Learning to Rank using Linear Regression

Anunay Rao
anunayra@buffalo.edu

1 Introduction

The project is to train linear regression model on LeToR dataset by using closed-form solution and then by using stochastic gradient descent approach. For the closed form solution we will be using Moore-Penrose pseudo-inverse method to apply matrix inversion. On the other hand, for stochastic gradient descent we will first randomly initialize the weights and then iteratively update them to get the optimum weights for which our model performs better. Then we will study the effect of various hyperparameters in both the closed form approach and the stochastic gradient descent approach.

2 Preprocessing of the Dataset and Basis Function

The LeToR dataset consists of input which is derived from query-document pair and the target variable which takes the value 0,1 or 2, But here we will obtain real value using linear regression which is more useful for ranking tasks as it avoids collision into only three possible values.

For this linear regression task we need to find the linear function which is a function of some functions called basis functions. For both the closed form solution and stochastic gradient descent solution we are using Gaussian function as the basis function which is given by:

$$e^{-(x-\mu)^T\ \Sigma^{-1}\ (x-\mu)}$$

where x is the input, μ is the mean around which gaussian function is spread i.e it denotes where the function is located and ∑ is the covariance matrix which gives the variance of a particular feature with respect to others. As it requires the inverted covariance matrix and it is known that if a matrix has any column with zero variance it is non-invertible. In the LeToR dataset we have column number 6,7,8,9 and 10 denoting IDF of body, IDF of anchor, IDF of title, IDF of URL and IDF of whole document respectively with zero variance, infact these features have zero value for all the samples therefore we need to filter out these columns in order to proceed with the linear regression task. Here, we are taking 10 Gaussian basis functions which will require 10 means so we form 10 clusters of the dataset and for each cluster we will have μ vector of dimension 1x41. Gaussian basis function is used because its value depends on the origin or certain point. It is symmetrical about central maximum.

3 Performance Metric

We will evaluate the solution obtained by using Root Mean Square (RMS) error defined as $E_{RMS} = \sqrt{2E(w^*)/N_V}$

where w^* is the solution and N_V is the size of dataset. Accuracy is not a good performance metric for this linear regression tasks and hence we will study the effect of various hyper-parameters on E_{RMS} and not accuracy.

4 Effect of various Hyper-parameters

This section presents the experiments conducted by varying the values of Hyper-parameters and how they affect the performance of the model.

4.1 CLOSED FORM SOLUTION

Here, we are using Moore-Penrose pseudo-inverse method to find the inverse of a matrix because we want the inverse of ϕ which is not a square matrix. We will be using regularization term λ to take care of overfitting.

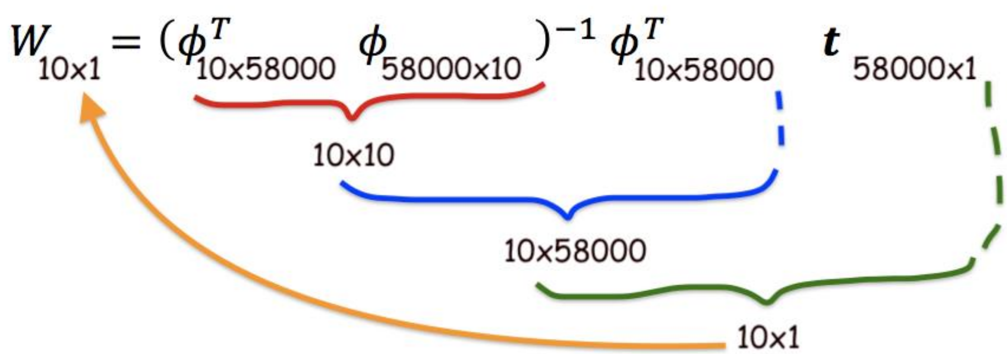


Figure 1: Dimensionality for Training

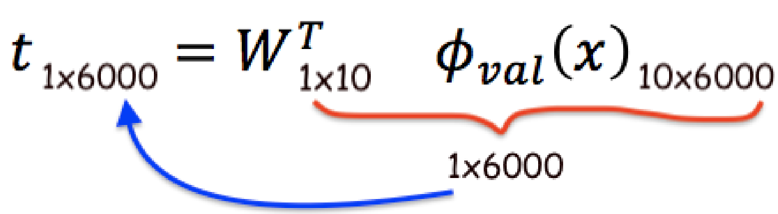


Figure 2: Getting target value from Validation Design Matrix

Default configuration:

- Regularization term, λ = 0.03
- Training Percent = 80
- Validation Percent = 10
- Testing Percent = 10
- Number of Basis Function, M = 10

4.1.1 Effect of Number of Basis Function (M)

	10	20	30	40	50	60	70	80	90
Erms_Training	0.549469	0.546123	0.543265	0.541604	0.540736	0.539609	0.539366	0.539054	0.538997
Erms_Validation	0.538428	0.53836	0.53718	0.536806	0.536946	0.536299	0.535902	0.53599	0.536013
Erms_Testing	0.627979	0.62619	0.622963	0.620712	0.619718	0.619219	0.618967	0.618747	0.619126

Table 1: E_{RMS} values for different values of M

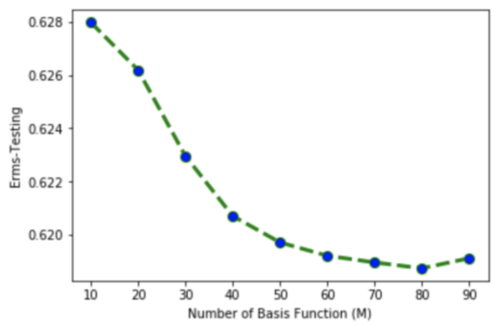


Figure 3: Variation of E_{RMS}-Testing with M

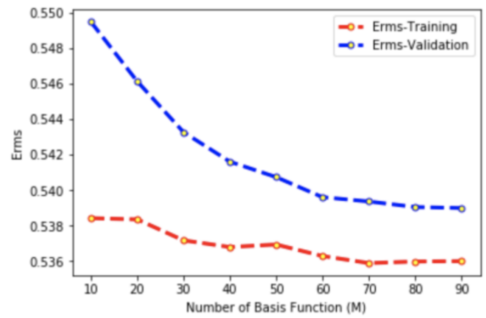


Figure 4: Variation of E_{RMS}-Training and Validation with M

Conclusion

As the number of basis function increases E_{RMS} decreases but if we will use too many basis functions then our model will overfit.

4.1.2 Effect of Regularization term (λ)

	0.03	0.3	3	30	300
Erms_Training	0.549469	0.549618	0.550344	0.551271	0.554998
Erms_Validation	0.538428	0.538764	0.539797	0.540525	0.543709
Erms_Testing	0.627979	0.628213	0.628197	0.62847	0.631591

