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix}}$$

Simplifying gets the following:

$$= \underset{X,Y}{\operatorname{argmax}} \sum_{i=1}^{k} -\frac{1}{2}\ln 2\pi\sigma_i^2 - \frac{(r_i - d_i(x,y))^2}{2\sigma_i^2} - \ln 2\pi\sigma_x\sigma_y - \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix}$$

We can drop the first term and third term since none of them are in terms of x or y giving us the final optimization problem:

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

plugging in for equation d_i:

$$= \underset{X,Y}{\operatorname{argmax}} \sum_{i=1}^{k} -\frac{(r_i - \sqrt{(x-x_i)^2 + (y-y_i)^2})^2}{\sigma_i^2} - \begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}^{-1}\begin{bmatrix} x \\ y \end{bmatrix}$$

in the implementation of our code we will change this to a minimization problem and drop the negatives as well as simplify the second term to make the implementation easier s.t. it is $\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)$. Therefore, the true location will be on the minimum contour location on the following plots.

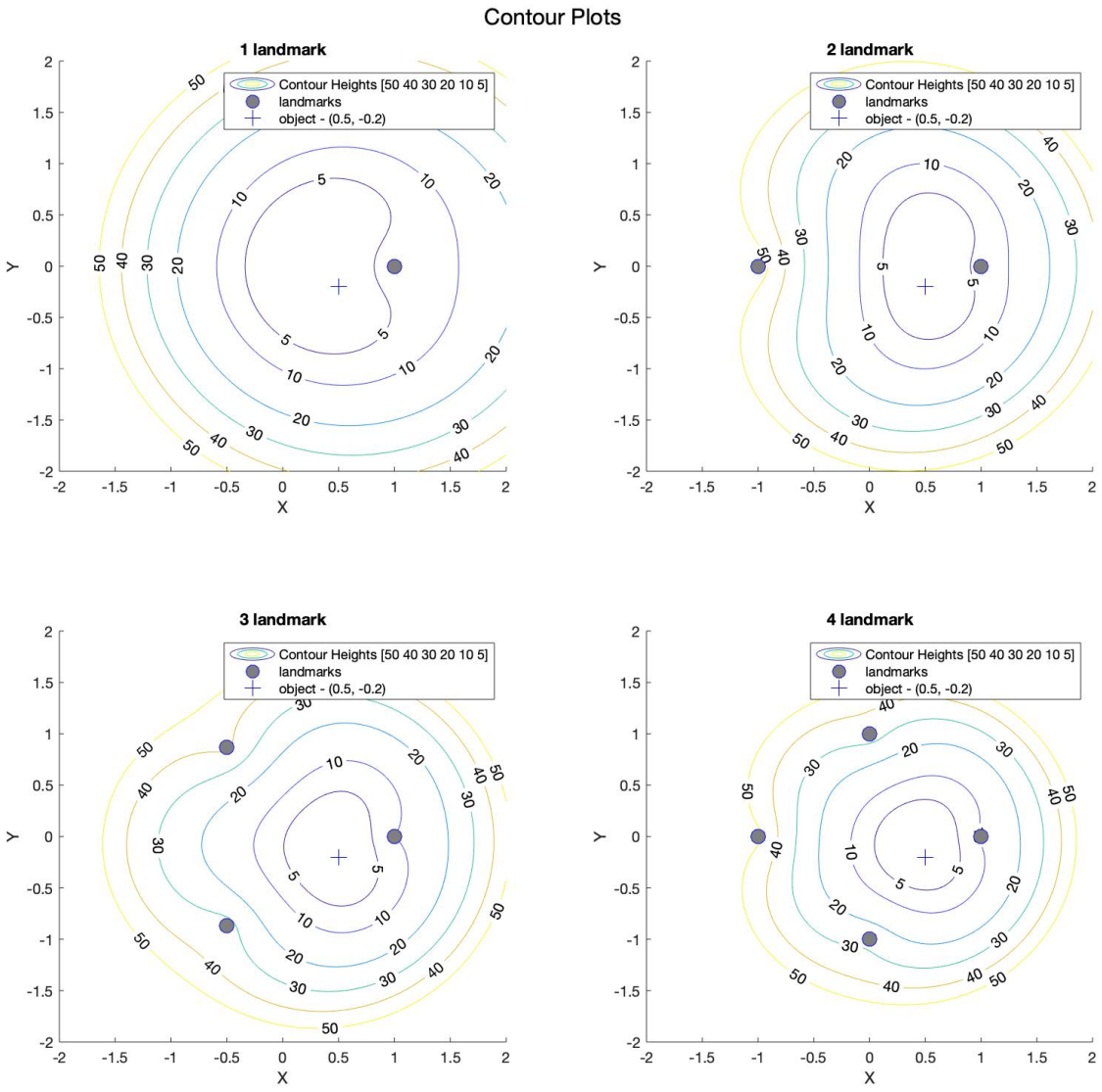

Figure 4: Contours for 1-4 landmarks

The code works as follows:

1. Initialize initial conditions, which includes true location, landmark locations, and noise values. Note: for our noise we use the the values as stated in the problem σ_x, $\sigma_y = 0.25$ and $\sigma_i = 0.1$
2. Generate samples r_i as shown in the problem statement. We check to ensure the value is not negative
3. for the number of landmarks we have loop through and calculate the sums that are highlighted above. Finally, add the last term also that is shown in the final highlighted

solution.

