PROJECT 1.2 LEARNING TO RANK USING LINEAR REGRESSION

Monisha Balaji
University of Buffalo
mbalaji@buffalo.edu

Abstract

Machine Learning provides people with a variety of problem-solving capabilities that progress without human intervention. The most widely referred problems are the classification and regression problems. Classification refers to identifying the input as one of the defined output classes to which it is of closest relevance to. The regression problem on the other hand focuses on predicting a real-valued output. It takes as input real or discrete values giving us a continuous output. The focus of this project is to perform regression to solve the LeToR, commonly known as the Learning to Rank problem. Linear Regression, one of the most commonly used learning algorithms, is used in this project. The goal is to perform linear regression by two different approaches: Closed-form solution and Stochastic Gradient Descent (SGD). This report records the working and performance of the two approaches with respect to Training, validation and testing accuracy of the models respectively.

1.INTRODUCTION

Linear Regression is a supervised learning algorithm that helps determine a linear relationship between the data and target variables. Considering a single input variable that produces a continuous output, we plot these values as a 2-dimensional graph with the input values along the x axis and the output values along the y axis. This depicts a cluster of points along the plane. The focus of linear regression is to produce a linear structure that tries to cover as many data points as possible. In other words, linear regression tries to achieve the best fit for all the data points. The line or the curve should be such that for every value of x, it should be closer to its corresponding value of y i.e. target value. When the model works on a single input variable or a single feature it is called Univariant linear regression. In this project we consider a Multivariant linear regression problem where each of the input variables have multiple features; in our case it includes 46 features. Here, we use the LeToR dataset for processing.

Learning to rank problem (LeToR) is an ideal task in the field of information retrieval. The problem is that the search engine should be able to retrieve information based on its relevance to the given query. For which the system should be capable of assigning relevance and ranking the retrieved data effectively. In our case, the system is given a dataset that contains the values of 46 features for different documents and also their corresponding relevance judgements. Relevance judgements are the user provided relevance labels for the documents in the dataset that help the system with learning. The system is trained on this dataset to understand the relationship between the features and the relevance judgements so that when a new relatively equivalent query is given the system can rank and retrieve accordingly.

2. PROCESSING OF DATASETS

Processing of the dataset is the initial stage of the project. As a preprocessing measure, the process starts with reading the contents from the data and target csv files and appending them to lists in the python code for processing. The features whose values i.e. the variance are zero are removed from the dataset while file reading. This reduces our feature count from 46 to 41. This is a vital step as the zero entries cause a hindrance in our processing when we compute a matrix inversion or determinant operations. The final step of preprocessing is partitioning datasets into Training, Validation and Testing sets. The Training data comprises of 80% of the overall dataset; Validation set comprises of 10% and Testing comprises of the last 10%. The partition should be such that there is no overlapping of data in any of the datasets. The model is then trained on the Training dataset and tested on the Validation and Testing sets. Now that the data is ready for processing, we perform linear regression by each of the two approaches for solving our problem. They are:

1. Closed-form solution
2. Stochastic Gradient Descent

2.1 CLOSED-FORM SOLUTION:

Closed-form solution works to minimize the errors in the linear regression equation. After plotting the points on the plane, linear regression produces an ‘hypothesis’ that helps form the linear curve. Now, we calculate the cost function (square of the difference between the predicted output and the actual output) with respect to the best fit line and use closed-form solution to try to minimize it. This approach provides a set of equations that helps to update the initial set of weights used and tries to compute the linear curve again. This is carried out recursively until the weights are adjusted such that the cost is minimum.

The equation that helps compute the target values is given by,

$$\mathbf{t} = \boldsymbol{w}^{\boldsymbol{T}} \ \boldsymbol{\Phi}(\boldsymbol{x})$$

where,
t is the Target values,
w^T is the Transpose of Weight matrix,
$\Phi(x)$ is the design matrix

The equation for the weights calculation is given by,

$$\boldsymbol{w}^* = (\boldsymbol{\lambda I} + \boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{t}$$

Where,
λI is the regularization term
Φ is the design matrix
t is the Target vector of the training set

The equation is constructed as such to help form a square matrix that helps with the matrix multiplication of the terms.

