CSE574: Introduction to ML(Fall'18)

Project 1.2: Learning to Rank using Linear Regression

Vishal Shivaji Gawade
Department of Computer Science
University at Buffalo
Buffalo, NY,14226
vgawade@buffalo.edu

Abstract

The goal of this project is to use machine learning to solve a problem that arises in Information Retrieval, one known as the Learning to Rank (LeToR) problem.
There are two tasks:
1. Train a linear regression model on LeToR dataset using a closed-form solution.
2. Train a linear regression model on the LeToR dataset using stochastic gradient descent (SGD).

1.Closed form solution for w

We trained our model using closed form solution which is given as,

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

Steps-

1. We used Gaussian radial basis function to solve this linear regression problem. To use Gaussian radial function, we required to find the center of basis function.
2. To find the mean we used k-means clustering. For this basis function, first use k-means clustering to partition the observations into M clusters. Then fit each cluster with a Gaussian radial basis function.
3. After that use these basis function for linear regression. Using k-means clustering we get mean that is Mu. After that using raw data we calculate Big Sigma value, which is our covariance matrix whose all non-diagonal elements ae zero.
4. After that we calculate phi value for individual datasets. We converted out input data into three parts.80% for training,10% for validation and 10% for testing our model.
5. Using training data, value of phi and our regulariser lambda we calculate our weight matrix which is shown in above formula.
6. Finally, we multiply weight matrix and transpose of phi value and we get our output matrix y.
7. After that we calculate root mean square error to evaluate our solution with actual output value.

2. Stochastic Gradient Descent solution for w

We trained our model using solution which is given as,

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)} \quad \text{Where} \quad \Delta\mathbf{w}^{(\tau)} = -\eta^{(\tau)} \nabla E$$

Steps-

1. To find the Stochastic Gradient Descent solution for w, we need to find value of delta E.
2. After we get the value of delta E, we find delta w using which we get w for current data point.
3. We use this current w, called as w now to calculate next weight matrix for next data point as shown in above formula. For first data point we take a random initial value w $^{(0)}$ .In our code we used weight matrix we got from closed form solution as w$^{(0)}$.
4. While calculating each weight matrix, we calculate root mean square value for each data point.
5. Finally, we take minimum root mean square value from all RMS values of data points in our code and say that this is the root mean square error for whole dataset.
6. We are training on 400 data points out of total 69K points for calculating RMS because of computational issue.

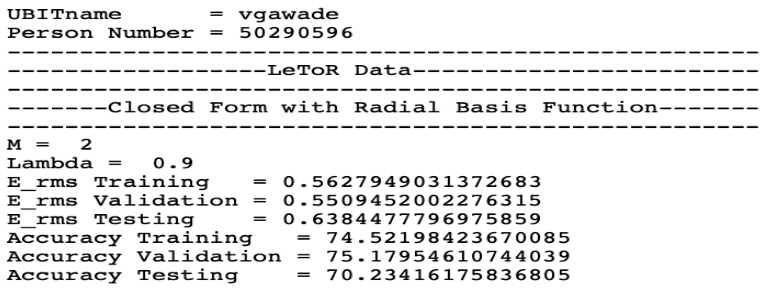
3. Python code

Python libraries used: from sklearn.cluster-KMeans, numpy , csv, math, matplotlib

3.1. Network Setting: The following hyperparameters combination gave me best accuracy and least RMS for closed form solution and SGD.