Table 2: Respective E_{RMS} values for different λ

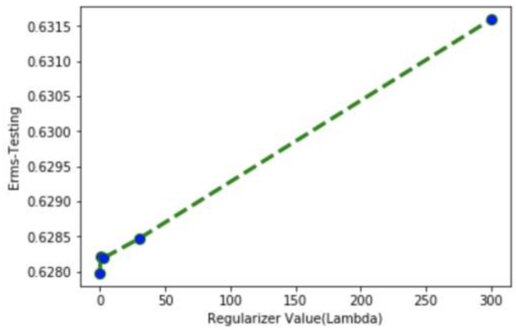


Figure 5: Variation of Erms-testing with regularizer

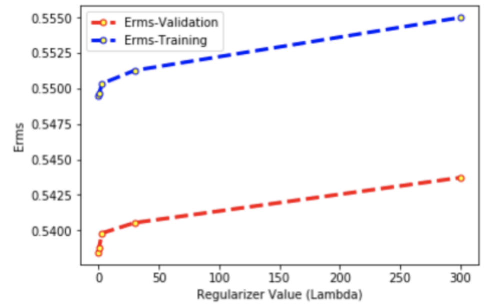


Figure 6: Variation of Erms Training and Validation with regularizer

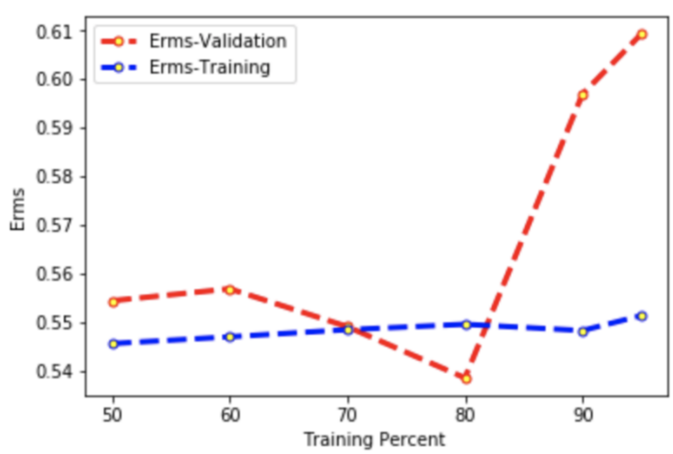
Conclusion

As the the value of λ increases the Erms also increases although for smaller values of λ there is not any drastic change in Erms. It is used to avoid overfitting so that our model does not strictly fit to the training data points.

4.1.3 Effect of Data Split (Training: Validation: Testing)

	50:25:25	60:20:20	70:15:15	80:10:10	90:5:5	95:2.5:2.5
Erms_Training	0.545498	0.546892	0.548359	0.549469	0.548179	0.551264
Erms_Validation	0.55432	0.55678	0.548985	0.538428	0.596895	0.609278
Erms_Testing	0.580895	0.584775	0.604691	0.627979	0.657624	0.704038

Table 3: Erms values for different Data Split



Conclusion

As the training data set increases the model is validated and tested on less number of data sets. As seen from the data collected by varying the data split we have that for 70:15:15 the Erms training and validation are almost same but on further increasing the training percent thereby reducing the validation and testing data set the Erms of Validation and Testing increased by big margin thereby indicating the overfitting of the model. The weights we have obtained are not optimum weights.

4.2 STOCHASTIC GRADIENT DESCENT

The gradient descent approach can be break drown in three steps:

- Randomly initialize the weights.
- Update the weights iteratively.

$$w_{new} = w_{old} + \Delta w$$

$$\Delta w_{old} = -\eta_{old} \nabla E$$

$$\nabla E = \nabla E_D + \lambda \nabla E_W$$

$$\nabla E_D = -(t_n - w_{old}^T \phi(x_n))\phi(x_n)$$

$$\nabla E_W = w_{old}$$

- Calculate the E_{RMS} value.

Here η is a learning rate which determines the step size in which weight gets updated. The lower the value of learning rate the model will converge slowly as it will take small step towards the minima whereas if the value of learning rate is high it will take bigger steps but then it is not guaranteed that it will converge to local minima. Therefore we need an optimum value of learning rate.

Learning Process and why training over only 400 sample:

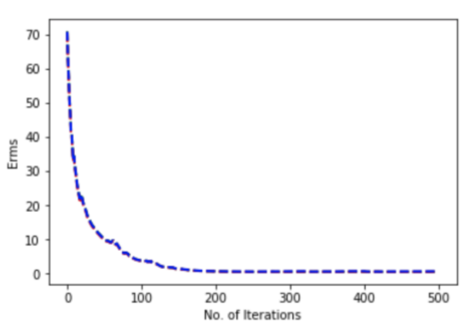


Figure 7:Variation of Erms with No. of Iterations

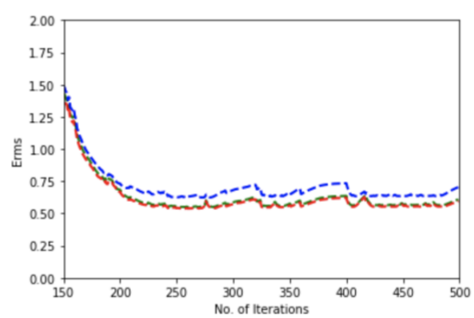


Figure 8: Magnified view of Figure 7

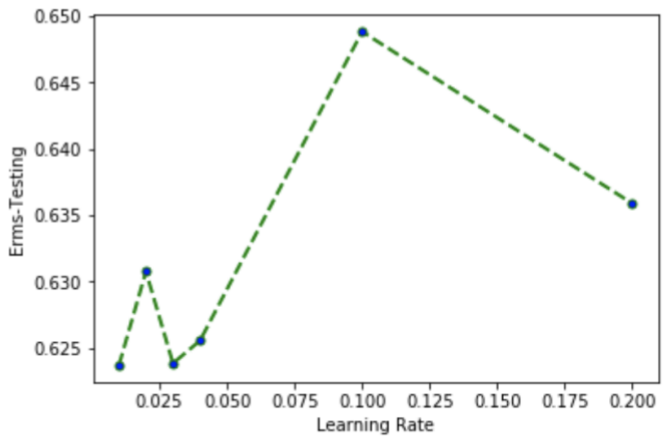
Conclusion

As it is quite evident from the above two graphs that after 400 iterations(samples) the Erms values becomes quite stable and so there is not any significant change in weights and so we stop at 400.

4.2.1 Effect of learning rate

	0.0001	0.001	0.01	0.02	0.03	0.04	0.1	0.2
Erms_Training	53.6082	13.1223	0.54964	0.55869	0.54955	0.55315	0.58417	0.56518
Erms_Validation	53.0331	12.7047	0.53846	0.54741	0.53842	0.5416	0.57105	0.55413
Erms_Testing	53.476	13.0067	0.62372	0.63076	0.62386	0.62556	0.64879	0.63588

Table 4: Erms values for different values of learning rate

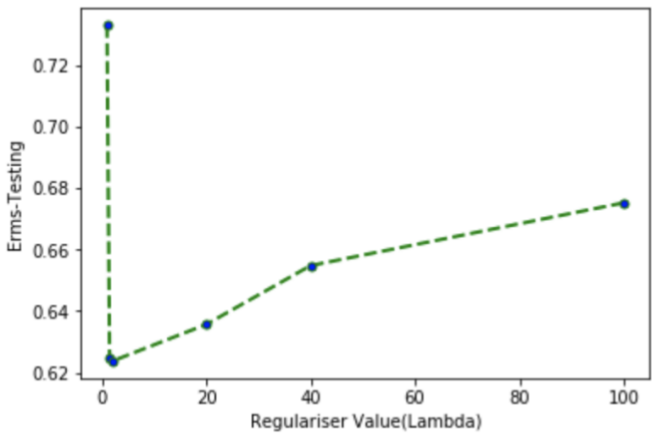


Conclusion

As we have considered only 400 samples(iterations) then it is quite evident that for η = 0.0001 we have higher Erms because it will converge slower as it is taking smaller steps towards the minima. The plot is not shown for η = 0.0001 and 0.001 to get the better understanding from the plot. As the learning rate increases it will take bigger steps towards minima but it may or may not converge. From the plot we have minimum Erms for η = 0.01. Therefore good learning rate is not a smaller value nor a higher value but it is a optimum value.

4.2.1 Effect of Regularization term (λ)

	0.2	1	1.5	2	20	40	100
Erms_Testing	12.9095	0.73303	0.62483	0.62372	0.63566	0.65481	0.67524



Conclusion

As the value of λ increases Erms also increases. There is an optimum value of λ for which the Erms is minimum. For a very low value of λ we have higher Erms which means the weight we have are not the optimum weights.