Initially, the system is trained with the Training dataset. So the above equation is adjusted such that the weight matrix is calculated from the design matrix and the values of the Target vector corresponding to the training set. Once the weights are adjusted for minimum cost, the target values for the Testing set is computed. Then, the accuracy and Root mean square error is computed to assess the performance of the system.

2.2 STOCHASTIC GRADIENT DESCENT:

SGD is the other approach that we use in this project to help minimize the cost function. The idea of gradient descent is that we differentiate the cost function with respect to the weights. This derivative gives us the slope of the curve and directs the change in path, along the U-bound gradient curve, to help reach the local minimum. Once the local minimum is reached, the slope becomes zero and there will not be any change in values. This helps determine that the minimum has been reached. In general, batch gradient descent is used where for each iteration of updating the weights, all the input dataset elements are processed. As in our case, considering at the Training dataset consists of 55699 input elements with 41 features each, each iteration will compute the derivation for all the input elements. As this is an exhaustive process, we use SGD where we use only one input point in one recursive call, to update the weights.

In both the approaches, we add an extra term to the equations to enhance the performance. This term is referred to as the regularization term (lambda). This value helps aid the performance of the approaches by helping them prevent overfitting into the data in which case the model is unable to generalize the solution. In our problem, we use E-RMS i.e. Root mean square error. This helps calculate the error between the predicted and actual. This is found by computing the square root of the sum of error differences divided by the total number of data.

3.WORKING WITH HYPERPARAMETERS

Hyper parameters are factors that eventually affect the performance of the system. These values are the ones that influence different important aspects of the model.

The hyper parameters that are of significant importance to us are:

- No. of Basis functions
- Learning Rate
- Regularization term (Lambda)
- Mean of the inputs in a cluster
- Variance of the clusters

3.1 BASIS FUNCTIONS:

Basis functions are functions that help handle the non-linearity between the input and target values. They consider the target values as a linear combination of a set of Basis functions where the number of Basis functions is subjective. In a general case, we can consider an identity basis function whose value is equal to the value of the input itself. There are a variety of functions

available that suit a variety of functionalities. In our case, we choose the Gaussian Radial Basis Function (RBF). Here, we use the concept of clustering to combine the target values into linear groups. We use K-Means algorithm. K-Means clustering is one of the most popularly used clustering algorithms. It uses the Euclidean distance between the points to cluster them together. Initially it randomly assigns centroids in the plane. Now, the distance of the points from the different centroids are calculated and it is then assigned to its nearest centroid. Then, the mean of all the points in a cluster is calculated to compute the new centroid. This is followed for all the clusters. Now again the distances of the points from the new computed centroids are calculated. This process of computing new centroids and cluster rearrangement happens recursively until all points are assigned a fixed cluster and the centroids are established. The number of clusters formed is equal to the number of Basis functions to be computed.
We compute the design matrix or the PHI matrix which is a matrix representation of all the Basis functions for each of the features put together. This comes every much in handy for the closed-form solution calculations. For our dataset, the design matrix is of the dimensions "Input x No. of Basis functions".

3.1.1 CLOSED FORM SOLUTION: Testing accuracy VS No. of Basis Functions

The correlation between the number of Basis functions (M) and the Testing accuracy of the model produced by Closed form solution is given below:

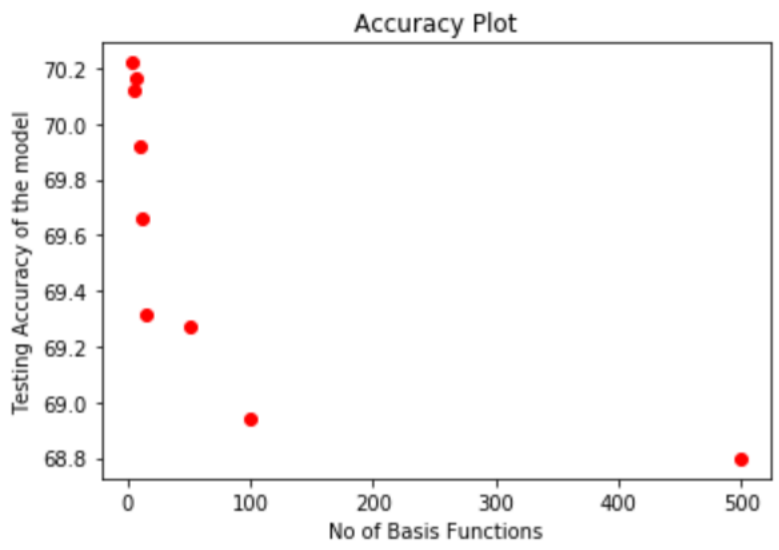

Fig 3.1: Testing Accuracy VS No of Basis Functions(M)

No of Basis Functions	Testing Accuracy
3	70.21979600632093
5	70.11923574199109
7	70.16233299813246
10	69.91811521333142
12	69.65953167648327
15	69.31475362735239
50	69.27165637121104
100	68.94124407412728
500	68.79758655365609

Table 3.1: Tabulation of the Basis Function Count and corresponding Testing accuracy values

The above graph depicts the relationship between the number of Basis functions and the testing accuracy of the model. Here the graph is plotted for different values of M ranging from 3 to 500. It portrays their corresponding testing accuracies. The drop in the points on this graph shows that as the number of Basis functions increase, the testing accuracy goes down. For values like 3 or 5, the accuracy is higher in comparison to the other values in the range. In our case, the count of the Basis functions is influenced by the number of clusters that are formed. This implies that smaller the number of clusters better is the accuracy of the model.

In general, there is no particular method to determine the number of clusters to be formed. It works on some heuristics. Here, increasing the number of clusters may affect the variance of the cluster region i.e. the spread of the region which in turn has an effect on the value of the Basis functions that might tend to affect the accuracy.

3.1.2 CLOSED FORM SOLUTION: No. of Basis Functions VS E-RMS

Next, the correlation between the number of Basis functions (M) and the Root mean square error (E-RMS) of the model produced by Closed form solution is given below:

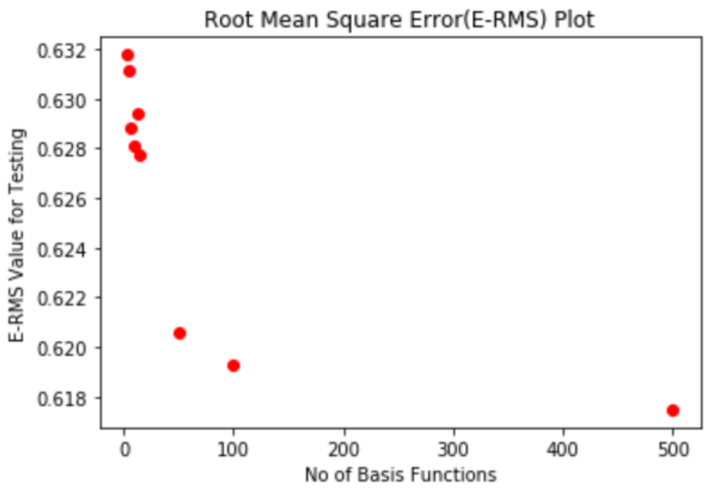

Fig 3.2: Number of Basis Functions VS E-RMS for Testing

No of Basis Functions	E-RMS For Testing
3	0.6317812714668385
5	0.6311308539478039
7	0.6288051919072434
10	0.6281031608753205
12	0.6293846550733718
15	0.6277580658083383
50	0.6205927045587823
100	0.6192613317793846
500	0.6174755514821012

Table 3.2: Tabulation of the Basis Function Count and corresponding E-RMS values

This graph shows the association of the Basis functions count with the E-RMS values. The graph is plotted for the same range of M and their corresponding values of testing E-RMS. Here again, for values like 3 or 5, the value of the error function is high. But as the number of Basis functions increase, the root mean square error decreases. In our case, with the increasing number of Basis functions(M), the value of E-RMS decreases and so does the accuracy. So choosing a large value for M in order to achieve a low error value means that we are compromising on the accuracy. Both accuracy and error value are significant aspects towards the performance of the system. So we need to be considerate about choosing a befitting value for M i.e. the number of Basis functions such that neither of the two factors are compromised.

