BM 59D Homework #2

Kaan Oktay, *2016701108*

I. Question 1

A. Part A

In this part, training data was used. Throughout this report, class labelled by '1' is represented as C_1 and class labelled by '-1' is represented as C_2. The class priors were $P(C_i)$ calculated as below.

$$P(C_1) = \frac{50}{125} = 0.4 \qquad P(C_2) = \frac{75}{125} = 0.6 \tag{1}$$

Also, the class likelihoods (assuming Gaussian distribution) were calculated as below and plotted in Fig. (1). In this equation, Gaussian distribution parameters μ_i and σ_i values were calculated using maximum likelihood estimation and they are basically class means and standard deviations respectively.

$$p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \cdot \exp(-\frac{(x-\mu_i)^2}{2\sigma_i^2}) \tag{2}$$

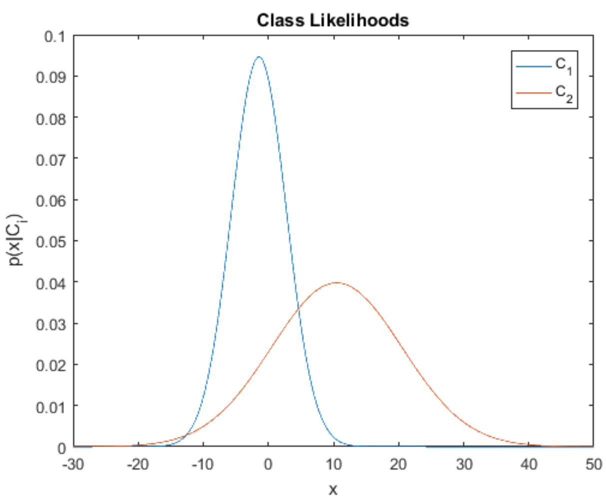

Fig. 1. Class likelihoods by assuming Gaussian distribution.

In addition, the evidence was calculated as below and plotted in Fig. (2).

$$p(x) = p(x|C_1) \cdot P(C_1) + p(x|C_2) \cdot P(C_2) \tag{3}$$

After the calculations above, class posteriors were calculated as below and plotted in Fig. (3).

$$P(C_i|x) = \frac{P(C_i) \cdot p(x|C_i)}{p(x)} \tag{4}$$

Then, risks for choosing actions (0/1 loss and no rejection) were calculated as below and plotted in Fig. (4) where α_i corresponds to the action of choosing class C_i.

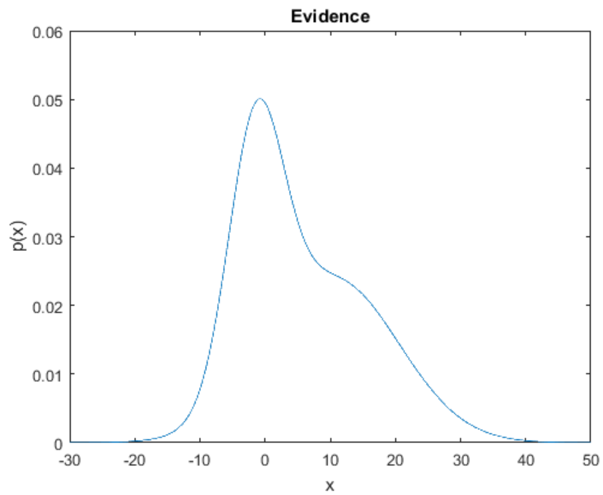

Fig. 2. Probability density of the evidence.

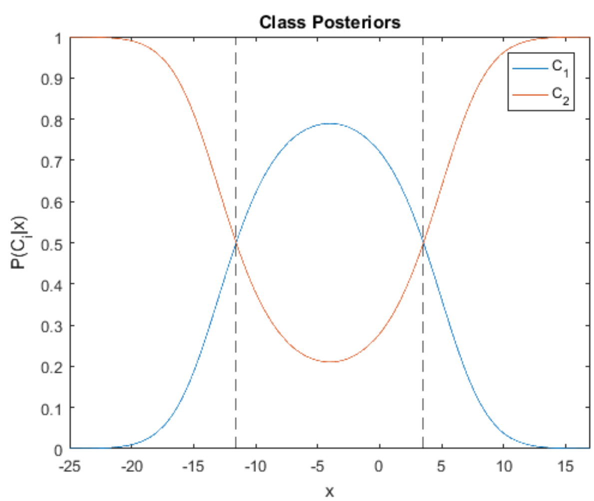

Fig. 3. Class posteriors.

$$R(\alpha_i|x) = 1 - P(C_i|x) \tag{5}$$

Also, the discriminants were calculated as below and plotted in Fig. (5).

$$g_i(x) = \log p(x|C_i) + \log P(C_i) \tag{6}$$

By looking at class posteriors, risks, and class discriminants we can easily see that class boundaries are the same (around $x = -11.57$ and $x = 3.54$). This is because we used the 0/1 loss for calculating risks. If we would use

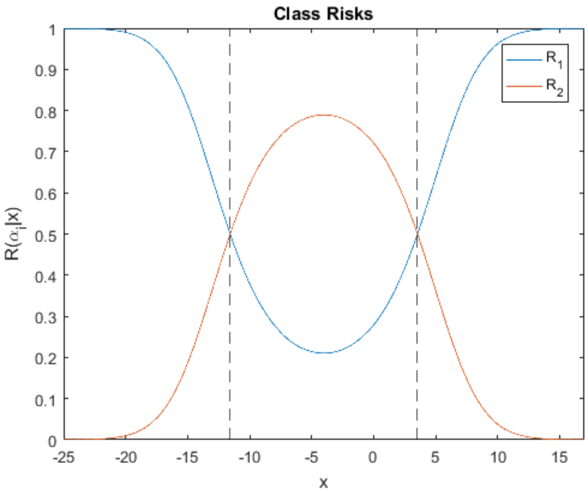

Fig. 4. Risks for choosing classes.