4. plot the given contour for specified number of landmarks
5. repeat for all landmarks

We can see that as we increase the number of landmarks our accuracy steadily increases. Although the position estimate for 3 and 4 landmarks is roughly similar, the total area inside the smallest contour decreases with 4 landmarks. We can see this by looking at the center of the innermost contour. as we increase the true position steadily moves closer to the center of the contour at 3 landmarks. The 3 and 4 landmarks have a fairly similar center value, but when we add a 4th landmark the area of the innermost contour shrinks thus increasing our certainty of the true location.

Problem 3

The Problem we want to solve is as follows:

Let $X \in \mathbb{R}^{4xN}$, $Y \in \mathbb{R}^{Nx1}$, and $\mathbf{w} \in \mathbb{R}^{4x1}$, where $\mathbf{w}$ is the vector of coefficients a, b, c, d, Y is the vectour of outputs and X is the matrix of input values

$$\begin{bmatrix} x_1^3 & x_2^3 & ... & x_N^3 \\ x_1^2 & x_2^2 & ... & x_N^2 \\ x_1 & x_2 & ... & x_N \\ 1 & 1 & ... & 1 \end{bmatrix}$$

$$\begin{aligned} \mathbf{w}_{MAP} &= \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}|Y) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} P(Y|\mathbf{w})P(\mathbf{w}) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \ln P(Y|\mathbf{w}) + \ln P(\mathbf{w}) \end{aligned}$$

We know that $\mathbf{w} \sim \mathcal{N}(0, \boldsymbol{\gamma}^2\mathbf{I})$, and given the noise we know that $P(Y|\mathbf{w}) \sim \mathcal{N}(X^T\mathbf{w}, \sigma^2)$. Therefore, after simplifying and dropping constants we get the following:

$$= \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\frac{(Y - X^T\mathbf{w})^T(Y - X^T\mathbf{w})}{\sigma^2} + \frac{1}{2}\mathbf{w}^T(\boldsymbol{\gamma}^2\boldsymbol{I})^{-1}\mathbf{w}$$

Taking the derivative and setting equal to zero we get the following(note since $(\boldsymbol{\gamma}^2\mathbf{I})^{-1}$ is symmetric and diagonal, when we derive we can take 2 times the matrix, see cookbook

$$0 = -\frac{X(Y - X^T\mathbf{w})}{\sigma^2} + (\boldsymbol{\gamma}^2\boldsymbol{I})^{-1}\mathbf{w}$$

factoring $\mathbf{w}$ out we get:

$$\mathbf{w}_{MAP}^T = \frac{(XY)^T}{\sigma^2}\left[\frac{XX^T}{\sigma^2} + (\boldsymbol{\gamma}^2\mathbf{I})^{-1}\right]^{-1}$$

The following figures will display our results for the tests chosen. Our real roots are chosen as -0.8, 0.2, and 0.8. Figures 5-7 show the samples to be N = 10 (as the problem requested), and taking 101 values of γ between $[10^{-3}, 10^3]$. Refer to the captions to see what each plot specifically shows. Next we will increase our sample rate to N=100 as the plots improve greatly with more samples. The noise is held constant with a $\sigma^2 = 0.75$ In general, as the value of gamma increases, the effect of the prior diminishes and we are left with the ML estimate. Since our given prior does not match our true values, as in our prior data does not help us achieve the right solution, as the value of gamma increases and thus it's affect on the MAP estimate decreases we achieve a lower error on our MAP estimate. However, as we see in the case of N = 10. If we don't have enough samples we have the potential to have a

large error. As we show for N=100, with enough samples we are closer able to approximate this function.