1.Closed form solution-
Input: M=2; λ=0.9

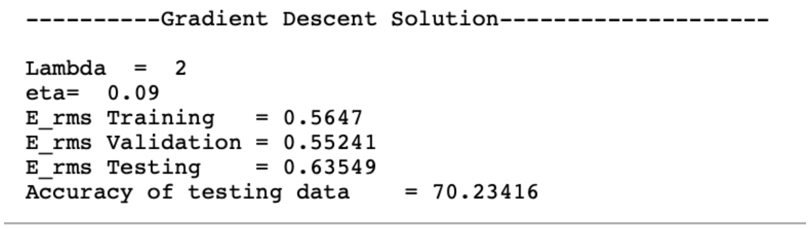
Output:



```
UBITname      = vgawade
Person Number = 50290596
----------------------------------------------------
-----------------LeToR Data-------------------------
----------------------------------------------------
-------Closed Form with Radial Basis Function-------
----------------------------------------------------
M =  2
Lambda =  0.9
E_rms Training   = 0.5627949031372683
E_rms Validation = 0.5509452002276315
E_rms Testing    = 0.6384477796975859
Accuracy Training    = 74.52198423670085
Accuracy Validation  = 75.17954610744039
Accuracy Testing     = 70.23416175836805
```

Figure 1: Output (Closed form solution)

2.Stochastic Gradient Decent solution-
Input: λ=2; η=0.09
Output:



```
----------Gradient Descent Solution--------------------

Lambda  =  2
eta=  0.09
E_rms Training   = 0.5647
E_rms Validation = 0.55241
E_rms Testing    = 0.63549
Accuracy of testing data    = 70.23416
```

Figure 2: Output (SGD)

3.2. Code understanding -

Q. Why don't we use different Mu values for training, validation, testing data?
Ans-We don't change Mu because we are using it for training our model. So, if we change it for validation/testing it will give us wrong results. And we calculate Mu using entire data set. For this reason, we don't use different Mu values.

Q. Why we used 3 and 200 to generate big sigma value?
Ans-We can use any value to multiply big sigma. It just that we have to check how multiplying value affects Erms. We are using 3 and 200 to scale the variables.

Q. In SGD solution, can we initialise weights to zero?
Ans-We can initialise W_ now with any values. We can even pick it up randomly. In our current solution we used Closed form solutions weight matrix as initial W_now for SGD.

Q. Why use K-means clustering?
Ans-We use K-means clustering to generate M basis functions. There are other clustering algorithms but we used K-means because it assumes that clusters are multidimensional spheres and it can return cluster centroids. Other algorithms

just assign a label to every data point is used. Using K-means we can find centroids which we will use to define our regression basis function.

Q. What is big sigma matrix?
Ans-Big sigma is 41*41 matrix called variance matrix over each feature array for training datasets. We use its inverse to calculate Phi value, which is used to calculate basis function.

Q. Can we calculate big sigma for each cluster and use it in our solution?
Ans-Yes, we can find big sigma for each cluster. But if we find the covariance of entire dataset, the covariance will be more effective. Also, the chances of determinant being zero will be less as compare to calculating it for each cluster as we have filtered out those columns of features based on entire dataset. This is why we kept big sigma common for every dataset.

Q. How number of clusters we use affect the basis function?
Ans-If we go on increasing number of clusters, it will reduce the value of variance in clusters. Variance is our big sigma so big sigma will be lower. So our basis function value would be higher.

Q. How can we tune Mu hyperparameter?
Ans- If we increase the number of basis functions, it need more mu by clustering. So that way it gets changed. Another way is by shuffling and splitting the dataset without clustering for a fixed M

4.Hyperparameter Tuning

Let's see how with different network settings, how model behaves and for which setting we get best performance from model.

All below configuration shows the result for Testing data set.

Configuration 1: Changing M basis function value from 1-100 for **Closed form solution**. And checking how it affects root means square value.

