

A Report on Learning to Rank using Linear Regression

Ekta Katiyar

Person Number: 50291702

October 10, 2018

Abstract

In this experiment we have studied how to resolve the problem of Learning to Rank by minimizing the root mean square error for the dataset on which we are training our model. This experiment also helped in understanding various aspects of hyperparameters. These are the parameters whose value is set before the learning process begins and tuning hyperparameters impacts the Learning process of the model. If hyperparameters are poorly chosen, then the network will learn slowly or might not learn at all.

Introduction

Our problem statement was to resolve the problem of Learning to Rank using Linear Regression model of machine learning.

Problem Definitions:

- First, we have to train our linear regression model with Learning to Rank dataset using Closed form solution in such a way that we obtain minimum root mean square error.
- Second, we have to train our linear regression model with Learning to Rank dataset using Stochastic Gradient descent in such a way that we obtain minimum root mean square error.

Experiments

Below is the list of various network tunings done in the hyperparameters while implementing closed form solution:

1. Change in the number of Basis functions (M).
2. Change in regularizer (C_Lambda).

Below is the list of network tuning done in the hyperparameters while training our model using Stochastic Gradient Descent(SGD):

1. Change in the regularizer (La).
2. Change in Learning rate (learningRate).
3. Change in weight during initialization(220).
4. Change in the number of epochs(400).

Explanations

- **LeToR Dataset:** In order to implement linear regression on a learning to rank (LeToR) dataset we will be using Microsoft LETOR 4.0 Dataset. We need to process Querylevelnorm.txt to extract the Target and Raw Data set. After processing it, we have obtained the csv files of Target and Raw Dataset as Querylevelnorm_t.csv and Querylevelnorm_X.csv respectively.

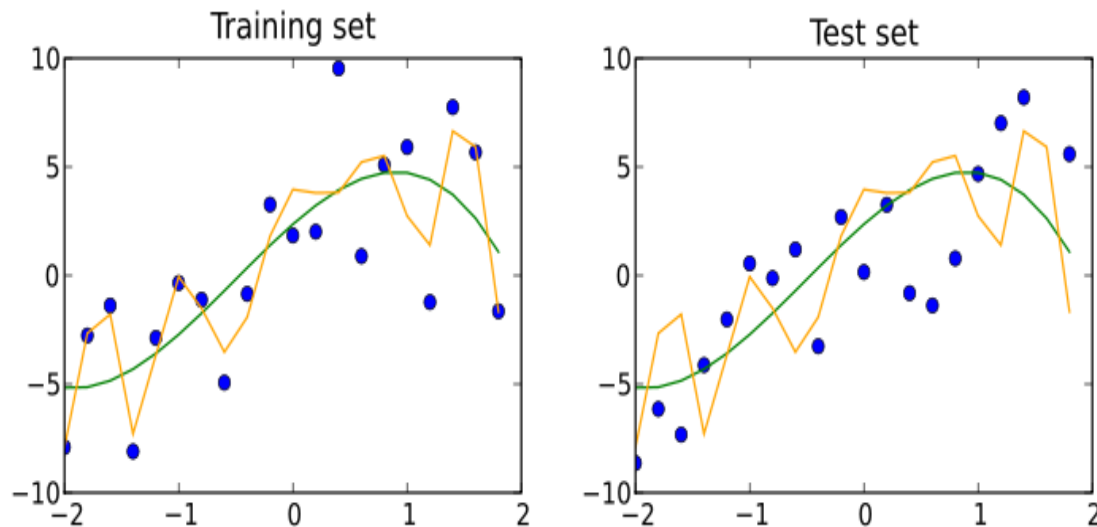
After obtaining the dataset we will convert the target dataset into vector and will generate the data matrix for data in Querylevelnorm_X.csv.

Inside dataMatrix[] all the rows represent the number of queries and the columns represent the number of features. While calculating the variance of dataMatrix we will be deleting the features for which the variance is very less. We will be using axis=1 or axis=0 to represent if we want to perform operation on the rows or the columns.