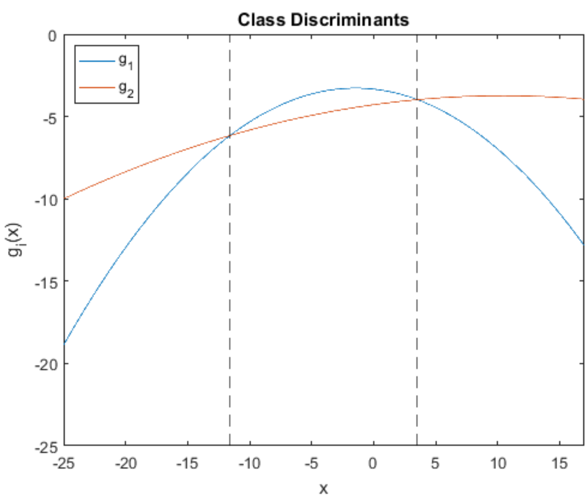

Fig. 5. Class discriminants.

asymmetric losses as in the next parts of the homework, class boundaries of risks and class discriminants would be different with respect to the symmetric case in this part. Class boundaries of them would always be equal to each other because $g_i(x) = \log(-R(\alpha_i|x))$. Class posteriors would always have the same class boundaries for all loss cases.

If class priors were assumed equal, class boundaries would be equal to x values at the intersections of class likelihoods. Because in that case, $P(C_i)$ values (also $p(x)$ values) in the calculations would be the same for both classes, only class likelihoods, $p(x|C_i)$, determine the class boundaries.

In order to find class boundaries analytically, the equations below were solved. Calculated class boundary values are the same as we found from plots. Therefore, the plot verifies our solution.

$$g_i(x) = \log(\frac{1}{\sqrt{2\pi}\sigma_i}) - \frac{(x-\mu_i)^2}{2\sigma_i^2} + \log P(C_i) \quad (7)$$

$$g_1(x) = g_2(x) \qquad g_1(x) - g_2(x) = 0 \quad (8)$$

$$\log(\frac{\sigma_2}{\sigma_1}) + \frac{(x-\mu_2)^2}{2\sigma_2^2} - \frac{(x-\mu_1)^2}{2\sigma_1^2} + \log(\frac{P(C_1)}{P(C_2)}) = 0 \quad (9)$$

$$0.867 + \frac{(x-10.484)^2}{2\cdot(10.029)^2} - \frac{(x+1.452)^2}{2\cdot(4.216)^2} - 0.406 = 0 \quad (10)$$

$$x \approx -11.57 \qquad x \approx 3.54 \quad (11)$$

For minimum risk, decision rule was found as the following expression.

$$\text{Choose } \alpha_i = \begin{cases} \alpha_1 & \text{if} -11.57 < x < 3.54 \\ \alpha_2 & \text{otherwise} \end{cases} \quad (12)$$

Using this decision rule, confusion matrices for training and validation sets were found as below. In these matrices, 1st row corresponds to actual class C_1 and 2nd row corresponds to actual class C_2. Also, 1st column corresponds to predicted class C_1 and 2nd column corresponds to predicted class C_2. Our classifier based on the decision rule above performs almost the same for both sets.

$$\text{Training Set Confusion Matrix} = \begin{bmatrix} 46 & 4 \\ 15 & 60 \end{bmatrix}$$

$$\text{Validation Set Confusion Matrix} = \begin{bmatrix} 44 & 6 \\ 14 & 61 \end{bmatrix}$$

B. Part B

In this part, we had a loss matrix as below.

$$\lambda = \begin{bmatrix} 0 & 0.5 \\ 1 & 0 \end{bmatrix} \quad (13)$$

Using this loss matrix, risks for choosing actions were calculated as below.

$$R(\alpha_1|x) = \lambda_{12}\cdot(1 - P(C_1|x)) = 0.5 - 0.5\cdot P(C_1|x) \quad (14)$$

$$R(\alpha_2|x) = \lambda_{21}\cdot(1 - P(C_2|x)) = 1 - P(C_2|x) \quad (15)$$

These risks for choosing actions were plotted in Fig. (6). As can be seen, the distance between class boundaries is higher than the distance between previous boundaries (around $x = -13.35$ and $x = 5.32$).

In this asymmetric loss case, choosing C_1 is less risky than choosing C_2 for a larger interval of x values compared to the symmetric loss case. This is because in asymmetric loss case, the loss due to predicting C_2 when the actual class is C_1 is more costly than the loss due to predicting C_1 when the actual class is C_2. Briefly, choosing C_1 is safer and less risky than the previous case for all x values.