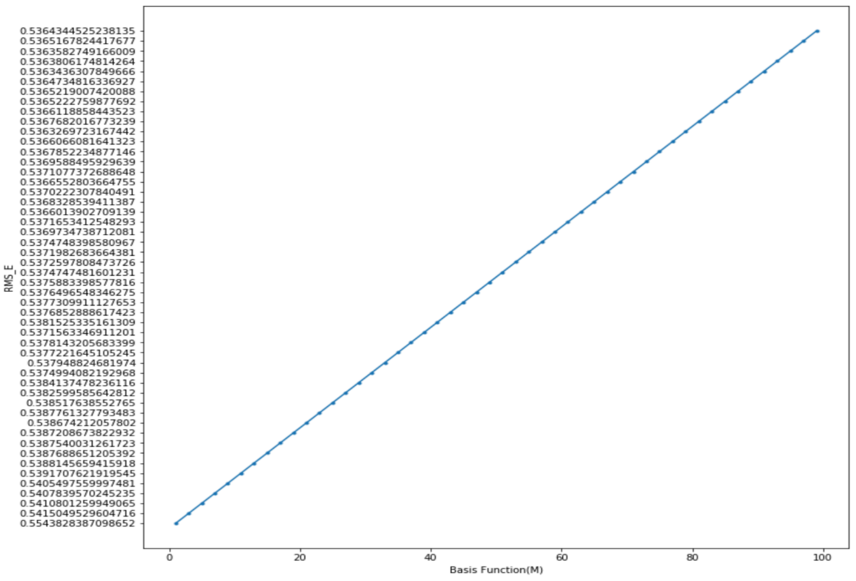


Figure 3: Basis function VS RMS_E graph

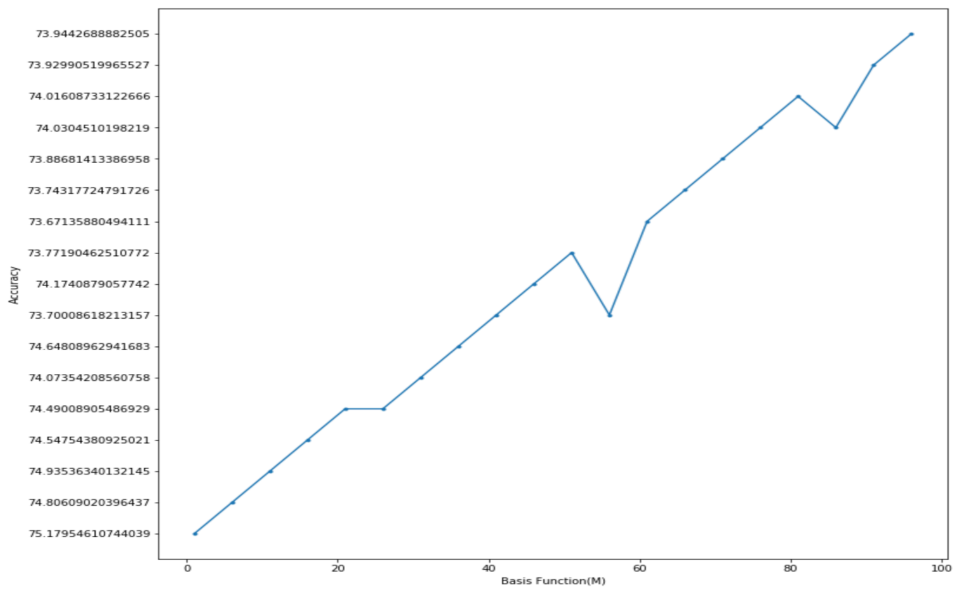


Figure 4: Basis function VS Accuracy (Validation) graph

Conclusion-If we use high value of M that means a greater number of clusters. If we use a greater number of clusters which will lower the variance, and if variance is lower that means Big Sigma value would be lower. All of this leads to increased value of basis function. Because of this root mean square value gets decreased as we increase value of M.

<u>Configuration 2</u>: Changing lambda, regulariser value from 1-10 for **Closed form solution**. And checking how it affects root means square value.

