A Report on Learning to Rank Using Linear Regression

Miki Padhiary
UBID: mikipadh Person Number: 50286289

October 11, 2018

Abstract

In this experiment, We studied how to solve Learning to Rank (LeToR) problem using machine learning. LeTor is a problem in Information Retrieval. We devised this as a problem of linear regression where we map an input vector x to a real-valued scalar target $y(x, w)$. Experiments on various hyper-parameters such as learning rate, number of basis function, regularization term were performed for their effect on the accuracy of the program. The results of hyper-parameters tuning is plotted in section 5.

1 Introduction

Following two approaches were taken:

1. Trained a linear regression model on LeToR dataset using a closed-form solution.
2. Trained a linear regression model on the LeToR dataset using stochastic gradient descent (SGD).

For the experiment, we had used Microsoft LETOR 4.0 Dataset. The entire dataset consists of 69623 query-document pairs(rows), each having 46 features. These 46-dimensional features are the input vector x for our linear regression model. The first column is the relevance label of the row. The larger the relevance label, the better is the match between query and document. Data were preprocessed and splitted into training(80%), validation(10%) and testing(10%) purposes. Training data is the data which is used to learn a model. Validation data is used while testing to check if the training is correctly performed or not. Testing data is the data on which the testing is done.

2 Linear Regression Model

Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). For our case, the linear regression function is defined as:

$$y(x, w) = w^T \phi(x) \tag{1}$$

where $\mathbf{w} = (\mathbf{w}_0,\mathbf{w}_1,...,\mathbf{w}_{m-1})$ is a weight vector to be learned from training samples and $\phi = (\phi_0,\phi_1,...,\phi_{m-1})^\top$ is a vector of $\mathbf{M}$ basis functions.

Gaussian radial basis function is defined as:

$$\phi_j(x) = exp(-\frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j) \tag{2}$$

where μ_j is the center of the basis function and $\mathbf{\Sigma}_j$ decides how broadly the basis function spreads.

Our objective is to minimize the sum-of-squares error

$$E_D = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2 \tag{3}$$

where ϕ is the design matrix

2.1 Closed Form Solution

Closed-form solution with least-squared regularization is defined by

$$\boldsymbol{W}* = (\lambda\boldsymbol{I} + \boldsymbol{\phi}^T\boldsymbol{\phi})^{-1}\boldsymbol{\phi}^T\boldsymbol{t} \tag{4}$$

The RMS Error is calculated using the following formulae:

$$\boldsymbol{E}_{RMS} = \sqrt{\frac{\boldsymbol{E}(w*)}{N_v}} \tag{5}$$

2.2 Stochastic Gradient Descent

Gradient descent is an algorithm that minimizes functions. Given a function defined by a set of parameters, gradient descent starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimize the function.

Weight is calculated using the following:

$$\boldsymbol{W}^{(\tau+1)} = \boldsymbol{W}^{(\tau)} + \Delta\boldsymbol{W}^{(\tau)} \tag{6}$$

where $\boldsymbol{\Delta W}^{(\tau)} = -\eta^\tau\nabla E$. Here, η^τ is the learning rate.

2.3 Steps Performed

1. Data Preprocessing
    (a) Extract features and target from the dataset.
    (b) Delete the features which have 0 variances.
    (c) Split the data into training, validation and testing data-set. The training set takes around 80% of the total. The validation set takes about 10% . The testing set takes the rest. The three sets should NOT overlap.

2. Steps for Closed Form Solution
    (a) Create a design matrix ϕ
    (b) Calculate weights.
    (c) Find RMS for validation dataset.
    (d) Find RMS for testing dataset.
    (e) Find RMS for training dataset.
3. Steps for Gradient Descent Solution
    (a) Initialize weight matrix.
    (b) Iteratively update weights for each datapoints.
    (c) Calculate $\mathbf{E}_{TrainingRMS}$ after every iteration.
    (d) Calculate $\mathbf{E}_{ValidationRMS}$ after every iteration.
    (e) Calculate $\mathbf{E}_{TestingRMS}$ after every iteration.