3.2 MEAN AND VARIANCE OF THE CLUSTER:

Mean refers to the average of all the points in a cluster. A mean exists for every cluster. Variance is defined as the spread of the Basis functions. It can be referred to as the area spread of each of the clusters formed from K-Means clustering. Therefore, these two values play a role in the calculation of the Basis function values.

The equation of the Gaussian Radial Basis Function is given as,

$$\boldsymbol{\Phi_j(x) = \mathbf{exp}\,(-\tfrac{1}{2}\,(x - \mu_j)^T \Sigma_j^{-1}(x - \mu_j))}$$

Where,
μ_j is the Mean of the jth feature
$\Sigma_j^{-1}(x - \mu_j)$ represents the Covariance matrix

Covariance matrix is the matrix representation of the variance calculated for each of the features. This is a diagonal matrix where the diagonal represents the variance of the features. The other entries of this matrix are zero as the variance of one feature with respect to another is zero. This is represented as σ_{ij} where i and j represent two different features.

3.3 LEARNING RATE:

Learning rate is another very important hyper parameter. This plays a very important role in SGD. The derivative that is calculated is multiplied with the learning rate. So the final value is influenced by the learning rate. Learning rate refers to how big the steps of learning are i.e. the frequency of learning. If learning rate is too high, the steps are too large that the algorithm misses the local minimum i.e. it overshoots the minimum and it starts moving up again. If it is too small, the convergence is very slow. So an ideal value for the learning rate is very essential.

3.3.1 STOCHASTIC GRADIENT DESCENT: Learning rate VS Testing Accuracy

The correlation between the learning rate and the testing accuracy of the model predicted by stochastic gradient descent is given below:

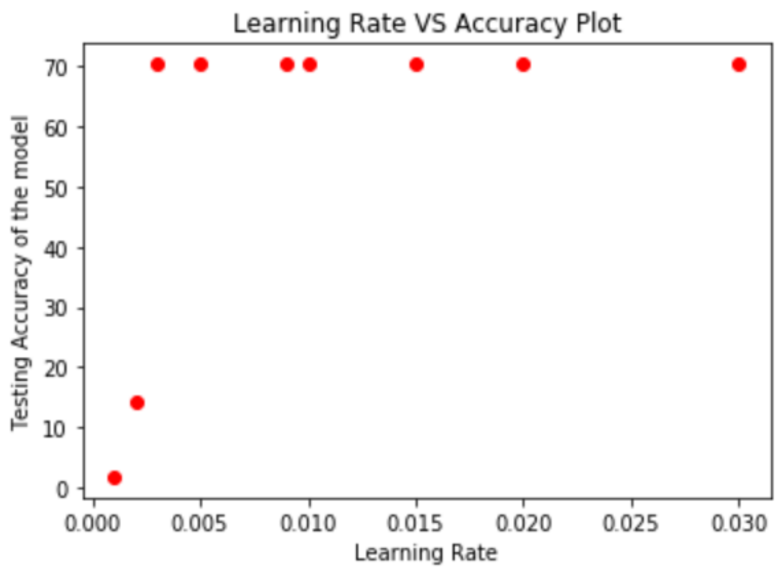

Fig 3.3: Learning Rate VS Testing Accuracy

The above graph represents the change in accuracy with respect to the change in learning rate. The learning rate is plotted along the x axis with its corresponding testing accuracy along the y axis. The graph is plotted from a range of 0.001 to 0.03. We can infer from the above graph that as the value of the learning rate increases, the accuracy increases. In general, the ideal values for learning rate lies between 0.001 to 0.1. Very high values of learning rate causes overshooting. In our case, as we have chosen a medial range as in not too high nor too low, it increases the accuracy of our model.