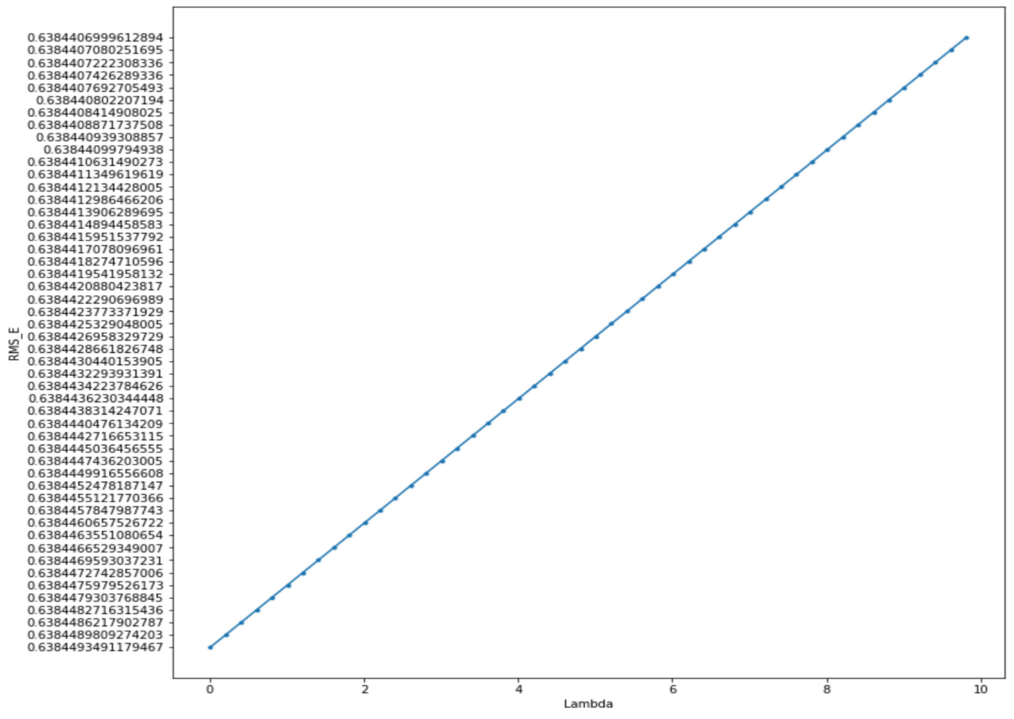


Figure 5: Lambda VS RMS_E graph

Conclusion-If we increment value of regulariser we don't see any noticeable change in value of root mean square error. What regularization does is to bias your parameter estimates to a certain regularization target. As graph shows bias term has very little effect on RMS value. So, we can say that bias term has very little effect on our model's RMS value.

Configuration 3: Changing lambda, regulariser value from 1-10 for **Stochastic gradient decent solution**. And checking how it affects root means square value.

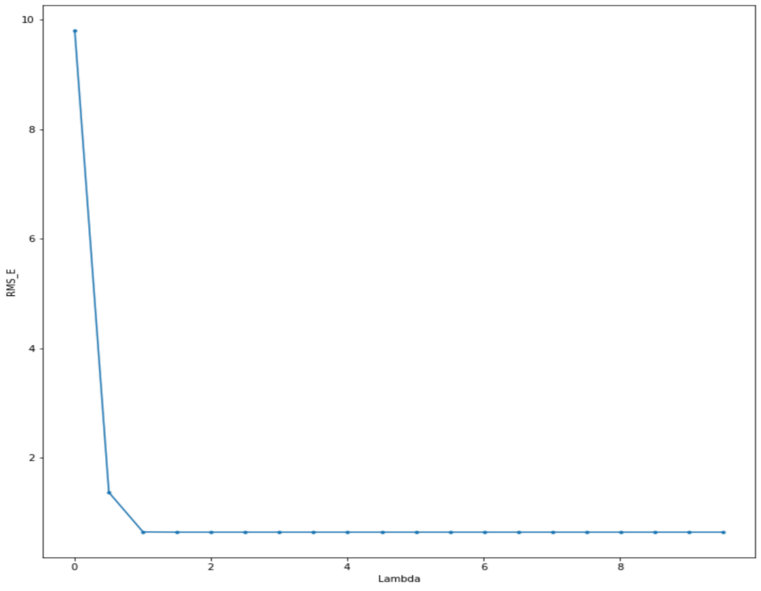


Figure 6: Lambda VS RMS_E graph (SGD)

Conclusion-If we increment value of regulariser we see that for SGD when lambda was 0.01 RMS value was nearly 10. But once we keep on incrementing lambda value it comes below 1. And after that we see very little effect on value of RMS. So we can say that when we keep regulariser value very low we can expect very high RMS error for our problem.