- **IsSynthetic:** This variable is used to check if the data is synthetically manufactured as synthetic data contains inconsistencies when trying to replicate the complexity found within the original dataset.
- **Training data:** The data we used to feed to our model for it to learn the flow of information is known as Training Data. We will be $\mu\mu$ model.
- **Validation data:** A validation dataset is a sample of data held back from training your model that is used to give an estimate of how skilled our model is after tuning the model hyperparameters.

We will be considering the 10% of our raw dataset as Validation Data for our model.

- **Test Data:** It gives an unbiased estimate of how well is our model is tuned when comparing or selecting between the final models. It follows the same probability distribution as the training dataset.



We will be considering the 10% of the whole dataset as the testing data.

- **Supervised Learning:** It refers to a learning methodology where we supervise a machine learning model. We teach the model with labelled dataset and it is trained on the attributes, features, observations, numerical data and categorical data of the training data set. After training, accuracy of the model will be determined using test dataset.
- **Unsupervised Learning:** In this approach, we let our model to work on its own which may not be visible to us. It uses machine learning algorithms that used unlabeled dataset.
- **Regularizer:** The model gets overfitted that will led it to the lower accuracy measure. Therefore, we require our model to unlearn i.e to avoid overfitting of the curve and noisy data points; by noisy we mean the data points that do not really represent the true properties of the data. Regularizer can be represented by λ
- **Radial Basis Function:** It is a real valued function ϕ whose value depends upon the distance from origin in such a way that

$$\phi(x) = \phi(\|x\|)$$

or alternatively on the distance from a point p , such that

$$\phi(x, p) = \phi(\|x - p\|)$$

Radial Basis function can be represented in the below form

$$y(x) = \sum_{i=1}^N w_i \phi(\|x - x_i\|)$$

where the approximation function $y(x)$ is represented as the sum of N radial basis functions

x_i is the center for each radial basis function

w_i is the coefficient of weights associated.

- **K-means clustering:** It is an unsupervised learning methodology that works on unlabeled data to find the groups based on the given features. K-means clustering algorithm returns us following

- The centroids of the K-clusters that can be represented by μ .
- Labels of the training data i.e each point is assigned to the single cluster.
- **ERMS:** It refers to the root mean square error which is a measure to represent the difference between the predicted value from the model and the value estimated by the model.

$$\text{ERMS} = \frac{1}{2}(t_n - \hat{t}_n)^2$$

Where t_n is the expected target value

And \hat{t}_n is the observed target value from the model

CLOSED FORM SOLUTION TO OBTAIN WEIGHTS

Closed form solution can be preferred for small datasets where calculation of heavy inverse matrix will not be a problem. The linear regression function in the closed form solution is defined as:

$$y = w_0x_0 + w_1x_1 + \dots + w_mx_m = \sum_{j=0}^m = \mathbf{w}^T \mathbf{x}$$

where y is the response variable, \mathbf{x} is an m -dimensional sample vector, and \mathbf{w} is the weight vector (vector of coefficients).

Step1: First step to proceed with the closed form solution is to obtain the training data set which is 80% of the data provided to us.

Step2: Second step is to determine the weights for all the data points using equation can be derived from

$$t = \mathbf{w}^T \phi(f)$$

where t is the target vector output for a data point

Therefore,

$$\mathbf{w} = \phi(f)^{-1} * t$$

Where ϕ refers to the design matrix

$$\phi = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_M(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_M(x_2) \\ \phi_1(x_n) & \phi_2(x_n) & \dots & \phi_M(x_n) \end{bmatrix}$$

where x_1 refers to the 1st data row.

and M refers to the number of basis functions.

