

Roll No:CS23E001  
Collaborators (if any):  
References/sources (if any):

Name: Shuvrajeet Das

- Use  $\text{\LaTeX}$  to write up your solutions (in the solution blocks of the source  $\text{\LaTeX}$  file of this assignment), submit the resulting rollno.asst2.answers.pdf file at Crowdmark by the due date, and properly 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. The 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 to 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  $u \geq 0$ , 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 value of  $w$  will be 0.

- (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:**

The value of  $w$  will be 0.

The squared loss for  $w = 0$  is 12

The logistic loss for  $w = 0$  is 19.36

The squared loss for  $w = 1$  is 1175

The logistic loss for  $w = 1$  is 485165213.66

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 data points is given below:

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

$$y = [-1 \ -1 \ -1 \ -1 \ -1 \ 1 \ 1 \ 1 \ 1 \ 1]^T$$

- (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 given matrices are:

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

$$y = [-1 \ -1 \ -1 \ -1 \ -1 \ 1 \ 1 \ 1 \ 1 \ 1]^T$$

The gradient is:  $\sum_{i=1}^{10} \sigma(y_i w_i x_i) y_i x_i$

The new weights are  $\frac{-[-32]}{2} = [1.5 \ -1]$

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

**Solution:** The Prediction will be -1

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

**Solution:** Support vector Machine or Neural Network can be used.

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

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

The below figures show decision boundaries and margins for SVMs learned on the exact same dataset. The 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.

SVM Images/fig1.PNG

**Solution:** The answer to the following question:

- (i) Linear Kernel with  $C = 1$ : Figure (b)
- (ii) Linear Kernel with  $C = 10$ : Figure (f)
- (iii) Linear Kernel with  $C = 0.1$ : Figure (c)
- (iv) RBF Kernel with  $k = 1, C = 3$ : Figure (e)
- (v) RBF Kernel with  $k = 0.1, C = 15$ : Figure (d)
- (vi) RBF Kernel with  $k = 10, C = 1$ : Figure (a)

The Linear kernel makes a linear decision boundary.

Reason, The more the value of  $C$  the tighter the boundary gets without giving any relaxation to the points and it becomes a hard margin classifier and perfectly classifies the data without allowing any slackness.

The Gaussian kernel makes a curve decision boundary

Thus, if the  $k$  is very large, only the points within a certain distance can affect the predicting point. In other words, larger  $k$  tends to make a local classifier, smaller  $k$  tends to make a much more general classifier.

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:** (i) The model will suffer high bias if the train error and test error are quite high and decrease gradually throughout the training (ii) The model will suffer high variance if the training error is low but the test error is quite high

- (b) (2 points) Start with the code for polynomial regression from the tutorial (the code without in-built package functions 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  $\lambda$  as input. **Do not use any inbuilt functions from Python packages (except for plotting functions and functions to compute polynomial features for each data point).**

**Solution:**

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

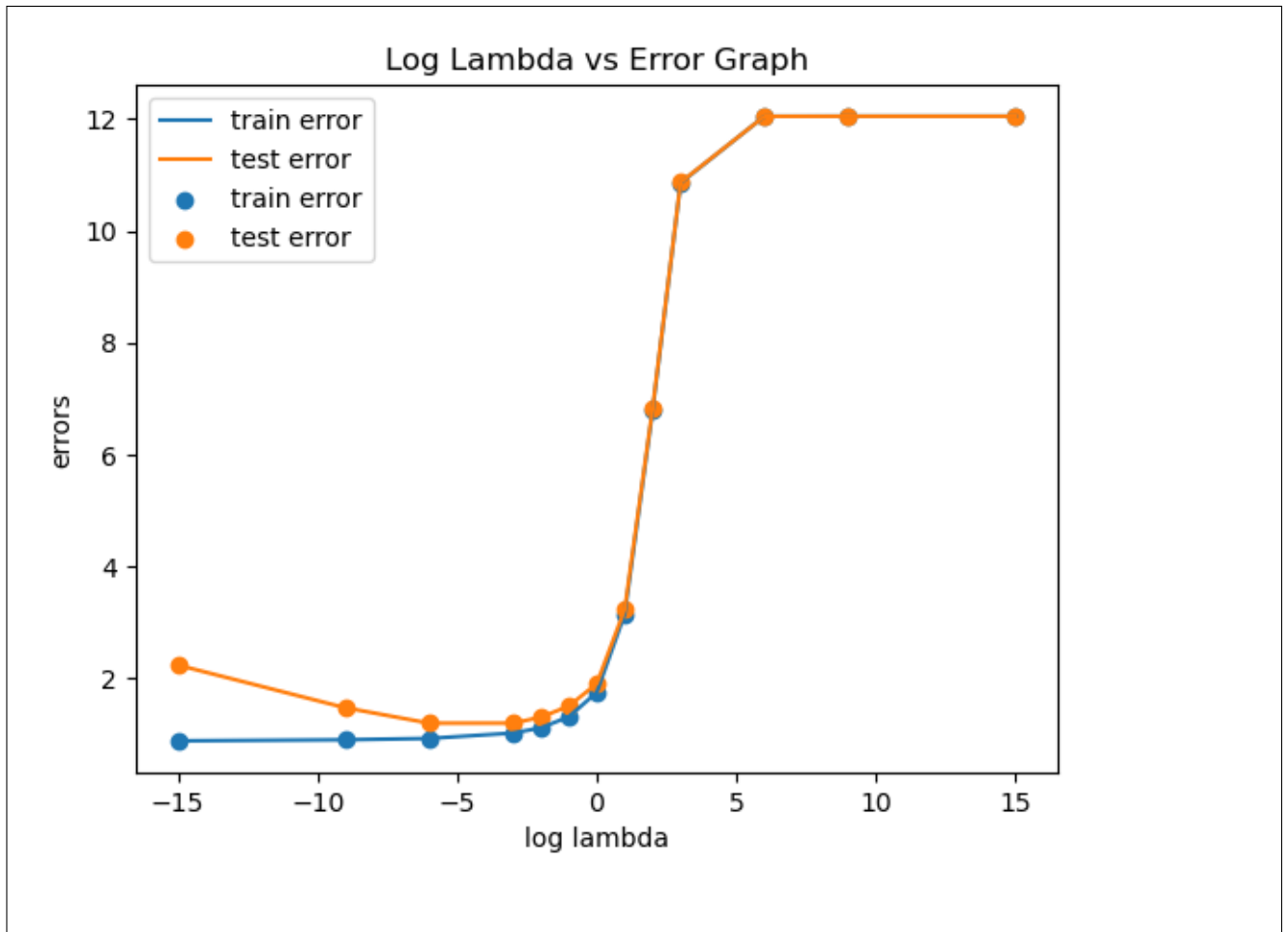
$$\{10^{-15}, 10^{-9}, 10^{-6}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^1, 10^2, 10^3, 10^6, 10^9, 10^{15}\}$$

