IITM-CS5691 : Pattern Recognition and Machine Learning — Release Date: Nov 1, 2023
Assignment 3 — Due Date: Nov 27, 2023, 23:59
(— Note: Assignment is doable by Nov 13 though!)

Roll No: CS23M034 — Name: Kushagra Jain
Collaborators (if any): Ravindra (CS23M054)
References/sources (if any):

- Use LaTeX to write-up your solutions (in the solution blocks of the source LaTeX file of this assignment), submit the resulting rollno.asst2.answers.pdf file at Crowdmark by the due date, and propery drag that pdf's answer pages to the corresponding question in Crowdmark (do this properly, otherwise we won't be able to grade!). (Note: **No late submissions** will be allowed, other than one-day late submission with 10% penalty or four-day late submission with 30% penalty.)
- Please upload to moodle a rollno.zip file containing three files: rollno.asst2.answers.pdf file mentioned above, and two code files for the programming question (rollno.ipynb file and rollno.py file). Do not forget to upload to Crowdmark your results/answers (including Jupyter notebook **with output**) for the programming question.
- Collaboration is encouraged, but all write-ups must be done individually and independently, and mention your collaborator(s) if any. Same rules apply for codes written for any programming assignments (i.e., write your own code; we will run plagiarism checks on codes).
- If you have referred a book or any other online material or LLMs (Large Language Models like ChatGPT) for obtaining a solution, please cite the source. Again don't copy the source *as is* - you may use the source to understand the solution, but write-up the solution in your own words (this also means that you cannot copy-paste the solution from LLMs!). Please be advised that *the lesser your reliance on online materials or LLMs* for answering the questions, *the more your understanding* of the concepts will be and *the more prepared you will be for the course exams*.
- Points will be awarded based on how clear, concise and rigorous your solutions are, and how correct your answer is. The weightage of this assignment is 12% towards the overall course grade.

1. (6 points) [A DIRECT/DISCRIMINANT APPROACH TO CLASSIFICATION] For the dataset below, we would like to learn a classifier, specifically a discriminant of the form: $\hat{y} = \text{sign}(wx)$ (assume $\text{sign}(u) = +1$ if $u \geqslant 0$, and -1 otherwise).

x	y
-1	-1
1	$+1$
20	$+1$

Let $z_i := z(x_i) := wx_i$. For a training dataset of size n, the parameter w of the classifier can be learnt by minimizing the

(L1) 0-1 loss function aka misclassification error $\sum_{i=1}^{n}(1 - \text{sign}(y_i z_i))/2$,

(L2) squared loss function $\sum_{i=1}^{n}(y_i - z_i)^2$, or

(L3) logistic loss function $\sum_{i=1}^{n} \log(1 + \exp(-y_i z_i))$.

(a) (2 points) The 0-1 loss function is the most intuitive choice to build a good classifier. What value of w will lead to such a good classifier for this dataset: $w = 0$ or $w = 1$?

Solution:

The 0-1 loss function, $\mathcal{L}_1(w)$, is a measure of misclassification. It assigns a loss of 1 for every misclassified point and 0 for correctly classified points. Mathematically, it can be expressed as:

$$\mathcal{L}_1(w) = \frac{1}{2}\sum_{i=1}^{n}(1 - \text{sign}(y_i z_i))$$

where $z_i = wx_i$ is the discriminant function for the i-th data point.

The goal is to find a parameter w that minimizes this loss. Let's consider the two candidate values, $w = 0$ and $w = 1$.

x	y	$z_i(w=0)$	$z_i(w=1)$	$y_i z_i(w=0)$	$y_i z_i(w=1)$
-1	-1	0	-1	0	1
1	$+1$	0	1	0	1
20	$+1$	0	20	0	20

The sign function, denoted as $\text{sign}(u)$, is defined as follows:

$$\text{sign}(u) = \begin{cases} +1, & \text{if } u \geqslant 0 \\ -1, & \text{if } u < 0 \end{cases}$$

Now, let's evaluate the expression for both values of w:

For $w = 0$:

$$\begin{aligned}\mathcal{L}_1(0) &= \frac{1}{2}\left[(1 - \text{sign}(0)) + (1 - \text{sign}(0)) + (1 - \text{sign}(0))\right] \\ &= \frac{1}{2}\left[(1-1) + (1-1) + (1-1)\right] \\ &= \frac{1}{2}(0) = 0\end{aligned}$$