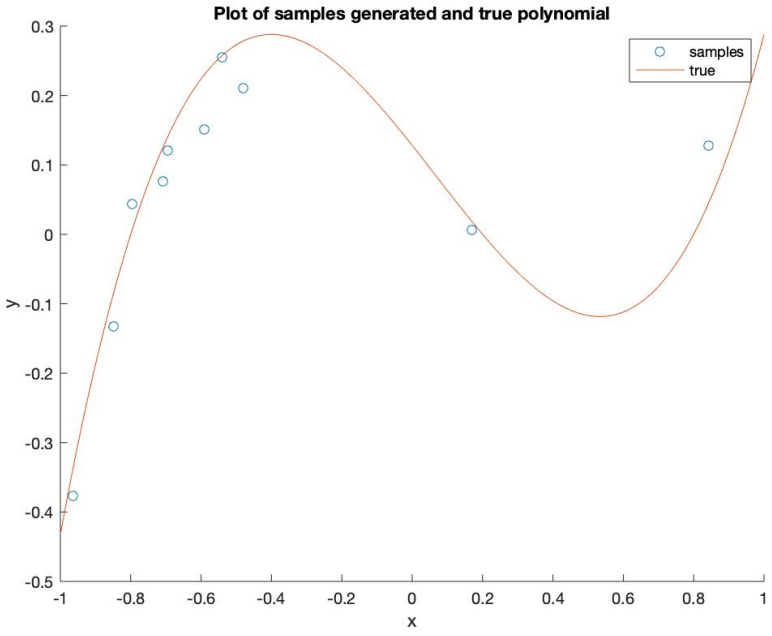

Figure 5: Samples and true plot for N=10

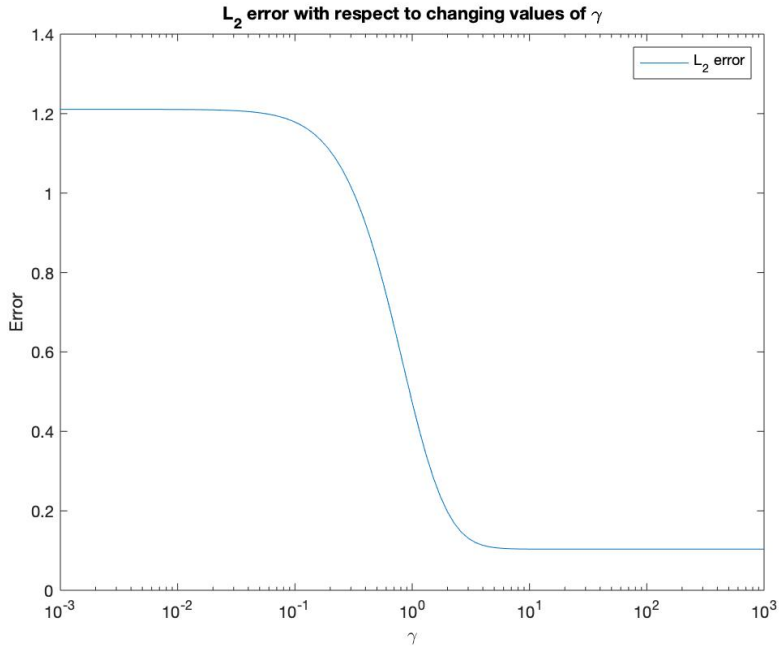

Figure 6: ℓ_2 distance for values of γ with N = 10

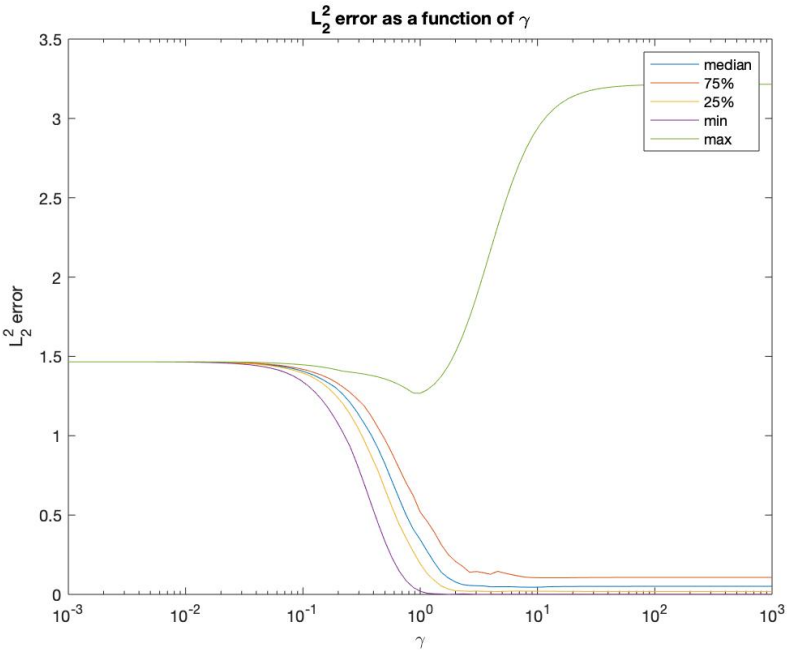

Figure 7: Squared error for 100 tests at each value of γ with N = 10

Now we will show the same results will N = 100, which performs much better. All other parameters hold the same.