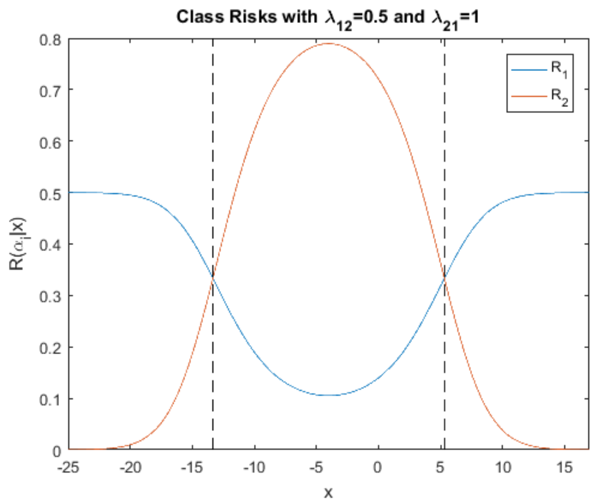

Fig. 6. Class risks with asymmetric loss.

New decision rule was found as follows for minimum risk.

$$\text{Choose } \alpha_i = \begin{cases} \alpha_1 & \text{if} -13.35 < x < 5.32 \\ \alpha_2 & \text{otherwise} \end{cases} \quad (16)$$

Using this new decision rule, confusion matrices for training and validation sets were found as below. In these matrices, 1st row corresponds to actual class C_1 and 2nd row corresponds to actual class C_2. Also, 1st column corresponds to predicted class C_1 and 2nd column corresponds to predicted class C_2 as before.

$$\text{Training Set Confusion Matrix} = \begin{bmatrix} 47 & 3 \\ 22 & 53 \end{bmatrix}$$

$$\text{Validation Set Confusion Matrix} = \begin{bmatrix} 48 & 2 \\ 17 & 58 \end{bmatrix}$$

As we expected, the number of predicted class C_1 increases and the number of predicted class C_2 decreases in this asymmetric loss case because λ_{12} is lower than λ_{21}. This increases the classification performance among the samples with actual class C_1. However, this also increases the number of the wrong predictions (C_1) among the samples with actual class C_2.

C. Part C

In this part, we had an extra action of rejection with a loss of $\lambda_3 = 0.2$. Also we had the asymmetric loss introduced in the previous part. Using these losses, risks for choosing actions were calculated as below. Here, α_3 corresponds to the action of rejection.

$$R(\alpha_1|x) = \lambda_{12} \cdot (1 - P(C_1|x)) = 0.5 - 0.5 \cdot P(C_1|x) \quad (17)$$

$$R(\alpha_2|x) = \lambda_{21} \cdot (1 - P(C_2|x)) = 1 - P(C_2|x) \quad (18)$$

$$R(\alpha_3|x) = \lambda_3 \cdot (P(C_1|x) + P(C_2|x)) = \lambda_3 \quad (19)$$

These risks for choosing actions were plotted in Fig. (7). As can be seen, the number of class boundaries increased from 2 to 4 because of the new rejection action (around $x = -14.83$, $x = -10.31$, $x = 2.27$ and $x = 6.79$). In this case, the rejection was less risky than the actions for choosing the class C_1 and C_2 in some regions so new class boundaries were needed.

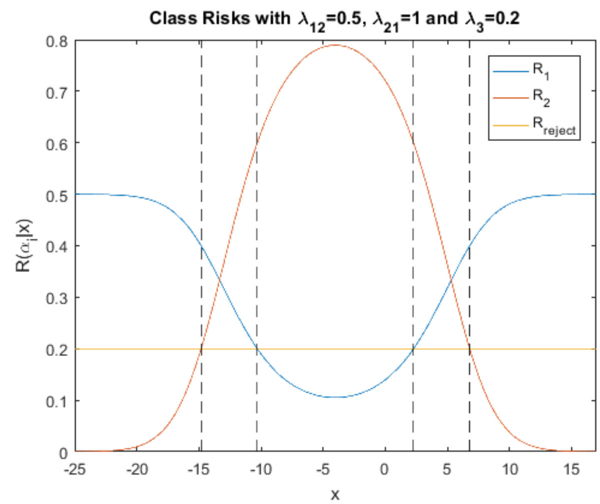

Fig. 7. Class risks with both asymmetric loss and rejection loss.

New decision rule was found as follows for minimum risk.

$$\text{Choose } \alpha_i = \begin{cases} \alpha_2 & \text{if} \quad x < -14.82 \\ \alpha_3 & \text{if} \quad -14.82 \leq x < -10.30 \\ \alpha_1 & \text{if} \quad -10.30 \leq x < 2.28 \\ \alpha_3 & \text{if} \quad 2.28 \leq x < 6.8 \\ \alpha_2 & \text{if} \quad 6.8 \leq x \end{cases} \quad (20)$$

Using this new decision rule, confusion matrices for training and validation sets were found as below. In these matrices, 1st row corresponds to actual class C_1 and 2nd row corresponds to actual class C_2. In addition, 1st column corresponds to predicted class C_1, 2nd column corresponds to predicted class C_2 and 3rd column corresponds to rejection.

$$\text{Training Set Confusion Matrix} = \begin{bmatrix} 41 & 2 & 7 \\ 11 & 44 & 20 \end{bmatrix}$$

$$\text{Validation Set Confusion Matrix} = \begin{bmatrix} 40 & 0 & 10 \\ 11 & 53 & 11 \end{bmatrix}$$

As we expected, the number of wrong decisions (misclassifications) decreases for both classes in this case because of rejection. The action of rejection is less risky than the actions for choosing the class C_1 and C_2 in some regions so it is preferred for some x values. The rejection action decreases misclassifications among both classes but it also decreases the number of true classifications for both classes. This is because for this case, even if the action of choosing C_1 is less risky than the action of choosing C_2 (or vice versa), it must always be less risky than the rejection action to be the chosen action. Otherwise, rejection will always be favored and this decreases the number of predictions as C_1 and C_2 among actual classes.