For $w = 1$:

$$\begin{aligned}\mathcal{L}_1(1) &= \frac{1}{2}\left[(1 - \text{sign}(1)) + (1 - \text{sign}(1)) + (1 - \text{sign}(20))\right] \\ &= \frac{1}{2}\left[(1-1) + (1-1) + (1-1)\right] \\ &= \frac{1}{2}(0) = 0\end{aligned}$$

Both $w = 0$ and $w = 1$ result in the same value of the 0-1 loss function, which is 0. So, in this case, both $w = 0$ and $w = 1$ would lead to a classifier with the same performance according to the 0-1 loss function.

(b) (4 points) Between these values of w ($w = 0$ vs. $w = 1$), determine what value is preferred by the squared and logistic loss functions. Report the actual losses for these w values, and argue which loss function is better.
(Note: Optimizing squared loss is equivalent to applying linear regression methodology to solve this classification problem - did it work fine when there are outliers like $x = 20$ in the dataset?)

Solution:
Square Loss
Mathematically Square Loss function,can be expressed as:

$$\mathcal{L}_2(w) = \sum_{i=1}^{n} (y_i - z_i)^2$$

where $z_i = wx_i$ is the discriminant function for the i-th data point.
The goal is to find a parameter w that minimizes this loss. Let's consider the two candidate values, $w = 0$ and $w = 1$.

x	y	$z_i(w=0)$	$z_i(w=1)$
-1	-1	0	-1
1	$+1$	0	1
20	$+1$	0	20

Now, let's evaluate the expression for both values of w:
For $w = 0$:

$$\begin{aligned}\mathcal{L}_2(0) &= \left[(-1-0)^2 + (1-0)^2 + (1-0)^2\right] \\ &= \left[(-1)^2 + (1)^2 + (1)^2\right] \\ &= 3\end{aligned}$$

For $w = 1$:

$$\begin{aligned}\mathcal{L}_2(1) &= \left[(-1-(-1))^2 + (1-1)^2 + (1-20)^2\right] \\ &= \left[(0)^2 + (0)^2 + (-19)^2\right] \\ &= 361\end{aligned}$$

In $w = 0$ and $w = 1$ the loss value of $w = 0$ is less than $w = 1$ So, in this case,$w = 0$ would lead to a classifier with the optimal performance according to the square loss function.

Logistic Loss
Mathematically Logistic Loss function,can be expressed as:

$$\mathcal{L}_3(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i z_i))$$

where $z_i = wx_i$ is the discriminant function for the i-th data point.
The goal is to find a parameter w that minimizes this loss. Let's consider the two candidate values, $w = 0$ and $w = 1$.

x	y	$z_i(w=0)$	$z_i(w=1)$	$y_i z_i(w=0)$	$y_i z_i(w=1)$
-1	-1	0	-1	0	1
1	$+1$	0	1	0	1
20	$+1$	0	20	0	20

Now, let's evaluate the expression for both values of w:
For $w = 0$:

$$\begin{aligned}\mathcal{L}_3(0) &= [\log(1+\exp(-0)) + \log(1+\exp(-0)) + \log(1+\exp(-0))] \\ &= [\log(2) + \log(2) + \log(2)] \\ &= 3\log(2) = 2.08\end{aligned}$$

For $w = 1$:

$$\begin{aligned}\mathcal{L}_3(1) &= [\log(1+\exp(-1)) + \log(1+\exp(-1)) + \log(1+\exp(-20))] \\ &= [0.313 + 0.313 + 0] \\ &= 0.626\end{aligned}$$

In $w = 0$ and $w = 1$ the loss value of $w = 1$ is less than $w = 0$ .So, in this case,$w = 1$ would lead to a classifier with the optimal performance according to the logistic loss function.

So, we will use the Logistic Loss Function with $w = 1$ because it incurs less loss compared to the square loss with $w = 0$.

2. (6 points) [THINKING LOGISTIC-ALLY...] Consider the scenario in which a user maintains a dataset consisting of songs that he has downloaded over a period of time. He also tracks the likes (-1)/dislikes(+1) for each song along with a set of features X_1, X_2. X_1 is a binary variable that takes value 1 if the song is sung by his favorite singer and X_2 corresponds to song duration in minutes. This dataset with 10 datapoints is given below:

$$[X_1\ X_2] = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 5 & 10 & 13 & 2 & 3 & 5 & 2 & 10 & 10 & 3 \end{bmatrix}^T$$

$$y = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$$

   (a) (3 points) Train a logistic regression model on this dataset by performing the gradient descent algorithm steps by setting the initial weights to 0. No bias (intercept term) is required and the step size $\eta = 1$. Report the updated weights at the end of two iterations.