Since it may not be a square matrix so in order to find the inverse of it, we will use

$$\mathbf{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

In order to calculate the value of $\phi(x)$, we need to determine μ using K- means clustering and then Spread for Gaussian radial basis functions such that

$$\phi(x) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Using the above equation, we will be obtaining the Big Sigma matrix

$$\Sigma = \begin{bmatrix} \sigma_{11}^2 & 0 & 0 \\ 0 & \sigma_{22}^2 & 0 \\ 0 & 0 & \sigma_{33}^2 \end{bmatrix} \text{ ..and so on}$$

Now after computing the dimensionality of Big Sigma and ϕ and regularizer(λ), we can get the dimensionality of weight for each data row in the model.

Step 3: Third step after obtaining the weight we will need to calculate the ERMS for the training data using

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

In order to avoid the noisy data point that causes overfitting of the curve and to achieve the better generalization, we will be using regularizer

$$E(w) = E_D(w) + \lambda E_W(w)$$

Step 4: Similarly, we can calculate the ERMS for validation and testing data set.

STOCHASTIC GRADIENT DESCENT SOLUTION TO OBTAIN WEIGHTS

Step 1: First step is the initialization of the random weights for the nodes. In SGD, we update the weights after each training sample.

Step 2: We will then update the weights iteratively using below equation

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

$$\text{where } \nabla E_D = -\{t_n - w^T \phi(x_n)\} \phi(x_n)$$

$$\text{and } \nabla E_W = w$$

$$\text{therefore, } \lambda \nabla E_W = \lambda w$$

Step 3: Here we will determine the target value \mathbf{t} using $\mathbf{t} = \mathbf{w}^T \phi(\mathbf{x})$, where $\phi(\mathbf{x})$ will vary for training, validation and testing dataset.

Step 4: We need to calculate the ERMS now by using

$$E_{RMS} = \sqrt{2E(w^*)/N_V}$$

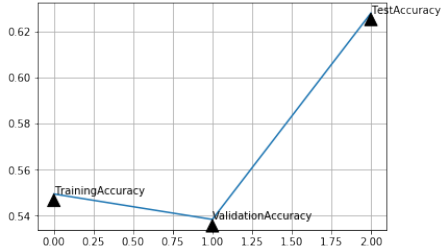
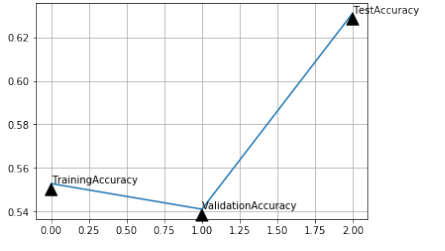
where w^* is the solution and N_V is the size of the test dataset.

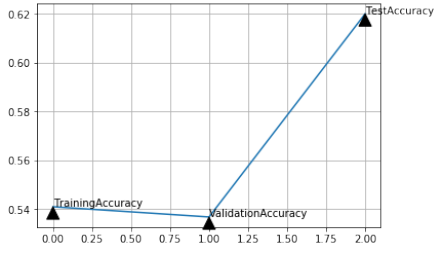
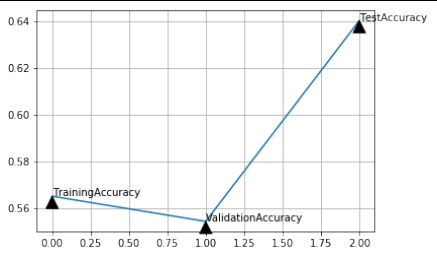
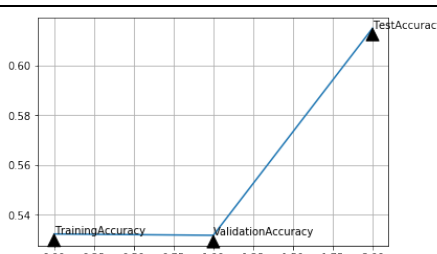
and
$$E(w^*) = \frac{1}{2} (t_n - \hat{t}_n)^2$$

Step 5: Similarly, we will calculate the E_{RMS} for validation and testing data set.