D. Part D

We should again look at the confusion matrix below which was found in the Part A. As before, 1st row corresponds to actual class C_1 and 2nd row corresponds to actual class C_2. Also, 1st column corresponds to predicted class C_1 and 2nd column corresponds to predicted class C_2. For this part, we can say C_1 is the positive (+1) and C_2 is the negative (-1).

$$\text{Training Set Confusion Matrix } = \begin{bmatrix} 46 & 4 \\ 15 & 60 \end{bmatrix}$$

$$\text{Validation Set Confusion Matrix } = \begin{bmatrix} 44 & 6 \\ 14 & 61 \end{bmatrix}$$

Then desired parameters for training set were calculated as below.

$$\mathrm{TN} = 60 \quad \mathrm{TP} = 46 \quad \mathrm{FN} = 4 \quad \mathrm{FP} = 15 \tag{21}$$

$$\text{Sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = \frac{46}{46+4} = 0.92 \tag{22}$$

$$\text{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = \frac{60}{60+15} = 0.8 \tag{23}$$

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} = \frac{46}{46+15} \approx 0.75 \tag{24}$$

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}} = \frac{60}{60+4} \approx 0.94 \tag{25}$$

$$\text{Accuracy} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TN}+\mathrm{TP}+\mathrm{FN}+\mathrm{FP}} = \frac{106}{125} \approx 0.85 \tag{26}$$

Then desired parameters for validation set were calculated as below.

$$\mathrm{TN} = 61 \quad \mathrm{TP} = 44 \quad \mathrm{FN} = 6 \quad \mathrm{FP} = 14 \tag{27}$$

$$\text{Sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = \frac{44}{44+6} = 0.88 \tag{28}$$

$$\text{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = \frac{61}{61+14} \approx 0.81 \tag{29}$$

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} = \frac{44}{44+14} \approx 0.76 \tag{30}$$

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}} = \frac{61}{61+6} \approx 0.91 \tag{31}$$

$$\text{Accuracy} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TN}+\mathrm{TP}+\mathrm{FN}+\mathrm{FP}} = \frac{105}{125} = 0.84 \tag{32}$$

When asymmetric losses are used as in the part B, TP and FP increase while TN and FN decrease because choosing positive class has lower loss than choosing negative has in this case. This causes an increase in sensitivity and a decrease in specificity. If we assume that the rates of changes in TP, FP, TN and FN are equal, we would expect that PPV, NPV and accuracy values do not change in this case.

In addition to asymmetric losses, when rejection is used as in the part C, we expect that all the TN, TP, FN and FP values should decrease with respect to asymmetric loss case without rejection. On the contrary, all sensitivity, specificity, PPV, NPV and accuracy values should increase because in this case, our classifier does not label an input as C_1 or C_2 as long as choosing them is not sufficiently confident. When choosing them is not confident, it chooses rejection and one needs another evidences to predict the class of input confidently.

II. Question 2

In this question, we were supposed to perform least squares regression by fitting polynomials of order 0 to 9. For this purpose, normal equation below was used to find coefficients ($\mathbf{w}$) of polynomials

$$\mathbf{w} = (\mathbf{X^T X})^{-1}\mathbf{X^T r} \tag{33}$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_d^1 \\ 1 & x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & & & & \\ 1 & x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix} \tag{34}$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \quad \mathbf{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix} \tag{35}$$

In our case, we were given a univariate input $\mathbf{x} = [x^1 \quad x^2 \quad \cdots \quad x^N]^T$ and if the degree of polynomial which would be fitted is d, our features would be as follows: $x_0 = 1$, $x_1 = x$, $x_2 = x^2, \cdots, x_d = x^d$. Also, we were given an actual output curve ($r = x^3 - x + 1$). In addition, we were given three different outputs with different noise levels.

For each polynomial degree from 0 to 9, the normal equation was solved using respective $\mathbf{X}$ matrices to find coefficients. Then using these coefficients, fitted polynomials were found and errors were calculated for both noisy data and actual data using mean square error method. This was done for three different noise levels $\sigma_n = 0.1$, $\sigma_n = 0.3$ and $\sigma_n = 0.5$. Both the error with actual data and noisy data for different noise levels were plotted with respect to polynomial order in Fig. (8), (9) and (10).

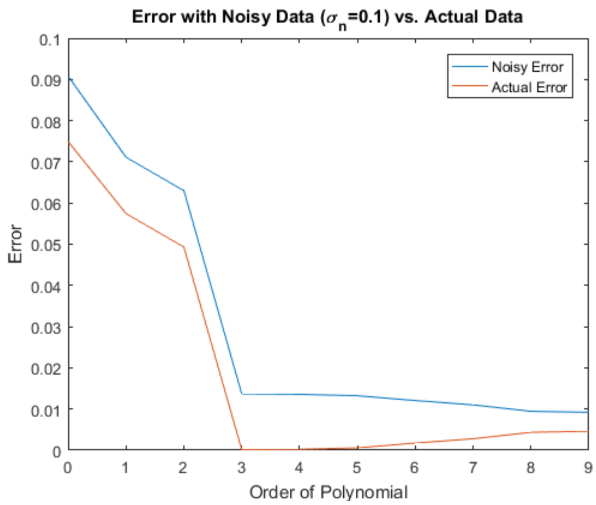

Fig. 8. Error with actual and noisy ($\sigma_n = 0.1$) data w.r.t. polynomial order.