Learning Rate	Testing Accuracy
0.001	1.81008
0.002	14.35139
0.003	70.33472
0.005	70.33472
0.009	70.33472
0.01	70.33472
0.015	70.33472
0.02	70.33472
0.03	70.33472

Table 3.3: Tabulation of the Learning Rate and corresponding Accuracy

3.3.2 STOCHASTIC GRADIENT DESCENT: Learning rate VS E-RMS

The correlation between the learning rate and the testing accuracy of the model predicted by stochastic gradient descent is given below:

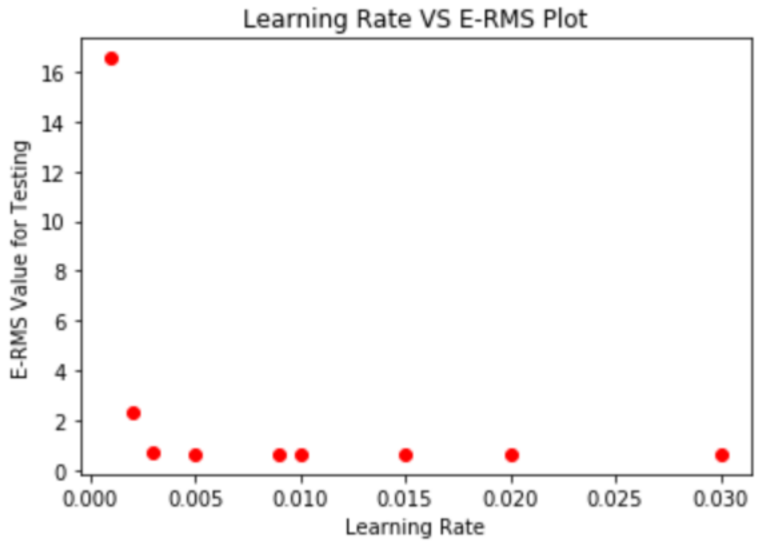

Fig 3.4: Learning Rate VS E-RMS

The above graph represents the change in root mean square error with respect to the change in learning rate. The learning rate is plotted along the x axis with its corresponding E-RMS along the y axis. The graph is plotted for the same range of 0.001 to 0.03. We can infer from the above graph that with increase in the values of the learning rate, the E-RMS values decrease. This shows that with an ideal learning rate, the error is low. But for very high values of the learning rate, the error would increase as there would be overfitting of data.

Learning Rate	E-RMS For Testing
0.001	16.54029741122756
0.002	2.30247776311019
0.003	0.6869613199721558
0.005	0.6453552749493607
0.009	0.6377919234273236
0.01	0.6369660736986316
0.015	0.6360072035481381
0.02	0.6360072035481381
0.03	0.6360072035481381

Table 3.4: Tabulation of the Learning Rate and corresponding E-RMS values

3.4 REGULARIZATION TERM (λ):

The regularization term plays an important role in maintaining the accuracies. It helps to prevent overfitting in the system. Overfitting causes the system to become unable to generalize the learning and makes it very much prone to the given input dataset. This has a major impact on the training, validation and testing accuracies. If the system overfits, it produces a high training accuracy but a low validation and testing accuracy. This is because the model is unable to generalize the solution to the new data. So this regularization term helps perform a generalized learning.