3 Experiments

Various experiments were performed on the following hyper-parameters and their accuracy and sum of squared errors is observed:

1. Regularization Term λ
2. Choosing Learning Rate η
3. Number of Basis Functions M
    (a) **Centers for Gaussian radial basis functions** μ_j
    (b) **Spread for Gaussian radial basis functions** σ_j
    (c) **k-means clustering:** K-means clustering is a type of unsupervised learning, which is used when we have unlabeled data (i.e., data without defined categories or groups). This algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. The centroids of the K clusters, is used to label new data.

4 Hyper-parameters at a glance

A hyper-parameter is a parameter whose value is set before the learning process begins. Different model training algorithms require different hyper-parameters. The training algorithm learns the parameters from the data. For our situation, the following hyper-parameters were modified:

Hyper-parameters
Regularization Term λ
Number of Basis Functions M
Learning Rate η

Regularization Term λ: Regularization is a technique used to solve over-fitting problems. It is penalizing the loss function by adding a multiple of L norms of the weight

vector v. Regularization tuning can be performed by **cross-validation**. Cross Validation is dividing the training data, train the model for a fixed value of λ and test it on remaining subsets. Repeat the process while varying λ. Select the best λ that minimizes the error function.

Number of Basis Functions M: The space T is divided into **K** clusters $\{C_1, C_2, ...C_k\}$ that corresponds to the number of basis functions.The centroid of these clusters form the μ_j vector. As data-set is not linear we use basis functions, which introduces non linearity.

Learning Rate η: Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect to the loss gradient. The lower the value, the slower we travel along the downward slope.

5 Investigations

The best accuracy and error retrieved after changing various parameters are described in the form of graphs below. This experiment is carried out in two parts.

```
-------Closed Form with Radial Basis Function-------
-----------------------------------------------------
M = 10
Lambda = 0.9
E_rms Training   = 0.5428452341882348
E_rms Validation = 0.536676106164761
E_rms Testing    = 0.6223687219124756
Training Dataset Accuracy   = 73.7230470924074
Validation Dataset Accuracy = 74.07354208560758
Testing Dataset Accuracy = 69.35785088349375
```

```
----------Gradient Descent Solution--------------------
M = 15
Lambda  = 0.0001
eta=0.01
E_rms Training   = 0.54444
E_rms Validation = 0.54152
E_rms Testing    = 0.62281
Training Dataset Accuracy  = 74.52198423670085
Validation Dataset Accuracy = 75.17954610744039
Testing Dataset Accuracy    = 70.23416175836805
```

Figure 1: Closed Form Vs. Gradient Descent Solution Results

5.1 Closed Form Solution

5.1.1 Regularization Term λ:

The program is executed for λ values ranging from 0.0002 to 100 and the graphs obtained is plotted below.[Refer Figure 2]

5.1.2 Number of Basis Functions M:

The program is executed for M values ranging from 1 to 200 and the graphs obtained is plotted below.[Refer Figure 3]

5.2 Stochastic Gradient Descent

5.2.1 Regularization Term λ:

The program is executed for λ values ranging from 0.001 to 5 and the graphs obtained is plotted below.[Refer figure 4]