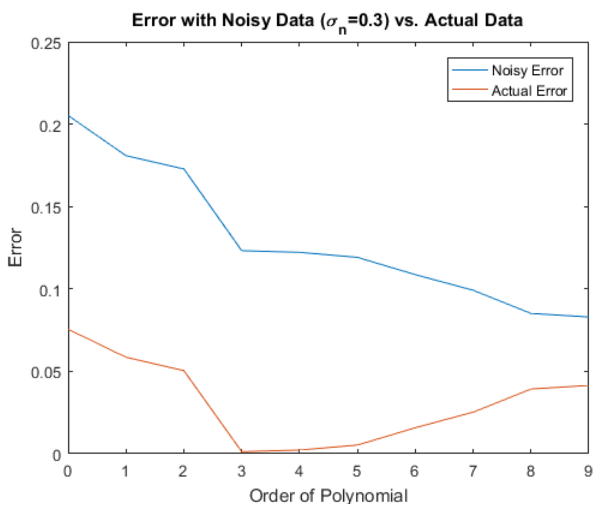

Fig. 9. Error with actual and noisy ($\sigma_n = 0.3$) data w.r.t. polynomial order.

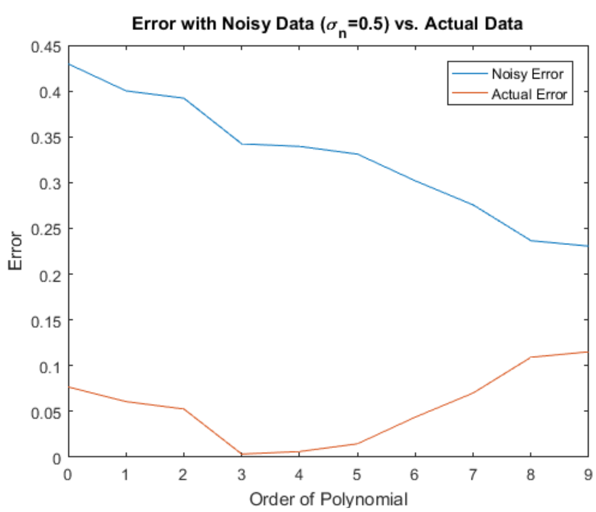

Fig. 10. Error with actual and noisy ($\sigma_n = 0.5$) data w.r.t. polynomial order.

As can be easily seen, error with actual data is always minimum when the fitted polynomial has the degree of 3. This is because actual data itself is a 3^{rd} degree polynomial. However with noisy data, the error can not be minimum at the degree of 3. This is because the data is no longer a 3^{rd} degree polynomial with noise and it is much more complex. Therefore, error with noisy data monotonically decreases as the degree of fitted polynomial increases. Also, error with actual data is always lower than the error with noisy data because of the complexity of noisy data.

When we look at the plots with different noise levels, we can easily see that both errors increases for all orders as the noise level increases. With more noise, the data becomes much more complex and this increases the error of fitted polynomials. Now, we can inspect the fittings of polynomials with different order by looking at Fig. (11), (12), (13), (14), (15) and (16).

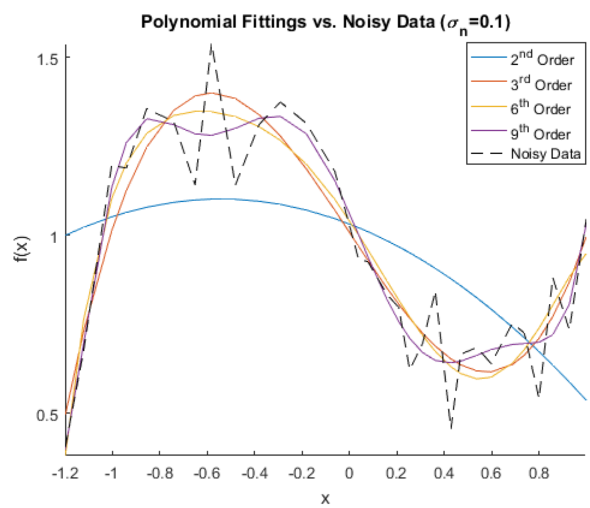

Fig. 11. Error with actual and noisy ($\sigma_n = 0.1$) data w.r.t. polynomial order.

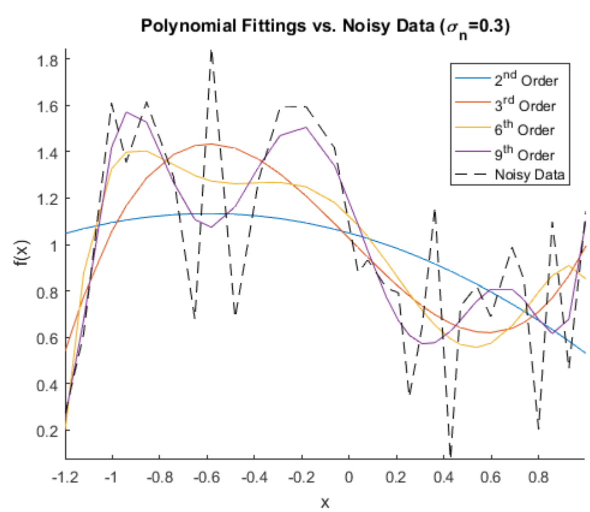

Fig. 12. Error with actual and noisy ($\sigma_n = 0.3$) data w.r.t. polynomial order.