5 Other Variations

5.1 Deleting Column with no information:

In the LeToR dataset the column number 46 namely, Number of child page has the value 0 for all the samples except one. Therefore deleting this column and checking out its effect:

Closed Form:	
Before deleting:	After Deleting:
E_rms Training = 0.5488846063481925 E_rms Validation = 0.5373461870705957 E_rms Testing = 0.6273650154146978	E_rms Training = 0.5494694067137487 E_rms Validation = 0.5384281741389716 E_rms Testing = 0.6279788453854206
Gradient Descent:	
Before deleting	After deleting:
E_rms Training = 0.54986 E_rms Validation = 0.53777 E_rms Testing = 0.6247	E_rms Training = 0.55044 E_rms Validation = 0.53899 E_rms Testing = 0.62518

Conclusion

Negligible change in Erms and thus column 46 (Number of child page) was useless.

5.2 Hierarchical Clustering

An attempt to find the optimum number of clusters but with limited computational power and time I could only get the dendrogram for 30% of the training dataset.

Therefore, determining clusters for it so the longest vertical distance without any horizontal line passing through it is selected and a horizontal line is drawn through it. The number of vertical lines this newly created horizontal line passes is equal to number of clusters

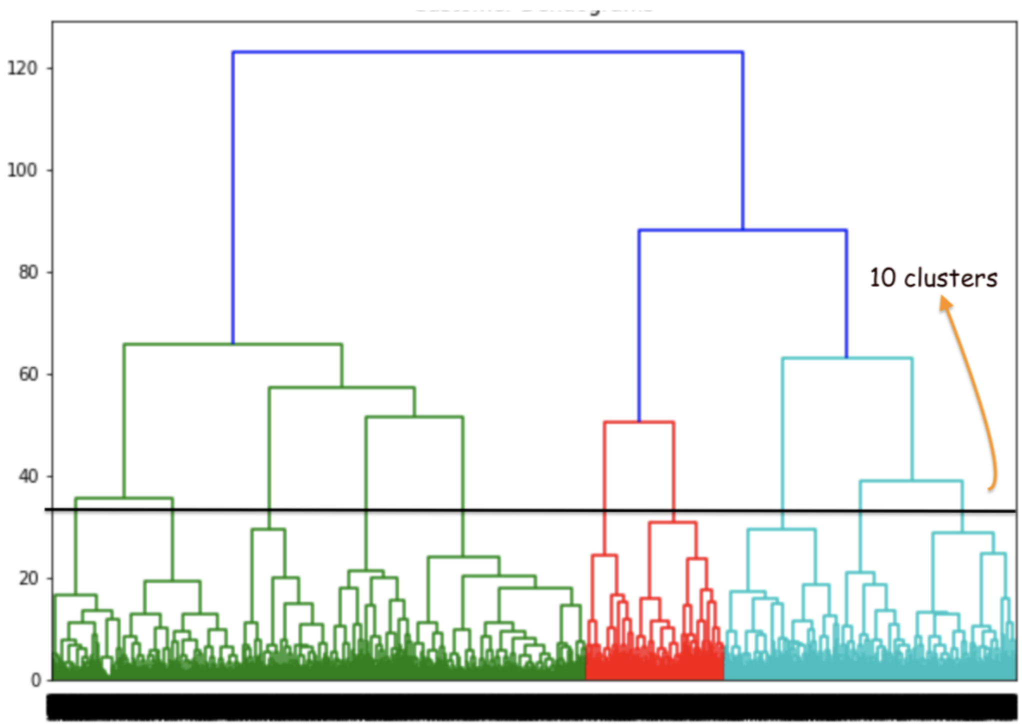


Figure 9: Dendrogram for 30% Validation Data

Conclusion

So after cutting the dendrogram we have 10 clusters.

References

[1] Stack Abuse. (2018). *Hierarchical Clustering with Python and Scikit-Learn*. [online] Available at: https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/ [Accessed 5 Oct. 2018].

[2] Towards Data Science. (2018). Machine Learning – Towards Data Science. [online] Available at: https://towardsdatascience.com/machine-learning/home [Accessed 10 Sep. 2018].

[3] Brownlee, J. (2018). Machine Learning Mastery. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/