3.4.1 CLOSED FORM SOLUTION: Regularization term VS Testing Accuracy

The correlation between the regularization term and the testing accuracy of the model predicted by closed form solution is given below:

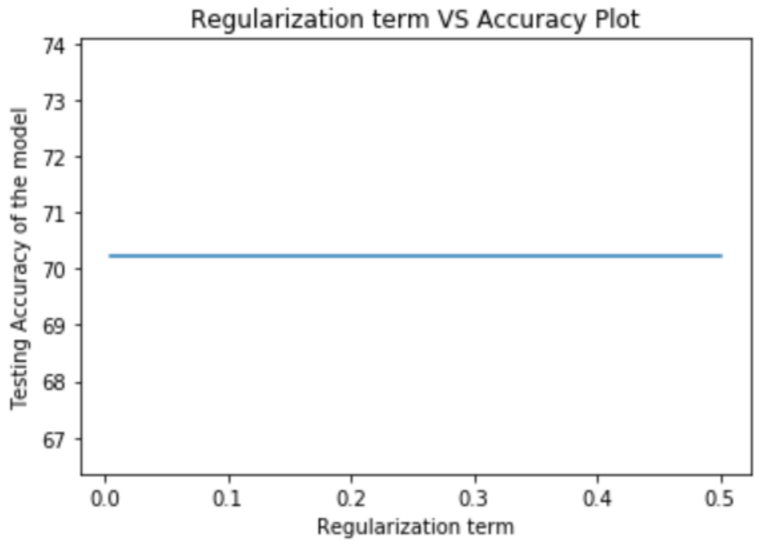

Fig 3.5: Regularization term VS Testing Accuracy

Regularization term	**Testing Accuracy**
0.005	70.21979600632093
0.05	70.21979600632093
0.09	70.21979600632093
0.25	70.21979600632093
0.5	70.21979600632093

Table 3.5: Tabulation of the Regularization term and corresponding Testing Accuracy

The above graph shows the relation between the regularization term and the testing accuracy of the system. This term is used to help avoid overfitting. The graph depicts a constant value for accuracy. The inference is that when we choose ideal values for other hyper parameters such as learning rate, no. of basis functions etc. such that the system does not overfit, in such cases the accuracy remains constant any value of lambda.

3.4.2 CLOSED FORM SOLUTION: Regularization term VS E-RMS

Regularization term	**E-RMS For Testing**
0.005	0.6317829556957587
0.05	0.6317820612440738
0.09	0.6317812714668385
0.25	0.6317781618544113
0.5	0.6317734598785693

Table 3.6: Tabulation of the Regularization term and E-RMS

The above table depicts the values of the regularization and its corresponding E-RMS values. Again, this value is a constant for varying values of lambda. This remains constant as the system is running on other hyper parameter values that do not involve overfitting. So, the value of lambda does not have a signification effect on the E-RMS.

3.4.3 STOCHASTIC GRADIENT DESCENT: Regularization term VS Testing Accuracy

The correlation between the regularization term and the testing accuracy of the model predicted by stochastic gradient descent is given below:

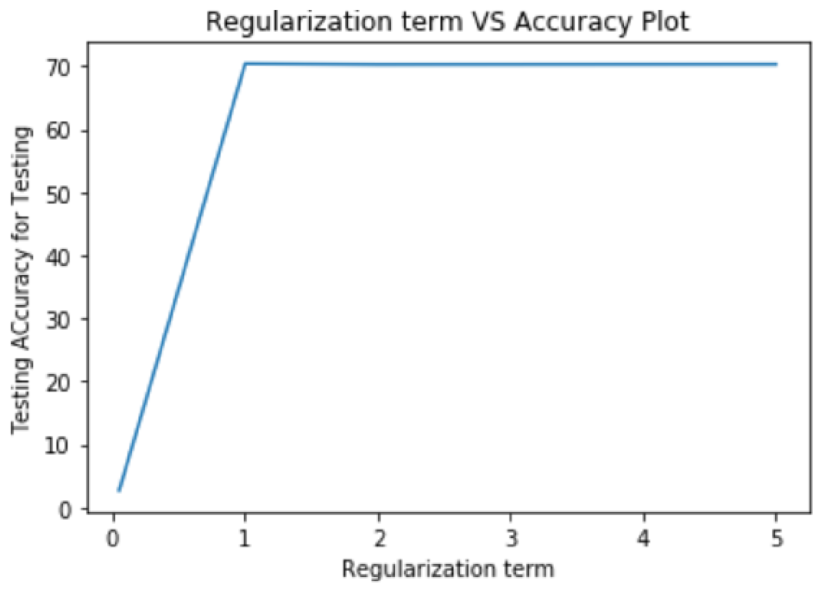

Fig 3.6: Regularization term VS Testing Accuracy

Regularization term	Testing Accuracy
0.05	2.672029880764258
1	70.4065507829335
2	70.27725901450941
5	70.30599051860365