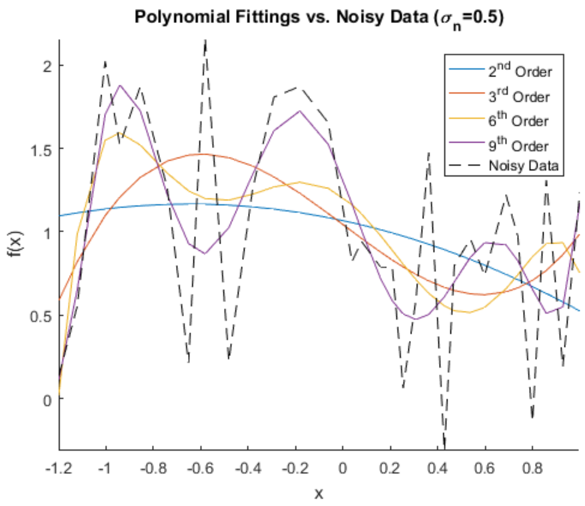

Fig. 13. Error with actual and noisy ($\sigma_n = 0.5$) data w.r.t. polynomial order.

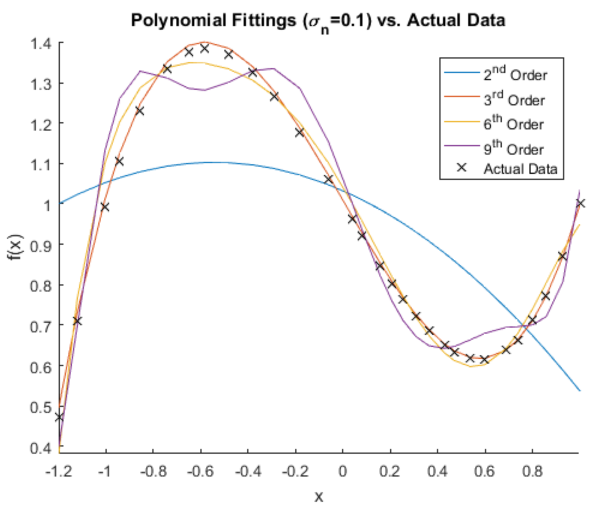

Fig. 14. Error with actual and noisy ($\sigma_n = 0.1$) data w.r.t. polynomial order.

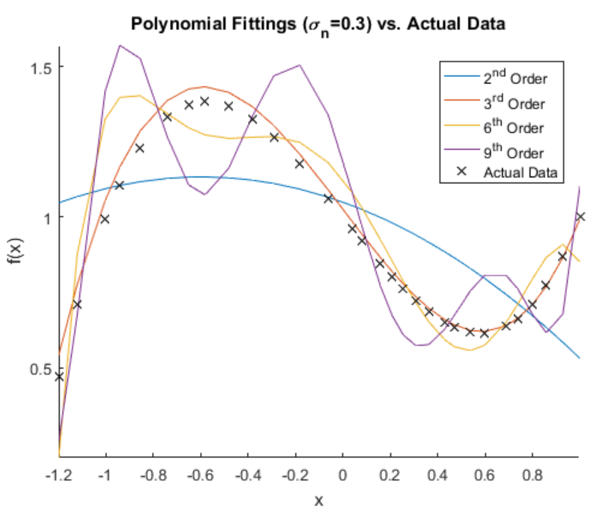

Fig. 15. Error with actual and noisy ($\sigma_n = 0.3$) data w.r.t. polynomial order.

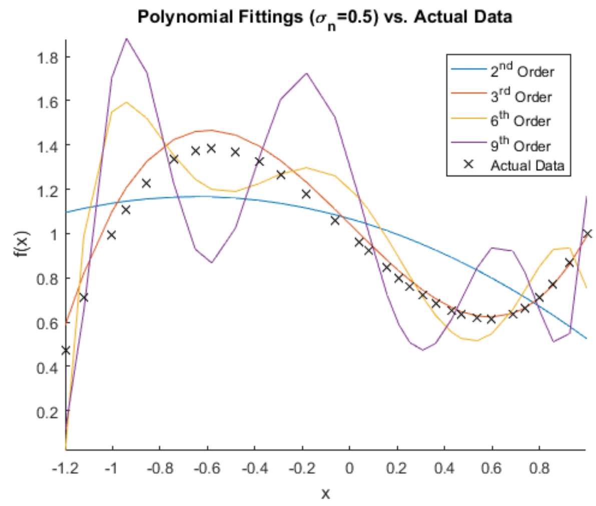

Fig. 16. Error with actual and noisy ($\sigma_n = 0.5$) data w.r.t. polynomial order.

As can be seen, 3^{rd} order polynomial is always the best fitting among all orders for actual curve. Also, fitting of 3^{rd} order onto actual data becomes worse as noise increases.

When we use look at the polynomial fittings of noisy data, it is clear that as the order increases, fitting becomes better. Again, fitting becomes worse as the noise level increases.