Solution: The logistic loss function is given by:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \mathbf{w}^T \mathbf{x}_i\right)\right)$$

The gradient of this loss function with respect to the weights (w) is:

$$\nabla\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} \sigma(-y_i \mathbf{w}^T \mathbf{x}_i)(-y_i \mathbf{x}_i)$$

Now, let's perform two iterations of the gradient descent algorithm. The update rule for the weights ($\mathbf{w}$) is given by:

$$\mathbf{w_{t+1}} \leftarrow \mathbf{w_t} - \eta \nabla\mathcal{L}(\mathbf{w})$$

where η is the step size.

Iteration 1

Given, $\eta = 1$ and $w_t = [0 \quad 0]$

w_0	w_1	x_1	x_2	y	$w^T X$	$-y_i w^T X$	$-y_i x_1$	$-y_i x_2$	$\sigma(-y_i w^T X)$
0	0	1	5	−1	0	0	1	5	0.5
0	0	0	10	−1	0	0	0	10	0.5
0	0	1	13	−1	0	0	1	13	0.5
0	0	0	2	−1	0	0	0	2	0.5
0	0	1	3	−1	0	0	1	3	0.5
0	0	0	5	1	0	0	0	−5	0.5
0	0	0	2	1	0	0	0	−2	0.5
0	0	1	10	1	0	0	−1	−10	0.5
0	0	0	10	1	0	0	0	−10	0.5
0	0	0	3	1	0	0	0	3	0.5

The updated weight after first iteration is :

$$\nabla\mathcal{L}(\mathbf{w}) = 0.5 \times \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1.5 \end{bmatrix}$$

$$\mathbf{w}_{t+1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 1 \times \begin{bmatrix} 1 \\ 1.5 \end{bmatrix} = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix}$$

So the updated weight is $\mathbf{w}_{t+1} = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix}$

Iteration 2
Given, $\eta = 1$ and $w_t = [-1 \quad -1.5]$

w_0	w_1	x_1	x_2	y	$w^T X$	$-y_i w^T X$	$-y_i x_1$	$-y_i x_2$	$\sigma(-y_i w^T X)$
-1	-1.5	1	5	-1	-8.5	-8.5	1	5	0.0002
-1	-1.5	0	10	-1	-15	-15	0	10	$3.06 * 10^{-7}$
-1	-1.5	1	13	-1	-20.5	-20.5	1	13	$1.25 * 10^{-9}$
-1	-1.5	0	2	-1	-3	-3	0	2	0.0474
-1	-1.5	1	3	-1	-5.5	-5.5	1	3	0.00407
-1	-1.5	0	5	1	-7.5	7.5	0	-5	0.9994
-1	-1.5	0	2	1	-3	3	0	-2	0.9526
-1	-1.5	1	10	1	-16	16	-1	-10	0.9999998
-1	-1.5	0	10	1	-15	15	0	-10	0.9999996
-1	-1.5	0	3	1	-4.5	4.5	0	3	0.989

The updated weight after first iteration is :

$$\nabla \mathcal{L}(\mathbf{w}) = 0.0002 * \begin{bmatrix} 1 \\ 5 \end{bmatrix} + 3.06 * 10^{-7} * \begin{bmatrix} 0 \\ 10 \end{bmatrix} + 1.25 * 10^{-9} * \begin{bmatrix} 1 \\ 13 \end{bmatrix} + 0.0474 * \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

$$+0.00407 * \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 0.9994 * \begin{bmatrix} 0 \\ -5 \end{bmatrix} + 0.9526 * \begin{bmatrix} 0 \\ -2 \end{bmatrix} + 0.9999998 * \begin{bmatrix} -1 \\ -10 \end{bmatrix}$$

$$+0.9999996 * \begin{bmatrix} 0 \\ -10 \end{bmatrix} + 0.989 * \begin{bmatrix} 0 \\ 3 \end{bmatrix} = \begin{bmatrix} -0.9958 \\ -29.76 \end{bmatrix}$$

$$\mathbf{w}_{t+1} = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix} - 1 \times \begin{bmatrix} -0.9958 \\ -29.76 \end{bmatrix} = \begin{bmatrix} -0.0042 \\ 28.26 \end{bmatrix}$$

So, the updated weight is $\begin{bmatrix} -0.0042 \\ 28.26 \end{bmatrix}$.