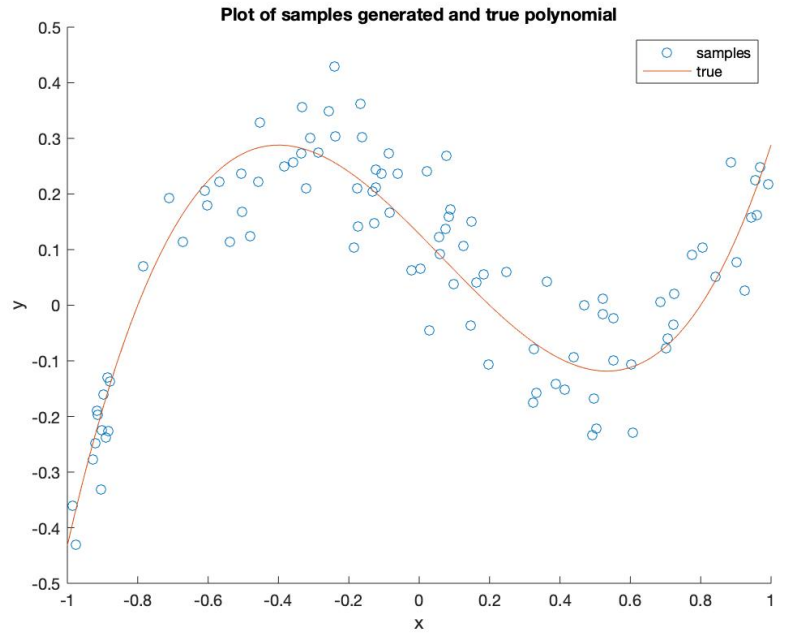

Figure 8: Samples and true plot for N=100

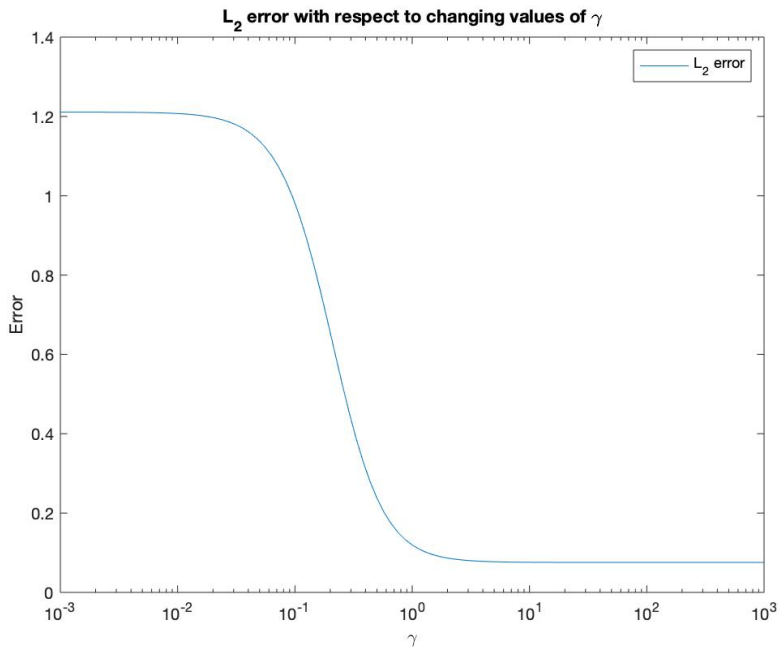

Figure 9: ℓ_2 distance for values of γ with N = 100

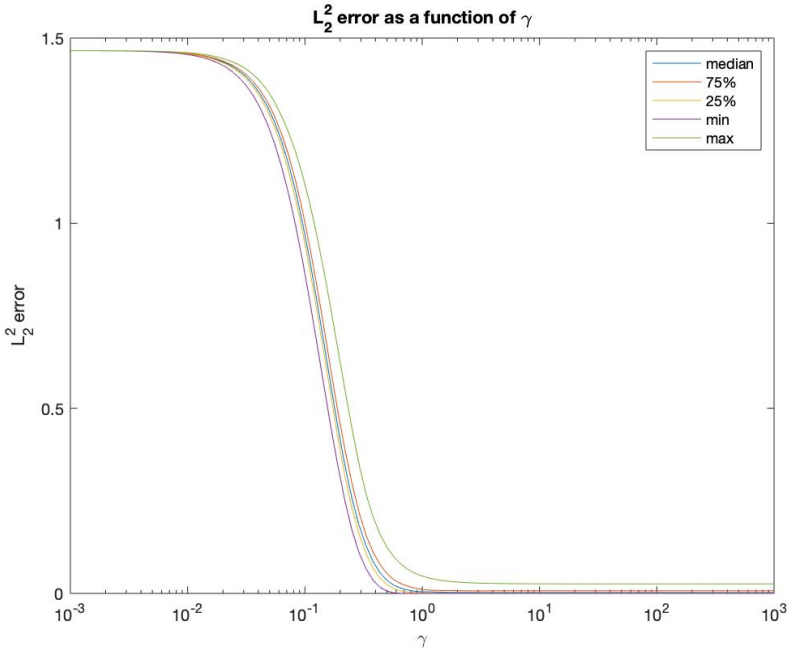

Figure 10: Squared error for 100 tests at each value of γ with N = 100

THE FOLLOWING ARE SUPPLEMENTARY ONLY AND SHOW THE SOLUTION

USING SUMS INSTEAD OF TRUE VECTOR MATH.

$$\begin{aligned}\mathbf{w}_{MAP} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|D) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(D|\mathbf{w})P(\mathbf{w}) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i=1}^{N} \ln P(d_i|\mathbf{w}) + \ln P(\mathbf{w})\end{aligned}$$

We know that $\mathbf{w} \sim \mathcal{N}(0, \boldsymbol{\gamma}^2\mathbf{I})$, and given the noise we know that $P(D|\mathbf{w}) \sim \mathcal{N}(\mathbf{w}^T\mathbf{x}_i, \sigma^2)$ where $\mathbf{w}$ is the vector of coefficients a, b, c, d. and $\mathbf{x}_i$ is the vector of x values $(x_i^3, x_i^2, x_i, 1)$. Therefore, after simplifying and dropping constants we get the following:

$$= \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2}\frac{(y_i - \mathbf{w}^T\mathbf{x}_i)^2}{\sigma^2} + \frac{1}{2}\mathbf{w}^T(\boldsymbol{\gamma}^2\boldsymbol{I})^{-1}\mathbf{w}$$

To find the solution we take the derivative with respect to w and set it equal to zero which gets us the following:

$$0 = \sum_{i=1}^{N} \frac{-2}{2}\frac{(y_i - \mathbf{w}^T\mathbf{x}_i)\mathbf{x}_i^T}{\sigma^2} + \frac{2}{2}\mathbf{w}^T(\boldsymbol{\gamma}^2\boldsymbol{I})^{-1}$$

the matrix, $\boldsymbol{\gamma}^2\boldsymbol{I}$ is diagonal so taking the derivative we can take two times the original matrix since $(\boldsymbol{\gamma}^2\boldsymbol{I}) = (\boldsymbol{\gamma}^2\boldsymbol{I})^T$

$$0 = \sum_{i=1}^{N} \frac{-y_i\mathbf{x}_i^T + \mathbf{w}^T\mathbf{x}_i\mathbf{x}_i^T}{\sigma^2} + \mathbf{w}^T(\boldsymbol{\gamma}^2\boldsymbol{I})^{-1}$$

$$\mathbf{w}^T(\boldsymbol{\gamma}^2\boldsymbol{I})^{-1} + \sum_{i=1}^{N} \frac{\mathbf{w}^T\mathbf{x}_i\mathbf{x}_i^T}{\sigma^2} = \sum_{i=1}^{N} \frac{y_i\mathbf{x}_i^T}{\sigma^2}$$

$$\mathbf{w}^T\left[(\boldsymbol{\gamma}^2\boldsymbol{I})^{-1} + \sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\sigma^2}\right] = \sum_{i=1}^{N} \frac{y_i\mathbf{x}_i^T}{\sigma^2}$$

$$\mathbf{w}_{MAP}^T = \sum_{i=1}^{N} \frac{y_i\mathbf{x}_i^T}{\sigma^2}\left[(\boldsymbol{\gamma}^2\boldsymbol{I})^{-1} + \sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\sigma^2}\right]^{-1}$$

If we generalize this sum to using matrices as described above we get

$$\mathbf{w}_{MAP}^T = \frac{Y^TX^T}{\sigma^2}\left[(\boldsymbol{\gamma}^2\boldsymbol{I})^{-1} + \frac{XX^T}{\sigma^2}\right]^{-1}$$