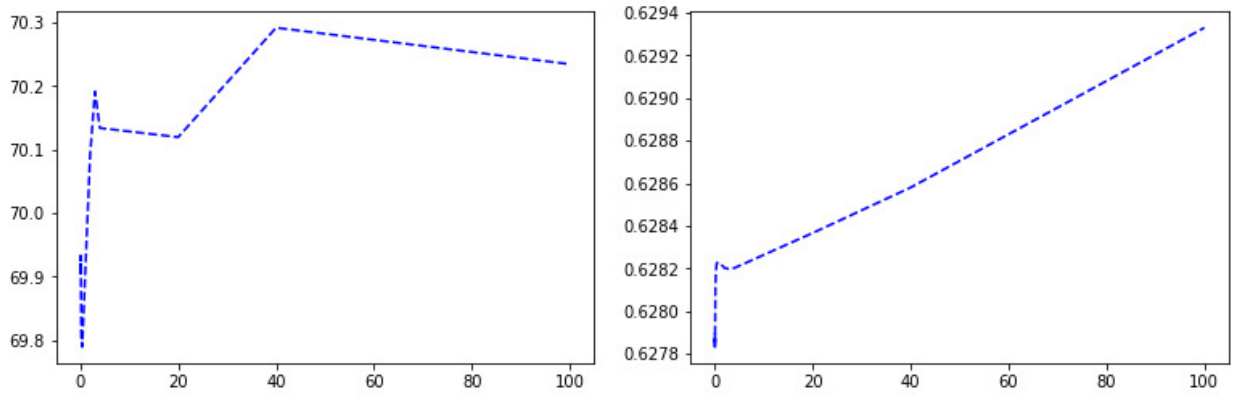

Figure 2: Accuracy Vs Error for Regularization

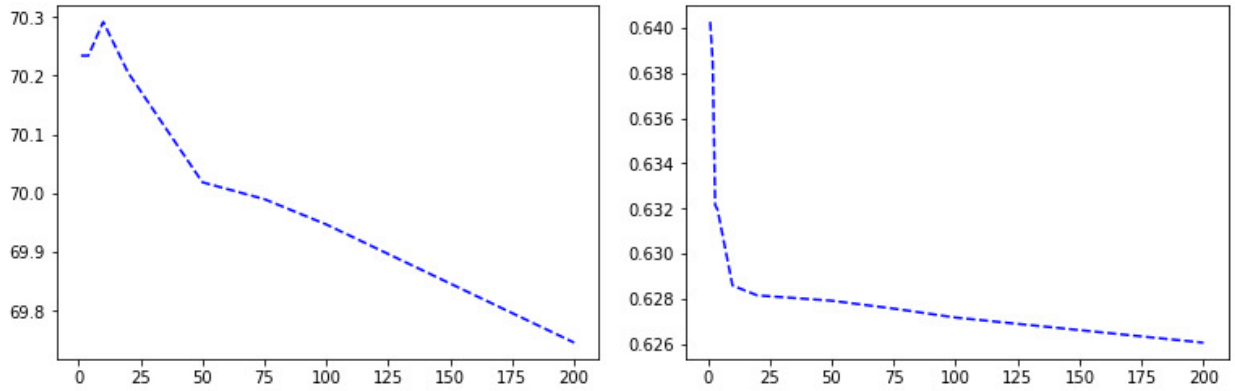

Figure 3: Accuracy Vs Error for Basis Function

5.2.2 Number of Basis Functions M:

The program is executed for M values ranging from 1 to 200 and the graphs obtained is plotted above.[Refer figure 3]

5.2.3 Learning Rate η

This is only applicable to Gradient Descent Solution. The program is executed for η values ranging from 0.00001 to 0.4 and the graphs obtained is plotted below.[Refer figure 5]

6 Results

6.1 Comparison between Closed Form and Stochastic Gradient Descent

This section mentions the comparison between the different hyper-parameters in both closed form and Stochastic Gradient Descent.