(b) (1 point) What will be the prediction for a new song with features [0, 20] using the trained logistic regression model?

Solution: To predict the value for a new song, we utilize the trained logistic regression model with parameters given as $\mathbf{W} = \begin{bmatrix} -0.0042 \\ 28.26 \end{bmatrix}$. The prediction is determined by evaluating the logistic function, also known as the sigmoid function, on the dot product of the input features and the model weights.
The logistic function is defined as:

$$\text{Prediction} = \begin{cases} +1, & \text{if } \sigma(\mathbf{W}^\mathsf{T}\mathbf{X}) > 0.5 \\ -1, & \text{if } \sigma(\mathbf{W}^\mathsf{T}\mathbf{X}) < 0.5 \\ \text{don't care}, & \text{if } \sigma(\mathbf{W}^\mathsf{T}\mathbf{X}) = 0.5 \end{cases}$$

Now, for the given input features $\mathbf{X} = \begin{bmatrix} 0 \\ 20 \end{bmatrix}$, we compute $\mathbf{W}^\mathsf{T}\mathbf{X} = (-0.0042 \times 0) + (28.26 \times 20) = 565.2$. Substituting this into the logistic function:

$$\sigma(\mathbf{W}^\mathsf{T}\mathbf{X}) = \frac{1}{1 + \exp^{-565.2}} \approx 1$$

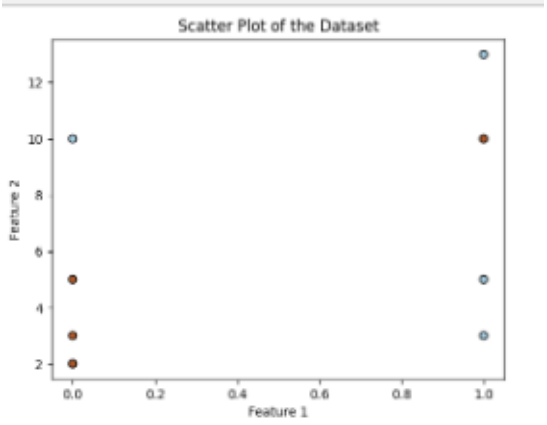
Hence, the data point will be classified to the **dislike** category.

(c) (2 points) Discuss if logistic regression is a good choice for addressing this specific problem. If not, what are other better options?

Solution:

Here, we are assuming a bias of zero, meaning that the decision boundary will always pass through the origin. In the provided figure, there is no straight line that can effectively separate both classes, even after multiple iterations. Consequently, our solution will not converge to a specific value. Therefore, logistic regression may not be a suitable choice for this scenario.

A better alternative could be using techniques like **Kernel Logistic Regression** or **Support Vector Machine (SVM)**, which can handle non-linear decision boundaries and may yield better results for the given dataset.



.

3. (12 points) [SVM'S TO THE RESCUE] A Gaussian or Radial Basis Function (RBF) kernel with inverse width $k > 0$ is

$$K(u, v) = e^{-k||u-v||^2}.$$

Below figures show decision boundaries and margins for SVMs learned on the exact same dataset. Parameters used for the different runs are as follows:

(i) Linear Kernel with C = 1

(ii) Linear Kernel with C = 10

(iii) Linear Kernel with C = 0.1

(iv) RBF Kernel with $k = 1$, C = 3

(v) RBF Kernel with $k = 0.1$, C = 15

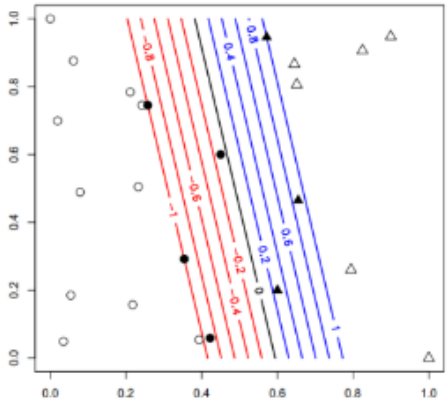
(vi) RBF Kernel with $k = 10$, C = 1

Find out which figure plot would have resulted after each run mentioned above. Justify your answer.