Appendix

1. Problem 1

```
m(:,1) = [-1;0]; Sigma(:,:,1) = 0.1*[10 -4;-4,5]; % mean and covariance of data pdf conditioned on label 3
m(:,2) = [1;0]; Sigma(:,:,2) = 0.1*[5 0;0,2]; % mean and covariance of data pdf conditioned on label 2
m(:,3) = [0;1]; Sigma(:,:,3) = 0.1*eye(2); % mean and covariance of data pdf conditioned on label 1
classPriors = [0.15,0.35,0.5]; classPriors1 = [0.15 0.35,0.51]; thr = [0,cumsum(classPriors1)];
N = 10000; u = rand(1,N); L = zeros(1,N); x = zeros(2,N);
figure(1),clf, colorList = 'rbg';
for l = 1:3
    indices = find(thr(l)<=u & u<thr(l+1)); % fixed using classPriors1 adding a small term to last prior if u happens to be precisely 1, that sample will get omitted - needs to be fixed
    L(1,indices) = l*ones(1,length(indices));
    x(:,indices) = mvnrnd(m(:,l),Sigma(:,:,l),length(indices))';
    figure(1), plot(x(1,indices),x(2,indices),'.','MarkerFaceColor',colorList(l)); axis equal, hold on,
end
title('Original Class Distribution', 'fontsize', 16), xlabel('x_1'), ylabel('x_2');
disp("CLass 1: " + length(find(L==1))); disp("CLass 2: " + length(find(L==2))); disp("Class 3: " +length(find(L==3)));
%for l = 1:3
 %   indices = find(mvnpdf(x',m(:,l)',Sigma(:,:,l))*classPriors(l)>mvnpdf(x',m(:,mod(l+1,3))',Sigma(:,:,l+1))*classPriors(l+1) & mvnpdf(x',m(:,l)',Sigma(:,:,l))*classPriors(l)>mvnpdf(x',m(:,l+2)',Sigma(:,:,l+2))*classPriors(l+2));
%end
decision = zeros(1,N);
indices = find(mvnpdf(x',m(:,1)',Sigma(:,:,1))*classPriors(1)>mvnpdf(x',m(:,2)',Sigma(:,:,2))*classPriors(2) & mvnpdf(x',m(:,1)',Sigma(:,:,1))*classPriors(1)>mvnpdf(x',m(:,3)',Sigma(:,:,3))*classPriors(3));
decision(1,indices) = 1*ones(1,length(indices));
indices = find(mvnpdf(x',m(:,2)',Sigma(:,:,2))*classPriors(2)>mvnpdf(x',m(:,1)',Sigma(:,:,1))*classPriors(1) & mvnpdf(x',m(:,2)',Sigma(:,:,2))*classPriors(2)>mvnpdf(x',m(:,3)',Sigma(:,:,3))*classPriors(3));
decision(1,indices) = 2*ones(1,length(indices));
indices = find(mvnpdf(x',m(:,3)',Sigma(:,:,3))*classPriors(3)>mvnpdf(x',m(:,1)',Sigma(:,:,1))*classPriors(1) & mvnpdf(x',m
```

```
    (:,3)',Sigma(:,:,3))*classPriors(3)>mvnpdf(x',m(:,2)',Sigma
    (:,:,2))*classPriors(2));
decision(1,indices) = 3*ones(1,length(indices));
confusion = zeros(3);
errors = 0;
for i = 1:3
    for j =1:3
        confusion(i,j) = length(find(decision==i & L==j));
        if i ~= j
            errors = errors+length(find(decision==i & L==j));
        end
    end
end


disp("Confusion Matrix for Rows as decision and columns as true
    Labels: " );
disp(confusion);
disp("Total Errors: "+errors);
disp("Probability of error: " + (errors/N)*100 + "%");



figure(2); hold on;
plot(x(1,find(L==1 & decision ==1)), x(2,find(L==1 & decision
   ==1)), '.g');
plot(x(1,find(L==1 & decision ==2)), x(2,find(L==1 & decision
   ==2)), '.r');
plot(x(1,find(L==1 & decision ==3)), x(2,find(L==1 & decision
   ==3)), '.b');

plot(x(1,find(L==2 & decision ==1)), x(2,find(L==2 & decision
   ==1)), '+g');
plot(x(1,find(L==2 & decision ==2)), x(2,find(L==2 & decision
   ==2)), '+r');
plot(x(1,find(L==2 & decision ==3)), x(2,find(L==2 & decision
   ==3)), '+b');

plot(x(1,find(L==3 & decision ==1)), x(2,find(L==3 & decision
   ==1)), '*g');
plot(x(1,find(L==3 & decision ==2)), x(2,find(L==3 & decision
   ==2)), '*r');
plot(x(1,find(L==3 & decision ==3)), x(2,find(L==3 & decision
   ==3)), '*b');
legend('Class 1 Decision 1', 'Class 1 Decision 2', 'Class 1
   Decision 3', ...
```

```
'Class 2 Decision 1', 'Class 2 Decision 2', 'Class 2 Decision 3',...
'Class 3 Decision 1', 'Class 3 Decision 2', 'Class 3 Decision 3')
title('Minimum Error decision for 3 class problem', 'fontsize', 16),
xlabel('x_1'), ylabel('x_2')
%plot(x(1,find(L==1 & decision ==1)), x(2,find(L==1 & decision ==1)), '.');
```