III. Appendix

A. Question 1

```
%% PART A
load('Data.mat');
p1=xtr_classification(:,2);
p1(p1<0)=2;
xtr_classification(:,2)=p1;
p2=xval_classification(:,2);
p2(p2<0)=2;
xval_classification(:,2)=p2;
%Class Priors
n=length(xtr_classification(:,2));
n1=sum(xtr_classification(:,2)==1);
Class_1_Prior=n1/n;
Class_2_Prior=(n-n1)/n;
%Class Likelihoods
Class_1=xtr_classification(1:n1,1);
Class_2=xtr_classification(n1+1:end,1);
mean1=mean(Class_1); std1=std(Class_1);
mean2=mean(Class_2); std2=std(Class_2);
x=-30:0.01:50;
Class_1_Likelihood=(1/(std1*sqrt(2*pi))) ...
    *exp((-(x-mean1).^2)/(2*std1^2));
Class_2_Likelihood=(1/(std2*sqrt(2*pi))) ...
    *exp((-(x-mean2).^2)/(2*std2^2));
figure
plot(x,Class_1_Likelihood,'DisplayName','C_1')
hold on
plot(x,Class_2_Likelihood,'DisplayName','C_2')
xlim([-30 50])
xlabel('x')
ylabel('p(x|C_i)')
title('Class Likelihoods')
legend('show')
%Evidence
Evidence=Class_1_Likelihood*Class_1_Prior ...
    +Class_2_Likelihood*Class_2_Prior;
```

```
figure
plot(x,Evidence)
xlabel('x')
ylabel('p(x)')
title('Evidence')

%Class Posteriors
Class_1_Posterior=(Class_1_Prior* ...
    Class_1_Likelihood)./Evidence;
Class_2_Posterior=(Class_2_Prior* ...
    Class_2_Likelihood)./Evidence;
figure
p1=plot(x,Class_1_Posterior)
hold on
p2=plot(x,Class_2_Posterior)
Boundary = get(gca,'YLim');
line([-11.565 -11.565],Boundary,'Color','k', ...
    'LineStyle','--');
line([3.535 3.535],Boundary,'Color','k', ...
    'LineStyle','--');
xlim([-25 17])
xlabel('x')
ylabel('P(C_i|x)')
title('Class Posteriors')
legend([p1 p2], 'C_1','C_2')
%Risks
Class_1_Risk=1-Class_1_Posterior;
Class_2_Risk=1-Class_2_Posterior;
figure
p3=plot(x,Class_1_Risk)
hold on
p4=plot(x,Class_2_Risk)
Boundary = get(gca,'YLim');
line([-11.565 -11.565],Boundary,'Color','k', ...
    'LineStyle','--');
line([3.535 3.535],Boundary,'Color','k', ...
    'LineStyle','--');
xlim([-25 17])
xlabel('x')
ylabel('R(\alpha_i|x)')
title('Class Risks')
legend([p3 p4], 'R_1','R_2')
%Discriminants
Class_1_Discriminant=log(Class_1_Likelihood) ...
    +log(Class_1_Prior);
Class_2_Discriminant=log(Class_2_Likelihood) ...
    +log(Class_2_Prior);
figure
p5=plot(x,Class_1_Discriminant)
hold on
p6=plot(x,Class_2_Discriminant)
Boundary = get(gca,'YLim');
line([-11.565 -11.565],Boundary,'Color','k', ...
    'LineStyle','--');
line([3.535 3.535],Boundary,'Color','k', ...
    'LineStyle','--');
xlim([-25 17])
ylim([-25 0])
xlabel('x')
ylabel('g_i(x)')
title('Class Discriminants')
legend([p5 p6], 'g_1','g_2')

%%Decision Rules and Confusion Matrices
% For minimum risk, C_1   -11.57<x<3.54
%                   C_2   else

TrClassification=zeros(125,1);
ValClassification=zeros(125,1);
for i=1:125
    if xtr_classification(i,1)>-11.57 && ...
        xtr_classification(i,1)<3.54
        TrClassification(i)=1;
    else
        TrClassification(i)=2;
    end

    if xval_classification(i,1)>-11.57 && ...
        xval_classification(i,1)<3.54
        ValClassification(i)=1;
    else
        ValClassification(i)=2;
    end
end

TrConfusion=confusionmat(xtr_classification ...
    (:,2),TrClassification)
ValConfusion=confusionmat(xval_classification ...
    (:,2),ValClassification)

%% PART B
%Risks
Class_1_Risk=0.5*(1-Class_1_Posterior);
Class_2_Risk=1-Class_2_Posterior;
figure
p1=plot(x,Class_1_Risk)
hold on
p2=plot(x,Class_2_Risk)
Boundary = get(gca,'YLim');
line([-13.345 -13.345],Boundary,'Color','k', ...
    'LineStyle','--');
line([5.315 5.315],Boundary,'Color','k', ...
    'LineStyle','--');
xlim([-25 17])
xlabel('x')
ylabel('R(\alpha_i|x)')
title('Class Risks with \lambda_1_2=0.5 and ...
    \lambda_2_1=1')
legend([p1 p2], 'R_1','R_2')

%%Decision Rules and Confusion Matrices
% For minimum risk, C_1   -13.35<x<5.32
%                   C_2   else

TrClassification=zeros(125,1);
ValClassification=zeros(125,1);
for i=1:125
    if xtr_classification(i,1)>-13.35 && ...
        xtr_classification(i,1)<5.32
        TrClassification(i)=1;
    else
        TrClassification(i)=2;
    end

    if xval_classification(i,1)>-13.35 && ...
        xval_classification(i,1)<5.32
        ValClassification(i)=1;
    else
        ValClassification(i)=2;
    end
end

TrConfusion=confusionmat(xtr_classification ...
    (:,2),TrClassification)
ValConfusion=confusionmat(xval_classification ...
    (:,2),ValClassification)

%% PART C
%Risks
Class_1_Risk=0.5*(1-Class_1_Posterior);
Class_2_Risk=1-Class_2_Posterior;
Reject_Risk=ones(1,length(x))*0.2;
figure
p1=plot(x,Class_1_Risk)
hold on
p2=plot(x,Class_2_Risk)
p3=plot(x,Reject_Risk)
Boundary = get(gca,'YLim');
```

```
line([-14.825 -14.825],Boundary,'Color','k', ...
    'LineStyle','--');
line([-10.305 -10.305],Boundary,'Color','k', ...
    'LineStyle','--');
line([2.275 2.275],Boundary,'Color','k', ...
    'LineStyle','--');
line([6.795 6.795],Boundary,'Color','k', ...
    'LineStyle','--');
xlim([-25 17])
xlabel('x')
ylabel('R(\alpha_i|x)')
title('Class Risks with \lambda_1_2=0.5, ...
    \lambda_2_1=1 and \lambda_3=0.2')
legend([p1 p2 p3], 'R_1','R_2','R_r_e_j_e_c_t')

%%Decision Rules and Confusion Matrices
% For minimum risk, C_2   x<-14.82
%                   R     -14.83<x<-10.30
%                   C_1   -10.31<x<2.28
%                   R     2.27<x<6.8
%                   C_2   x>6.79

TrClassification=zeros(125,1);
ValClassification=zeros(125,1);
for i=1:125
    if xtr_classification(i,1)<-14.82
        TrClassification(i)=2;
    elseif xtr_classification(i,1)<-10.30
        TrClassification(i)=3;
    elseif xtr_classification(i,1)<2.28
        TrClassification(i)=1;
    elseif xtr_classification(i,1)<6.8
        TrClassification(i)=3;
    else
        TrClassification(i)=2;
    end

    if xval_classification(i,1)<-14.82
        ValClassification(i)=2;
    elseif xval_classification(i,1)<-10.30
        ValClassification(i)=3;
    elseif xval_classification(i,1)<2.28
        ValClassification(i)=1;
    elseif xval_classification(i,1)<6.8
        ValClassification(i)=3;
    else
        ValClassification(i)=2;
    end
end

TrConfusion=confusionmat(xtr_classification ...
    (:,2),TrClassification)
ValConfusion=confusionmat(xval_classification ...
    (:,2),ValClassification)
```