In these plots, circles are Class 1, triangles are Class 2, and solid points are support vectors.

Solution:

(I) Linear Kernel with C = 1

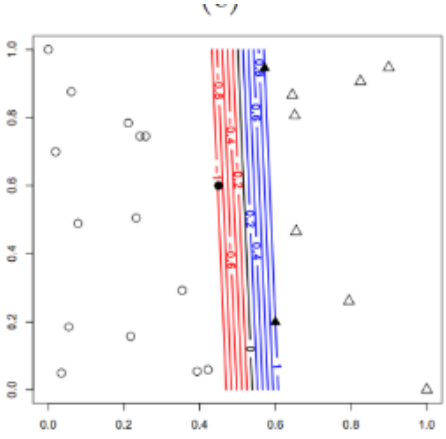


(b)

.

Justification : For the linear kernel decision margin will be the straight line so in the question image (b),(f) and (c) is having a decision margin as a straight line. The distance of support vector from the decision margin and the misclassification is based on the model complexity as model complexity is c=1 is given ,and the other 2 model complexity is c=10 and c=0.1 so the distance will not be that much greater and will not be that much small and this is given by image (b).

(II) Linear Kernel with C = 10

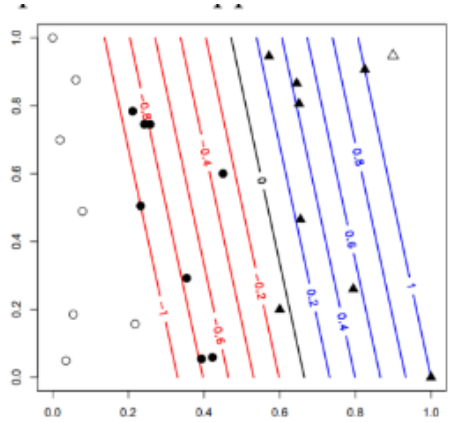


(f)

.

Justification : This is also linear kernel so the decision margin will be the straight line so in the provided figure (b),(f) and (c) is having a decision margin as a straight line. The distance of support vector from the decision margin and the misclassification is based on the model complexity as model complexity is c=10 is given ,and the other 2 model complexity is c=1 and c=0.1 so the distance between the decision boundary and the support vector is minimum and the misclassifcation will also be less which is given by figure number (f).

(III) Linear Kernel with C = 0.1

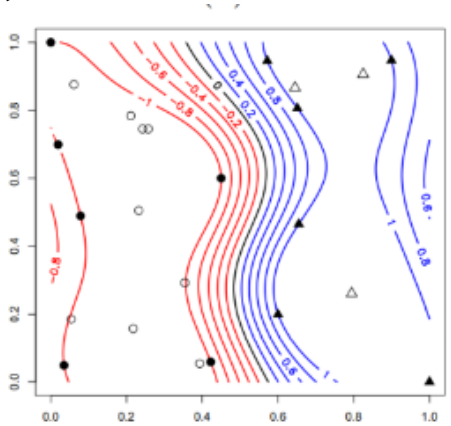


(c)

.

Justification : This is also linear kernel so the decision margin will be the straight line so in the provided figure (b),(f) and (c) is having a decision margin as a straight line. The distance of support vector from the decision margin and the misclassification is based on the model complexity as model complexity is c=0.1 is given ,and the other 2 model complexity is c=1 and c=10 so the distance between the decision boundary and the support vector is maximum and the misclassification will also be more which is given by figure number (c).

(IV) RBF Kernel with $k = 1$**, C = 3**

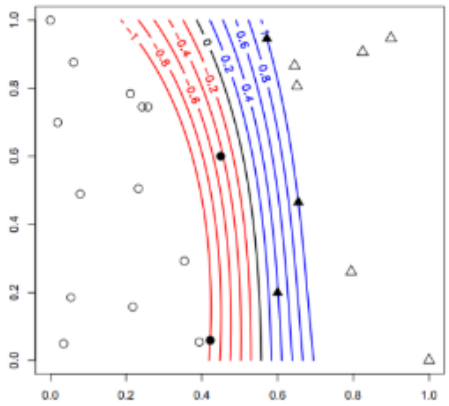


(e)

.

