CSE 574: Introduction to Machine Learning

Atrayee Nag
#50288651
atrayeen@buffalo.edu

Abstract

In this project, we are going to use Machine learning to solve the Learning to rank problem that arises in Information Retrieval process. We will solve the problem by formulating it into linear regression where we map an input vector x to a real-valued scalar target y(x, w). We have used the Closed form Solution to find out the weights and Stochastic Gradient Descent to train our model.

1 General Concepts

1.1 What is Learning to Rank Problem ?

Learning to rank is a sub-area of machine learning, studying methodologies and theories for automatically constructing a model from data for a ranking problem suppose [0, 1, 2]. Learning to rank uses supervised machine learning to discover the best order for a list of items, using features extracted from each item to give it a ranking. It does not provide a definite solution but instead assigns a relative order to each item over another. For web results some of the features are as follows:
1. Whether the page title matches the query
2. How many hits have been there on the page
3. How long is the content of the page
4. Age or popularity of the page
5. Whether the page can be viewed on a phone browser

Learning to rank ties machine learning to search engines is the process of identifying the most important features of document set to the users of those documents, and using those features to tune the search engine to return the best fit documents to each user on each search. Tokens are generated at search time which are looked up from the query at the inverted index, ranks the matching documents and retrieves and returns text associated with it to the user. The diagram below shows the LTR associated with webpages which corresponds to our training , validation and testing data.

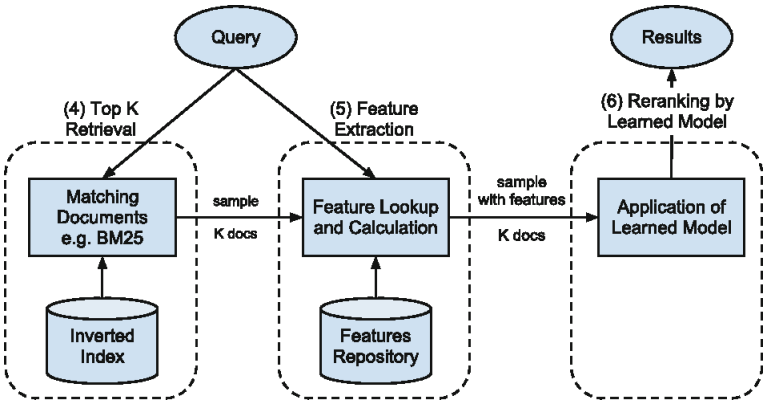

Figure 1: LTR on Search Engines

1.2 What is Linear Regression?

Simple Linear Regression is a type of regression where there is only one independent variable and the dependent variable y is linear to x. The target function is defined as below:

y ($\mathbf{x}$, $\mathbf{w}$) = w_0 + w_1 * x_1

Multivariate linear regression can be defined as a linear combination of input variables, where there are n number of independent variables. The goal of linear regression is find a polynomial function to approximate the target function. The target function for multivariate linear regression is as follows:

y ($\mathbf{x}$, $\mathbf{w}$) = w_0 + w_1 * x_1 + w_2 * x_2 + ... + w_n * x_n

The generalized equation is given by :

y ($\mathbf{x}$, $\mathbf{w}$) = $\mathbf{w}^\top$ $\phi(\mathbf{x})$

where w = (w_0, w_1, .., w_{M1}) is a weight vector to be computed from training samples and ϕ = (ϕ_0, .., ϕ_{M-1}) is a vector of M basis functions.

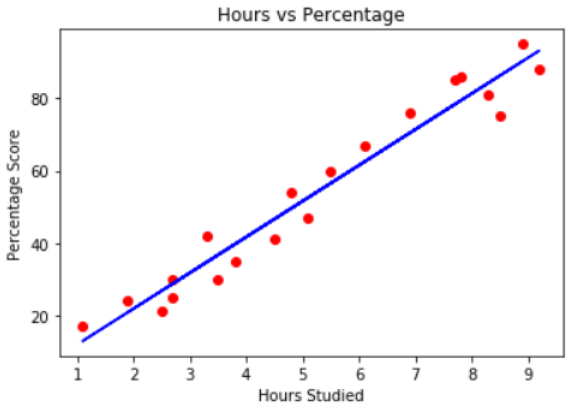

Figure 2: Linear Regression

2 What is Basis Function?

The plain form of Linear Regression is polynomial basis functions, one of big limitation is that they are global functions of the input variable, so that changes in one region of input space affect all other regions. This can be resolved by dividing the input space up into regions and fitting a different polynomial in each region, leading to spline functions. A common type of basis function for such models is the Gaussian basis function. This type of model uses the kernel of the normal (or Gaussian) probability density function (PDF) as the basis function. The equation for a Gaussian radial Basis Function is given as follows:

$\phi(\mathbf{x})$ = $\mathbf{exp}$ (- 0.5 ($\mathbf{x}$ - μ_j))$\sum_j^{-1}$($\mathbf{x}$ - μ_j)

where μ_j is the centroid of the basis function and $\sum_j$ decides how broadly the basis function spreads.

The below diagram displays a graph for Gaussian basis function:

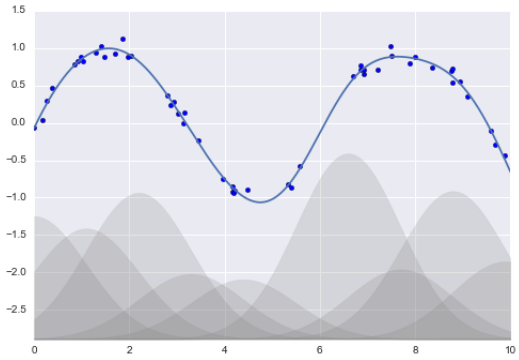

Figure 3: Basis Function

3 Steps involved in the Learning to Rank Problem

There are two steps involved in our Learning to rank problem.

1. We calculate the Root Mean Square Errors using the Closed form solution.
2. We train our model using the Stochastic Gradient Descent

3.1 Closed Form Solution

An equation is said to be a closed-form solution if it solves a given problem in terms of functions and mathematical operations from a given generally-accepted set. For example, an infinite sum would generally not be considered closed-form. However, the choice of what to call closed-form and what not is rather arbitrary since a new closed-form function could simply be defined in terms of the infinite sum.

The closed form solution for calculating the weights without regularization has the form:

$w_{ML} = (\phi^\top \phi)^{-1} \phi^\top \mathrm{t}$

where t = $\{t_1$, .., t_n $\}$ is the vector of outputs in the training data and ϕ is the design matrix

The closed-form solution with least-squared regularization has the below form:

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

where λ is the regularization factor.

```
kmeans = KMeans(n_clusters=M, random_state=0).fit(np.transpose(TrainingData))
Mu = kmeans.cluster_centers_

BigSigma     = GenerateBigSigma(RawData, Mu, TrainingPercent,IsSynthetic)
TRAINING_PHI = GetPhiMatrix(RawData, Mu, BigSigma, TrainingPercent)
W            = GetWeightsClosedForm(TRAINING_PHI,TrainingTarget,(C_Lambda))
TEST_PHI     = GetPhiMatrix(TestData, Mu, BigSigma, 100)
VAL_PHI      = GetPhiMatrix(ValData, Mu, BigSigma, 100)

TR_TEST_OUT  = GetValTest(TRAINING_PHI,W)
VAL_TEST_OUT = GetValTest(VAL_PHI,W)
TEST_OUT     = GetValTest(TEST_PHI,W)

```

```
TrainingAccuracy   = str(GetErms(TR_TEST_OUT,TrainingTarget))
ValidationAccuracy = str(GetErms(VAL_TEST_OUT,ValDataAct))
TestAccuracy       = str(GetErms(TEST_OUT,TestDataAct))
```

The steps involved in the Closed form solution to solve the Learning to rank problems are as follows:

a **Creating the training Dataset:** The training data is divided and a percentage of 80% of it is stored in the numpy array. While creating the dataset we are deleting 5 features which has no change across the dataset. It eliminates values which will give the covariance as 0, thus the inverse cannot be calculated. We further can eliminate features in the dataset which does not add anything to get the data.

b **Clustering the DataSet using K-means clustering:** This is a clustering method which takes in 2parameters: the number of basis functions we want to generate and the random state. We can input an array of centroids if we have previous knowledge of the centroids, in our case we do not have prior knowledge so just initialize it to 0 or None and let K means calculate it using the Euclidean distance.