Table 3.7: Tabulation of the Regularization term and Testing Accuracy

The above table depicts the values of the regularization and its corresponding testing accuracy for the model. The graph is plotted for different values of lambda ranging from 0.05 to 5. The graph shows a steep rise in the accuracy after a particular value of lambda, in this case 1. After the sudden spike to a higher accuracy value, it stays constant for all upcoming values of lambda thus avoiding signs of overfitting.

3.4.4 STOCHASTIC GRADIENT DESCENT: Regularization term VS E-RMS

The correlation between the regularization term and the E-RMS of the model predicted by stochastic gradient descent is given below:

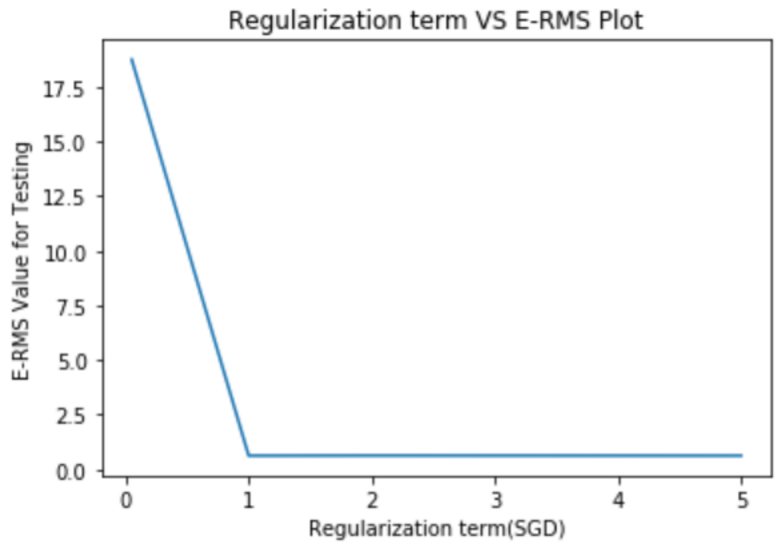

Fig 3.7: Regularization term VS E-RMS

Regularization term	**E-RMS**
0.05	18.761496815300184
1	0.6236074980309467
2	0.6307095247656204
5	0.6242509239418752

Table 3.8: Tabulation of the Regularization term and E-RMS

The above table depicts the values of the regularization and its corresponding E-RMS values. The graph shows a drop in the error rate after a certain value of lambda, in this case 1. After the drop to a low error value, it stays constant for all upcoming values of lambda thus avoiding signs of overfitting.

4.CONCLUSION

This project helps get a clear idea of the working of linear regression. It has also helped understand the functioning of the closed-form solution and the stochastic gradient descent. We calculated the updated weights by computing the error function and minimizing it by each of the approaches. Then finally computed the target values for the testing dataset. We also calculated the root mean square value and the accuracy for the datasets. This project helped get a deeper considering of the impact of the hyper parameters on the performance of the system. Thus, we performed a linear regression to solve the LeToR problem.

REFERENCES

[1] Linear Regression – Introduction to Machine Learning Algorithms: Linear Regression- https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a

[2] Basis Functions – Introduction to Linear Basis Function Models- https://chemicalstatistician.wordpress.com/2014/03/10/machine-learning-lesson-of-the-day-introduction-to-linear-basis-function-models/

[3] K-Means – Introduction to K-Means Clustering - https://www.datascience.com/blog/k-means-clustering

[4] Python functions - https://docs.scipy.org/