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

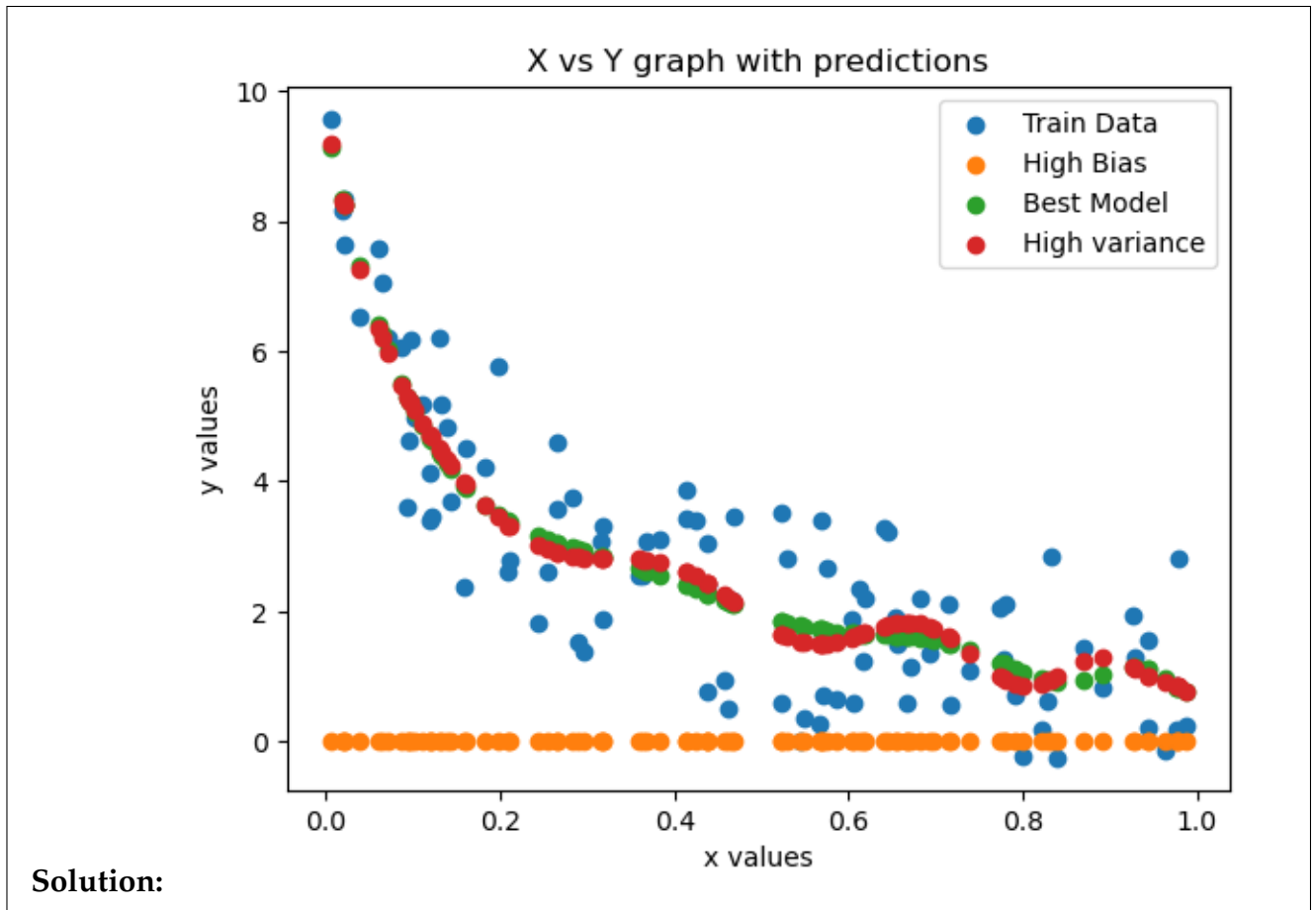
**Solution:** Look into the ipynb book

- (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:** The model with high variance is lambda: 1e-15  
The model with high bias is lambda: 10000000000000000.0  
The model with best Lambda: 1e-06



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



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.