c **Calculating BigSigma:**
1. Initialize the BigSigma into a 2-d numpy array of size of the features
2. For each row accross the dataset it calculates the variance and stores it in a diagonal matrix Bigsigma which represents the feature vector with respect to itself. We use a diagonal matrix because we are calculating variance for a feature and since 2 features are unrelated the variance will be 0.

d **Get PHI Matrix:** We formulate the Gaussian Radial Basis function which converts each row of the vectors - Data and Mu multiplied by inverse of BigSigma into a single scalar value and store it in the PHI matrix. The equation for calculating the PHI matrix is given by : $\phi_j(\mathbf{x}) = \mathbf{exp}$ (- 0.5 ($\mathbf{x}$ - μ_j))$\sum_j^{-1}$($\mathbf{x}$ - μ_j)

e **Get Weights using Closed Form:** We use the hyper parameter lambda to bound the weights within a certain range , so that even if one of the weight is spiked upto a large value, the lambda brings down the weight, so that no particular feature dominates and result in over fitting. We calculate the weights with the formula Moore-Penrose pseudo-inverse,i.e. the sum of squared inverse with regularization of the PHI matrix. The equation is given by: w^* = $(\lambda \mathbf{I} + \phi^\top \phi)^{-1} \phi^\top$t

f **Calculate Erms for the Dataset** This method returns the root mean quare errors when comparing the training / validation dataset with the target value. We first calculate the residuals i.e. the difference between the actual values and the predicted value and then square it for every dataset and divide it by the total number of datasets and then take a square root which gives us the corresponding name of the square calculation. The equation to find the same is given by:
E_{RMS} = $\sqrt{2E(w^*)/N_v}$
where w^* is the solution and N_V is the size of the test dataset.

3.2 Stochastic Gradient Descent

Stochastic gradient descent (often shortened to SGD), also known as incremental gradient descent, is an iterative method for optimizing the whole dataset at every step. It is a very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. As the algorithm sweeps through the training set, it performs the above update for each training example. Several passes can be made over the training set until the algorithm converges. If this is done, the data can be shuffled for each pass to prevent cycles.

```
W_Now         = np.dot(220, W)
La            = 2
learningRate  = 0.01
L_Erms_Val    = []
L_Erms_TR     = []
L_Erms_Test   = []
W_Mat         = []

for i in range(0,400):
    Delta_E_D     = -np.dot((TrainingTarget[i] - np.dot(np.transpose(W_Now)
    ,TRAINING_PHI[i])),TRAINING_PHI[i])
    La_Delta_E_W  = np.dot(La,W_Now)
    Delta_E       = np.add(Delta_E_D,La_Delta_E_W)
    Delta_W       = -np.dot(learningRate,Delta_E)
    W_T_Next      = W_Now + Delta_W
    W_Now         = W_T_Next

    TR_TEST_OUT   = GetValTest(TRAINING_PHI,W_T_Next)
    Erms_TR       = GetErms(TR_TEST_OUT,TrainingTarget)
    L_Erms_TR.append(float(Erms_TR.split(',')[1]))

    VAL_TEST_OUT  = GetValTest(VAL_PHI,W_T_Next)
    Erms_Val      = GetErms(VAL_TEST_OUT,ValDataAct)
    L_Erms_Val.append(float(Erms_Val.split(',')[1]))

    TEST_OUT      = GetValTest(TEST_PHI,W_T_Next)
    Erms_Test = GetErms(TEST_OUT,TestDataAct)
    L_Erms_Test.append(float(Erms_Test.split(',')[1]))
```

The steps involved in the Stochastic Gradient Descent solution to solve the Learning to rank problems are as follows:

a **Randomize the Weights:** We randomized the weights by multiplying a scalar value with the weights obtained from the closed form weight solution. However the very first step in gradient descent is randomizing the weights and we could have taken new random weights instead of using the weights from the previous solution.

b **Update the Weights:** We process the entire training set in one go and update all the weights by calculating the error multiplied by the learning rate. We train the model in a loop of 400 instead of the total training data because it has been observed after 400 datasets the weights are stagnant and do not change. This method is called Early Stopping.

c **Displaying the Erms values:** We are displaying training, validation and testing data on the fly during iteration. In a real world scenario, training a model could take multiple days, so it's the norm to printing out changes during the iteration as they can monitored in real time.