B. Question 2

```
%% LOAD AND ASSIGN DATA
clc;
clear all;
load('Data.mat');
%% POLYNOMIAL FITTING AND ERRORS
x=x_regression(:,4);
y=x_regression(:,1:3);
y_actual=x.^3-x+1;
Associated_Error=zeros(10,3);
Actual_Error=zeros(10,3);
Theta_Mat={zeros(10,10),zeros(10,10), ...
    zeros(10,10)};
Fitted_Poly={zeros(10,30),zeros(10,30), ...
    zeros(10,30)};

for i=1:3
    X=ones(30,1);
    for j=1:10
        Theta=(((X'*X)\(X'))*y(:,i))';
        Theta_Mat{i}(j,1:length(Theta))=Theta;
        Fitted_Poly{i}(j,:)=Theta*X';
        X=[X x.^j];
        Associated_Error(j,i)=mean((y(:,i)- ...
            (Fitted_Poly{i}(j,:))').^2);
        Actual_Error(j,i)=mean((y_actual- ...
            (Fitted_Poly{i}(j,:))').^2);
    end
end
%% PLOTTING ERRORS
for i=1:3
    figure;
    plot(0:9,Associated_Error(:,i), ...
        'DisplayName','Noisy Error')
    hold on
    plot(0:9,Actual_Error(:,i), ...
        'DisplayName','Actual Error')
    xlabel('Order of Polynomial')
    ylabel('Error')
    title(['Error with Noisy Data ...
        (\sigma_n=' num2str(0.5-(i-1)*0.2) ...
        ') vs. Actual Data'])
    legend('show')
end
%% PLOTTING FITTED CURVES
for i=1:3
    figure
    hold on
    plot(x,Fitted_Poly{i}(3,:),'', ...
        'DisplayName','2^{nd} Order')
    plot(x,Fitted_Poly{i}(4,:), ...
        'DisplayName','3^r^d Order')
    plot(x,Fitted_Poly{i}(7,:), ...
        'DisplayName','6^t^h Order')
    plot(x,Fitted_Poly{i}(10,:), ...
        'DisplayName','9^t^h Order')
    scatter(x,y_actual,'k','x', ...
        'DisplayName','Actual Data')
    xlabel('x')
    ylabel('f(x)')
    title(['Polynomial Fittings (\sigma_n=' ...
        num2str(0.5-(i-1)*0.2) ') vs. Actual ...
        Data'])
    legend('show')
    axis tight
end

for i=1:3
    figure
    hold on
    plot(x,Fitted_Poly{i}(3,:),'', ...
        'DisplayName','2^n^d Order')
    plot(x,Fitted_Poly{i}(4,:), ...
        'DisplayName','3^r^d Order')
    plot(x,Fitted_Poly{i}(7,:), ...
        'DisplayName','6^t^h Order')
    plot(x,Fitted_Poly{i}(10,:), ...
        'DisplayName','9^t^h Order')
    plot(x,y(:,i),'k--','DisplayName','Noisy ...
        Data')
    xlabel('x')
    ylabel('f(x)')
    title(['Polynomial Fittings vs. Noisy ...
        Data (\sigma_n=' ...
        num2str(0.5-(i-1)*0.2) ')'])
    legend('show')
    axis tight
end
```