2. Problem 2

```
clear all; close all; clc;

%set tru object location within the unit circle
r_true = [0.5  -.2];
%values to iterate over when doing contour plot
x = linspace(-2,2,200);

[X, Y] = meshgrid(x);
nsig = 0.1; xsig = 0.25; ysig = 0.25;
%set landmark locations
k(:,:,1) = [1  0; 0  0; 0  0; 0  0];%set landmark on unit circle
k(:,:,2) = [1  0; -1  0; 0  0; 0  0];
k(:,:,3) = [1  0; cos(2*pi/3) sin(2*pi/3); cos(4*pi/3) sin(4*pi/3); 0  0];
k(:,:,4) = [1  0; 0  1; -1  0; 0  -1];
figure(1);
sgtitle('Contour Plots')
r = zeros(1,4);
for i = 1:4
    for j = 1:i
        tmp = -1; %set initial condition of loop to be negative.
                  %Loop continues to take samples until we get a positive
                  %range measurement
        while tmp<0
            tmp = norm(r_true(1,:)'-k(j,:,i)')+normrnd(0,nsig);
        end
        r(j) = tmp;
    end
    z = 0;
    for j =1:i
        z = z+(((r(j)-sqrt(((X-k(j,1,i)).^2) + ((Y-k(j,2,i)).^2))).^2)/nsig);%-(1/2).*[X Y];%.*inv([nsig  0; 0  nsig]).*[X Y]';
```

```
        %z = z-sqrt(((X-k(i,1,j)).^2) + ((Y-k(i,2,j)).^2))
            ;%-(1/2).*[X Y];%.*inv([nsig 0; 0 nsig]).*[X Y]';
    end
        z = z+(X.^2/xsig + Y.^2/ysig);
        subplot(2,2,i);hold on;
        [C,h] = contour(X,Y,z,[50 40 30 20 10 5]);
        clabel(C,h);
        plot(k(1:j,1,j),k(1:j,2,j),'ob','markersize',10,'
            MarkerFaceColor',[0.5,0.5,0.5]);
        plot(r_true(:,1),r_true(:,2),'+b','markersize',10,'
            MarkerFaceColor',[0.5,0.5,0.5]);
        title([num2str(i), ' landmark']);
        loc_string = num2str(r_true);
        legend('Contour Heights [50 40 30 20 10 5]','landmarks'
            ,['object - (' num2str(r_true(1)) ', ' num2str(
            r_true(2)) ')' ])
        xlabel('X'); ylabel('Y');
        axis([-2 2 -2 2]);
end


% figure(1); hold on;
% plot(r_true(:,1),r_true(:,2),'+');
% plot(k1(:,1),k1(:,2),'o');
% scatter(k2(:,1),k2(:,2),'o');
% scatter(k3(:,1),k3(:,2)'o');
% %scatter(k4(:,1),k4(:,2),'o');
% grid on;
```

3. Problem 3.1 using vector Math

```

clear all, close all; clc;
N = 100; % Number of Samples
x = (-1+2*rand(1,N))'; % Generate N samples from [-1,1]
x_vec = [x.^3 x.^2 x ones(1,N)'];
w = ([1000 -200 -640 128]/1000)'; % true coefficients w real
    roots at (x+.8)(x-.2)(x-.8)
y = zeros(1,N)'; % used to store values of y generated from
    plot
nsig = .075;
noise = normrnd(0,nsig,N,1);
y = x_vec'*w+noise;

x_true = linspace(-1, 1);
figure(1); hold on;
scatter(x,y);
```

```
plot(x_true,(x_true+0.8).*(x_true-.2).*(x_true-.8));
legend('samples','true');
title('Plot of samples generated and true polynomial');
xlabel('x'); ylabel('y');

iterations = 101;
l2Map_error = zeros(1,iterations);
C=3;
val = logspace(-C,C,iterations);
w_map = zeros(1,4);
for z = 1:iterations
    Gamma = val(z);
    w_map = ((x_vec*y)/nsig)'/((x_vec*x_vec')/nsig + inv(Gamma
        ^2*eye(4)));
    w_map = w_map';
    l2Map_error(z) = norm(w-w_map);

end
figure(2);
semilogx(val,l2Map_error);
legend('L_2 error');
title('L_2 error with respect to changing values of \gamma');
xlabel('\gamma'); ylabel('Error');
```