Configuration 4: Changing learning rate eta value from 0.01-1 for **Stochastic gradient decent solution**. And checking how it affects root means square value.

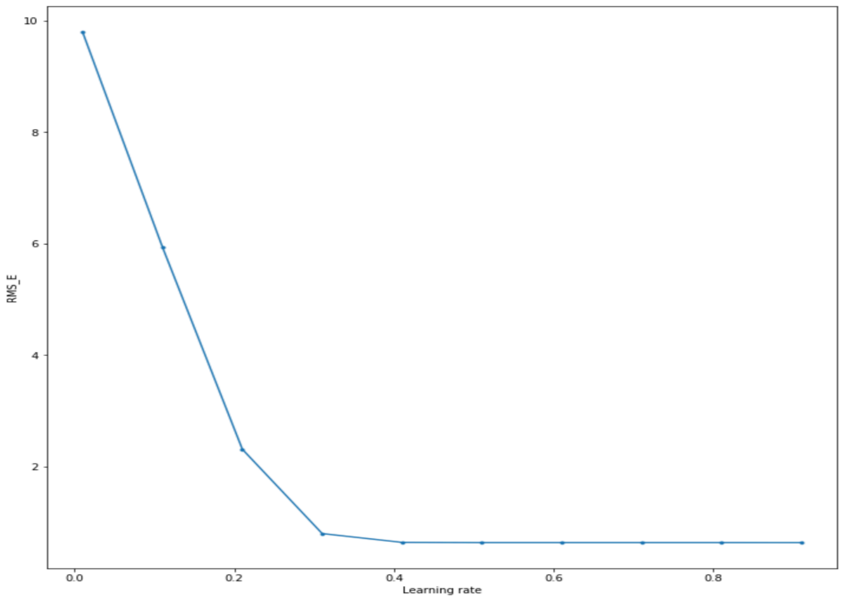


Figure 7: Learning rate VS RMS_E graph (SGD)

Conclusion-If we increment value of learning rate we see that it directly affects the root mean square error. But as we go on incrementing value of learning rate, after eta=0.4 we see steadiness in RMS value. So we should keep our learning rate value between 0.4-0.9 for better performance from our model.

5.Difference between Closed form solution and Stochastic gradient decent solution

Table 1: Difference between Closed form solution vs SGD

Closed Form Solution	**Stochastic gradient decent solution**
Computationally slower compare to SGD	Computationally faster compare to Closed form solution
The closed-form solution preferred for “smaller” datasets.	The SGD solution preferred for “larger” datasets if cost is not issue.
Inverse of X^TX may not exist if the matrix is non-invertible or singular.	We don’t take inverse of X
We get final weight matrix, we don’t update weights incrementally.	Using the SGD optimization algorithm, the weights are updated incrementally after each epoch.
Only issue of larger data set	The larger the training set, the slower our algorithm updates the weights and the longer it may take until it converges to the global cost minimum.

6.Drawbacks of Closed form solution

1. For most nonlinear regression problems there is no closed form solution.
2. Even in linear regression (one of the few cases where a closed form solution is available), it may be impractical to use the formula. The following example shows one way in which this can happen.
   Ex. Inverse of X^TX may not exist if the matrix is non-invertible or singular.
3. It takes lot of time to compute the results.
4. To find the solution we have to do lot of matrix multiplications and inverses because of that it is slower and can’t be used for large data sets.

References

[1] https://stats.stackexchange.com/questions/364554/intuition-behind-xtx-1-in-closed-form-of-w-in-linear-regression

[2] https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html