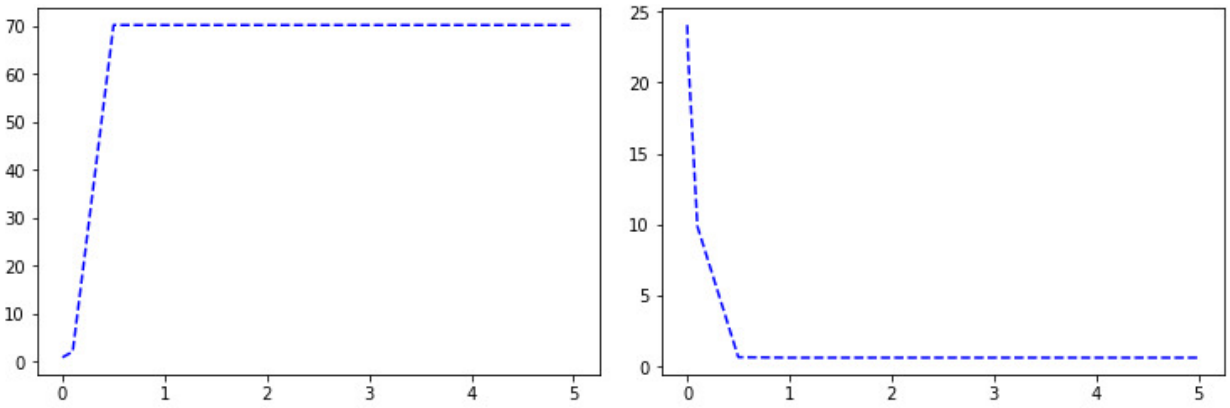

Figure 4: Accuracy Vs Error for Regularization Function

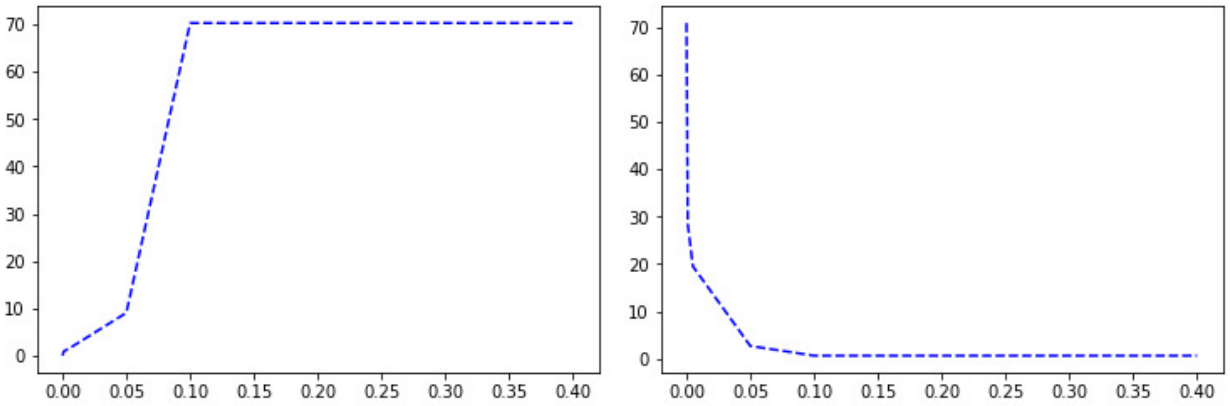

Figure 5: Accuracy Vs Error for Learning Rate

6.1.1 Regularization Term λ:

It is seen that for lower values of λ, the accuracy is low in the case of Closed Form. However as the value is increased, the accuracy increases but after a certain point it does not increase more for both Closed Form and Stochastic Gradient Descent solutions.

Reason When λ is low, the over-fitting is high and as a result error is high and accuracy goes for a toll. As we increase λ, the accuracy increases and λ is stabilized.

6.1.2 Number of Basis Functions M:

It has been observed that decreasing M to 1 does not have any impact on accuracy nor on the error. However, increasing it to 200, decreases both the accuracy and error values. **Reason** This could be attributed to the fact that an increase in number of clusters reduces the variance, Big Sigma(Σ) for both Closed Form and Stochastic Gradient Descent solutions.

6.1.3 Learning Rate η:

This is only applicable to Gradient Descent Solution. In the perfect case, if learning rate is increased then the error should decrease and the accuracy should increase. In our current scenario, as we go on increasing learning rate, the accuracy increases for a while and then it becomes constant. The error values goes on decreasing with increase in learning rate.

7 Conclusions

In this experiment, two machine learning algorithms namely Closed Form Solution and Stochastic Gradient Solution were studied to solve LeToR problem. Various changes to hyper-parameters were done to increase the accuracy and simultaneously minimize the sum-of-squares error. The best accuracy and error achieved was 70.23% and 62.28% in both the cases.

References

[1] Linear Regression
https://en.wikipedia.org/wiki/Linear_regression

[2] Regularization
https://codeburst.io/what-is-regularization-in-machine-learning-aed5a1c36590

[3] Smoothing Techniques
https://datascienceplus.com/smoothing-techniques-using-\
basis-functions-gaussian-basis/

[4] Project Description
https://ublearns.buffalo.edu/bbcswebdav/pid-4714290-dt-content-rid-19922272_1/courses/2189_24904_COMB/project1.2.pdf

[5] Gradient Descent
https://en.wikipedia.org/wiki/Gradient_descent

[6] Hyper-parameters
https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)

[7] Learning Rate and its impact
https://towardsdatascience.com/understanding-learning-rates-and\
-how-it-improves-performance-in-deep-learning-d0d4059c1c10