4. Problem 3.2 using vector Math

```

clear all, close all; clc;

N = 10; % Number of Samples
w = ([1000 -200 -640 128]/1000)'; % true coefficients w real
    roots at (x+.8)(x-.2)(x-.8)
y = zeros(1,N)'; % used to store values of y generated from
    plot
nsig = .075;

iterations = 101;
l2Map_error = zeros(1,iterations);
C=3;
val = logspace(-C,C,iterations);
w_map = zeros(1,4);
trials = 100;
stored = zeros(trials,iterations);

for l = 1:trials % number of trials to perform
    x = (-1+2*rand(1,N))'; % Generate N samples from [-1,1]
    x_vec = [x.^3 x.^2 x ones(1,N)'];
```

```
20      noise = normrnd(0,nsig,N,1);
21      y = x_vec'*w+noise;
22
23      for z = 1:iterations
24          Gamma = val(z);
25          A = zeros(1,4);
26          B = zeros(4);
27          w_map = ((x_vec*y)/nsig)'/((x_vec*x_vec')/nsig + inv(
                Gamma^2*eye(4)));
28          w_map = w_map';
29          l2Map_error(z) = norm(w-w_map);
30      end
31      stored(l,:) = l2Map_error;
32  end
33  figure(1);
34  %semilogx(val,l2Map_error);
35  stored = stored.^2;
36  semilogx(val, median(stored));hold on;
37  semilogx(val, prctile(stored,75));
38  semilogx(val, prctile(stored, 25));
39  semilogx(val, min(stored));
40  semilogx(val, max(stored));
41  legend('median','75%', '25%', 'min', 'max');
42  xlabel('\gamma'); ylabel('L_2^2 error')
43  title('L_2^2 error as a function of \gamma')
44
45  % figure(2);
46  % boxplot(stored, val);
47  % set(gca,'xscale','log');
```

5. Problem 3.1 using sums

```
1
2  clear all, close all; clc;
3  N = 100; % Number of Samples
4  x = (-1+2*rand(1,N))'; % Generate N samples from [-1,1]
5  x_vec = [x.^3 x.^2 x ones(1,N)'];
6  w = ([1000 -200 -640 128]/1000)'; % true coefficients w real
       roots at (x+.8)(x-.2)(x-.8)
7  y = zeros(1,N)'; % used to store values of y generated from
       plot
8  nsig = .075;
9  noise = normrnd(0,nsig,N,1);
10 y = x_vec'*w+noise;
11
12 x_true = linspace(-1, 1);
13 figure(1); hold on;
```

```
scatter(x,y);
plot(x_true,(x_true+0.8).*(x_true-.2).*(x_true-.8));
legend('samples','true');
title('Plot of samples generated and true polynomial');
xlabel('x'); ylabel('y');

iterations = 101;
l2Map_error = zeros(1,iterations);
C=3;
val = logspace(-C,C,iterations);
w_map = zeros(1,4);
for z = 1:iterations
    Gamma = val(z);
    w_map = ((x_vec*y)/nsig)'/((x_vec*x_vec')/nsig + inv(Gamma
        ^2*eye(4)));
    w_map = w_map';
    l2Map_error(z) = norm(w-w_map);

end
figure(2);
semilogx(val,l2Map_error);
legend('L_2 error');
title('L_2 error with respect to changing values of \gamma');
xlabel('\gamma'); ylabel('Error');
```

6. Problem 3.2 using sums

```

clear all, close all; clc;

N = 10; % Number of Samples
w = ([1000 -200 -640 128]/1000)'; % true coefficients w real
    roots at (x+.8)(x-.2)(x-.8)
y = zeros(1,N)'; % used to store values of y generated from
    plot
nsig = .075;

iterations = 101;
l2Map_error = zeros(1,iterations);
C=3;
val = logspace(-C,C,iterations);
w_map = zeros(1,4);
trials = 100;
stored = zeros(trials,iterations);

for l = 1:trials % number of trials to perform
    x = (-1+2*rand(1,N))'; % Generate N samples from [-1,1]
```

```
19      x_vec = [x.^3 x.^2 x ones(1,N)']';
20      noise = normrnd(0,nsig,N,1);
21      y = x_vec'*w+noise;
22
23      for z = 1:iterations
24          Gamma = val(z);
25          A = zeros(1,4);
26          B = zeros(4);
27          w_map = ((x_vec*y)/nsig)'/((x_vec*x_vec')/nsig + inv(
                Gamma^2*eye(4)));
28          w_map = w_map';
29          l2Map_error(z) = norm(w-w_map);
30      end
31      stored(l,:) = l2Map_error;
32  end
33  figure(1);
34  %semilogx(val,l2Map_error);
35  stored = stored.^2;
36  semilogx(val, median(stored));hold on;
37  semilogx(val, prctile(stored,75));
38  semilogx(val, prctile(stored, 25));
39  semilogx(val, min(stored));
40  semilogx(val, max(stored));
41  legend('median','75%', '25%', 'min', 'max');
42  xlabel('\gamma'); ylabel('L_2^2 error')
43  title('L_2^2 error as a function of \gamma')
44
45  % figure(2);
46  % boxplot(stored, val);
47  % set(gca,'xscale','log');
```