Justification : We are using RBF Kernel so the decision margin will be non linear so in the provided figure (e),(d) and (a) is having a decision margin non linear. RBF is based on 2 parameter first is K which is called gamma which represent flexibility(detailing) and another is C which is model complexity. Here K=1 and C=3 is given, So the figure which having normal flexibility and less misclassification point which is given by figure (e).

(V) RBF Kernel with $k = 0.1$**, C = 15**

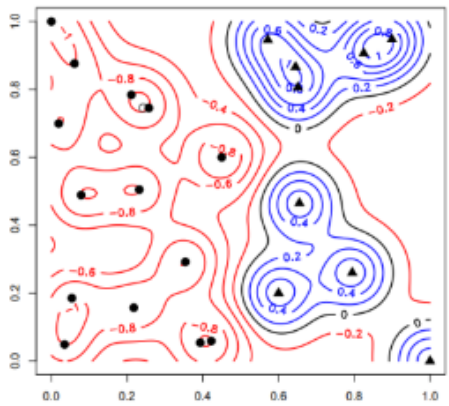


(d)

.

Justification : We are using RBF Kernel so the decision margin will be non linear so in the provided figure (e),(d) and (a) is having a decision margin non linear. RBF is based on 2 parameter first is K which is called gamma which represent flexibility(detailing) and another is C which is model complexity. Here K=0.1 and C=15 is given, So the figure which having less flexibility and very less misclassification point which is given by figure (d).

(VI) RBF Kernel with $k = 10$**, C = 1**



(a)

.

Justification : We are using RBF Kernel so the decision margin will be non linear so in the provided figure (e),(d) and (a) is having a decision margin non linear. RBF is based on 2 parameter first is K which is called gamma which represent flexibility(detailing) and another is C which is model complexity. Here K=10 and C=1 is given, So the figure which having more flexibility(detailed) and some misclassification point which is given by figure (a).

4. (12 points) [GUESS-TIMATING BIAS AND VARIANCE] You are given a dataset consisting of 100 datapoints in this folder. You have to fit a polynomial ridge regression model to this data.

   As seen in class, a model's error can be decomposed into bias, variance, and noise. A "learning curve" provides an opportunity to determine the bias and variance of machine learning models, and to identify models that suffer from high bias (underfitting) or high variance (overfitting). The "learning curve" typically shows the training error and validation/testing error on the y-axis and the model complexity on the x-axis.

   (a) (2 points) Read the last Section (Section 4) on "Bias and Variance in practice" in this document, and summarize briefly how you will heuristically find whether your model suffers from (i) high bias, or (ii) high variance, using only the train and validation/test errors of the model.

Solution: To heuristically identify whether a model suffers from high bias or high variance using train and validation/test errors, you can follow these general guidelines:
High bias (Underfitting):

- **Training error:** When the training error is high, this means that the model failed to capture the underlying pattern.
- **Validation/Test Error:** A high validation/test error indicates that the model is not good for new data.
- **Relationship:** Training and validation/testing errors are both high.

High variance (Overfitting):

- **Training error:** If the Training error is Low (model based on training data fits well), but validation/testing error is high, indicating overfitting.
- **Validation/Test Error:** If the model performs well on training data but not well on new data, this means the model is too hard and has captured noise in training.
- **Relationship:** There is a significant difference between training and validation/testing errors.

(b) (2 points) Start with the code for polynomial regression from the tutorial (the code without inbuilt package functions in in Tutorial #8), and add quadratic regularization functionality to the code. That is, your code should do polynomial regression with quadratic regularization that takes degree d and regularization parameter λ as input. Do not use any inbuilt functions from python packages (except for plotting functions and functions to compute polynomial features for each datapoint).

Solution:

```
# Cell type : CodeWrite
# write the function for Polynomial regression with quadratic regularization here.

Data=pd.read_csv("bayes_variance_data.csv")
x = Data.iloc[:, :-1].values
y = Data.iloc[:, -1].values
def designMatrix(data,degree):

    matrix=np.zeros((len(data),degree+1))
    for i in range(degree+1):
        matrix[:,i]=data[:,0]**i

    return matrix


def polyregression(x,y,lambda_l,degree):


    basis_function=designMatrix(x,degree)
    I = np.identity(degree+1)
    weights = np.linalg.inv(basis_function.T @ basis_function + lambda_l*I) @ basis_function.T @ y

    return weights

```

(c) (3 points) Run your code on the provided dataset for degree $d = 24$ and each λ in the set:

$$\{10^{-15}, 10^{-9}, 10^{-6}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{6}, 10^{9}, 10^{15}\}$$

i. Perform 5-fold cross-validation on the 100 data points (20 datapoints in each fold). For each validation fold, compute both training (4-folds-based) and validation (1-fold-based) errors using mean squared error measure.