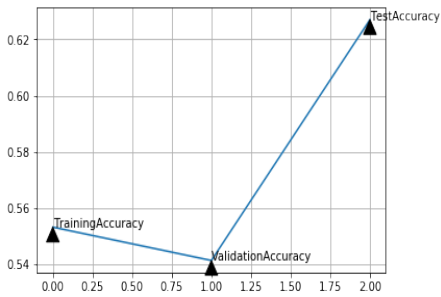
Observations

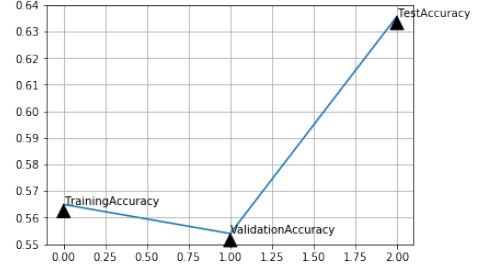
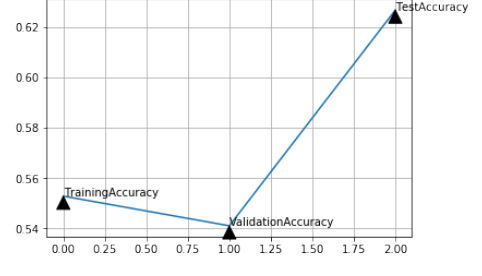
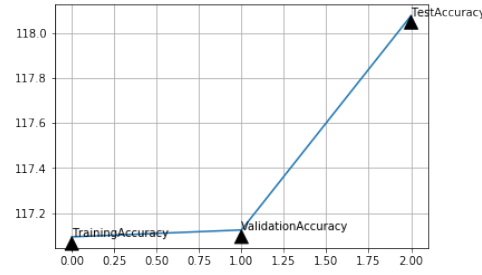
1. Tuning hyperparameters for the closed form solution:

Number of Basis functions (M)	Error for Training data(E_rms Training)	Error for Validation data(E_rms Validation)	Error for Test data(E_rms Testing)	Graph Plots
Number of Basis functions (M)= 10 Change in regularizer (C_Lambda)= 0.03	0.54946940 67137861	0.53842817 41389029	0.62797884 53856765	
Number of Basis functions (M)= 5 Change in regularizer (C_Lambda)= 0.05	0.54948278 34801559	0.53843202 88967555	0.62801160 19948301	

Number of Basis functions (M)= 50 Change in regularizer (C_Lambda)= 0.05	0.54104896 04461131	0.53687257 30935511	0.62004455 45218453	
Number of Basis functions (M)= 1 Change in regularizer (C_Lambda)= 0.09	0.53957325 8711102	0.53612918 44322806	0.61924463 67308256	
Number of Basis functions (M)= 1000 Change in regularizer (C_Lambda)= 0.01	0.53649035 28911144	0.53367007 86512604	0.61701165 63764696	

2. Tuning hyperparameters for the Stochastic gradient descent:

Hyperparameters Tuned	Error for Training data(E_rms Training)	Error for Validation data(E_rms Validation)	Error for Test data(E_rms Testing)	Graph Plots
Weight initialized during initialization= 220 Regularizer(La)= 2 learningRate= 0.01 Size of epochs= 400	0.55324	0.54139	0.627	

Weight initialized during initialization= 100 Regularizer(La)= 9 learningRate= 0.09 Size of epochs= 100	0.56494	0.55398	0.63572	
Weight initialized during initialization= 1 Regularizer(La)= 1 learningRate= 0.01 Size of epochs= 200	0.55284	0.54104	0.62682	
Weight initialized during initialization= 800 Regularizer(La)= 1 learningRate= 0.02 Size of epochs= 10	117.09415	117.1248	118.07643	

Conclusion

We have achieved an optimal model with the minimal fluctuations on changing the hyperparameters for both closed form solution and stochastic gradient descent method.

References

- [1] Closed form solution and SGD implementation code posted in UB Learns.
- [2] Pattern Recognition and Machine Learning Christopher M Bishop
- [3] <http://www.numpy.org/>
- [4] https://en.wikipedia.org/wiki/K-means_clustering