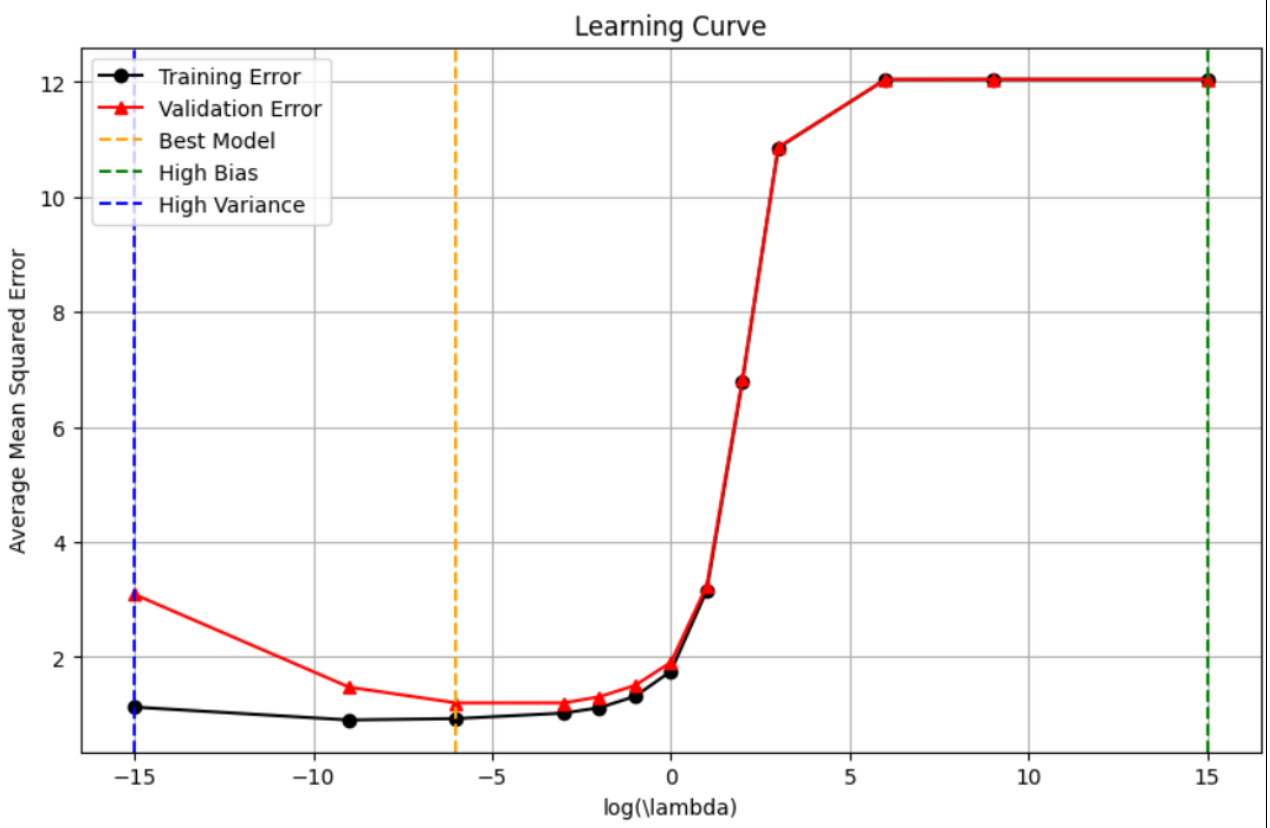
ii. Calculate the average training and validation errors across the 5 folds.

Solution:

```
Lambda : 1e-15 , Avg Training_Score : 1.1293072484655684, Avg Validation_Score : 3.0893479768818155
Lambda : 1e-09 , Avg Training_Score : 0.9049390557964335, Avg Validation_Score : 1.4763709462393968
Lambda : 1e-06 , Avg Training_Score : 0.9289095352448319, Avg Validation_Score : 1.2018605233987243
Lambda : 0.001 , Avg Training_Score : 1.0250164435942533, Avg Validation_Score : 1.2024482423646607
Lambda : 0.01 , Avg Training_Score : 1.1182713821628945, Avg Validation_Score : 1.3083494992307132
Lambda : 0.1 , Avg Training_Score : 1.316856603094962, Avg Validation_Score : 1.507711870235863
Lambda : 1 , Avg Training_Score : 1.7527687088435822, Avg Validation_Score : 1.908688609510601
Lambda : 10 , Avg Training_Score : 3.1536491522838106, Avg Validation_Score : 3.236568615625045
Lambda : 100 , Avg Training_Score : 6.795106529891221, Avg Validation_Score : 6.835281971815999
Lambda : 1000 , Avg Training_Score : 10.85622836351834, Avg Validation_Score : 10.865802820523331
Lambda : 1000000 , Avg Training_Score : 12.044458030522495, Avg Validation_Score : 12.044470085851863
Lambda : 1000000000 , Avg Training_Score : 12.045854156518427, Avg Validation_Score : 12.045854168576998
Lambda : 1000000000000000 , Avg Training_Score : 12.045855554286609, Avg Validation_Score : 12.045855554286623
```

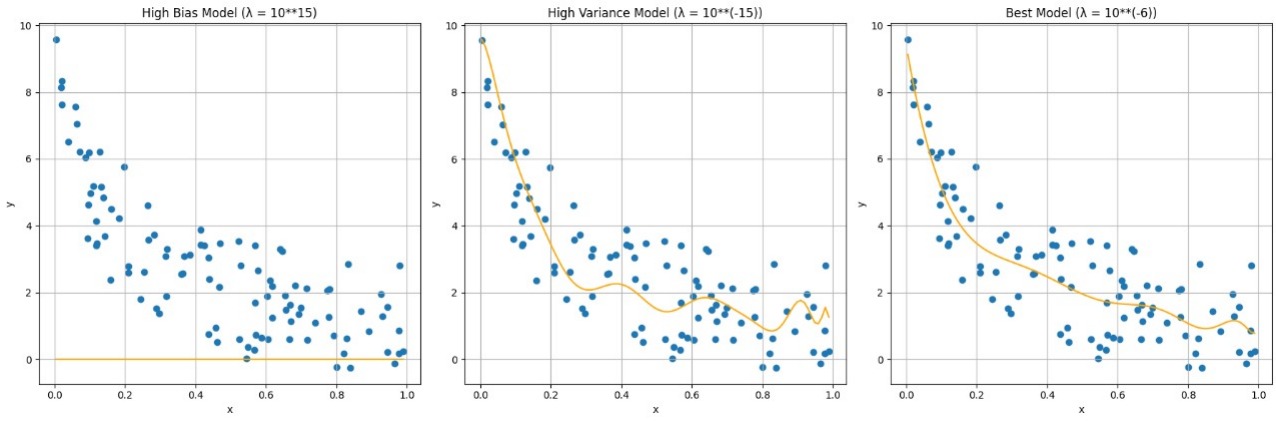
(d) (3 points) Construct a learning curve by plotting the average training and validation errors against the model complexity ($\log_{10} \lambda$). Based on this learning curve, identify the (i) model with the highest bias, (ii) model with the highest variance?, and (iii) the model that will work best on some unseen data.

Solution:


Learning Curve
Training Error
Validation Error
Best Model
High Bias
High Variance
Average Mean Squared Error
log(\lambda)
12
10
8
6
4
2
−15
−10
−5
0
5
10
15

(e) (2 points) Plot the fitted curve to the given data ($\hat{y}$ against x curve) for the three models reported in part (e), and superimposed with the training and validation datapoints for any one fold.

Solution:


High Bias Model (λ = 10**15)
High Variance Model (λ = 10**(-15))
Best Model (λ = 10**(-6))
x
y

Please use the template.ipynb file in the same folder to prepare your solution. Provide your results/answers in the pdf file you upload to Crowdmark named rollno.asst3.answers.pdf, and submit your pdf and code separately also in this moodle link. The pdf+code submitted should be a rollno.zip file containing three files: rollno.asst3.answers.pdf, rollno.ipynb file (including your code as well as the exact same results/plots uploaded to Crowdmark) and the associated rollno.py file.