We are optimizing the weights for next iteration using this formula:

$\mathbf{w}^{r+1} = \mathbf{w}^{r} + \Delta\mathbf{w}^{r}$

where weight update $\Delta\, w^{r} = \eta^{r}\, \nabla E$, w^{r} is learning rate

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

in which ∇E_D = -$(t_n$ - $w^{r\top}\, \phi(x_n)))\phi(x_n)$

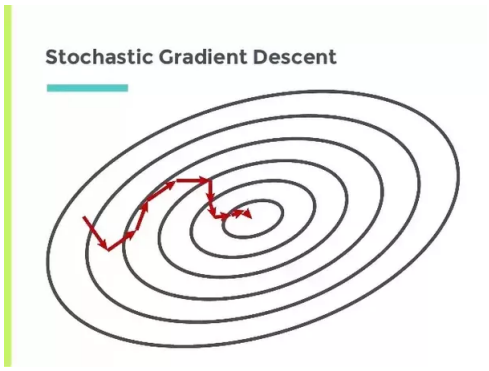

Figure 4: Stochastic Gradient Descent

4 Results on changing the HyperParameters

1. **Case 1: Effect of changing Number of Clusters/Basis Functions**

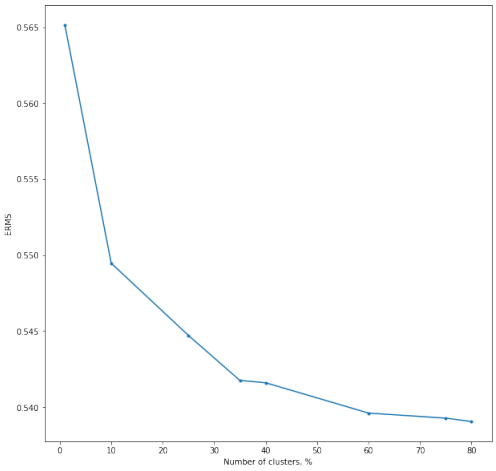

Figure 5: Ermc vs Cluster

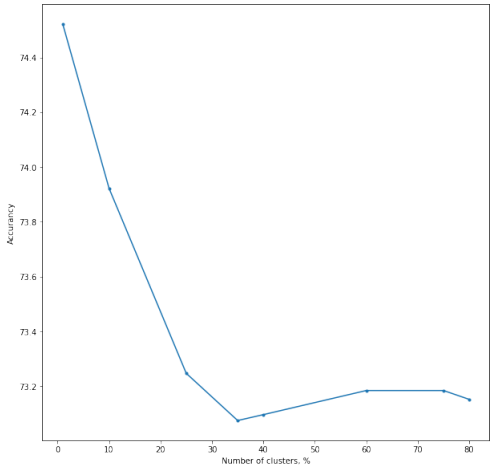

Figure 6: Accuracy Vs Cluster

2. **Case 2: Effect of changing Lambda/Regularization**

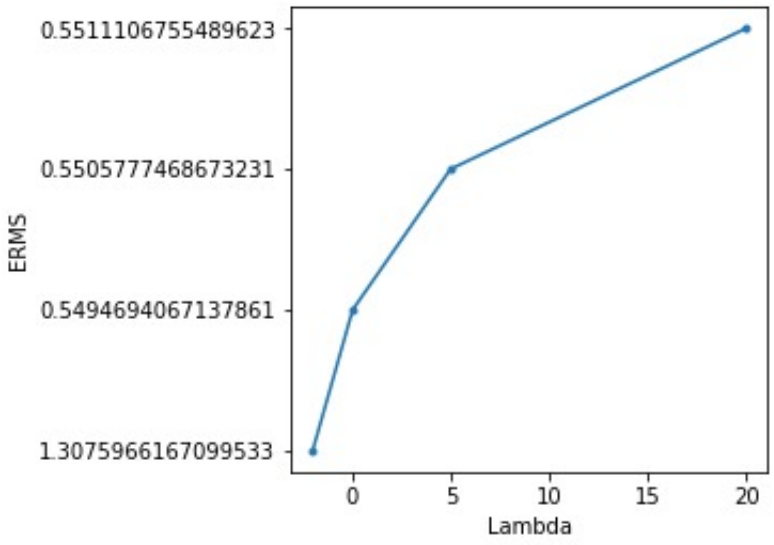

Figure 7: Ermc vs Lambda

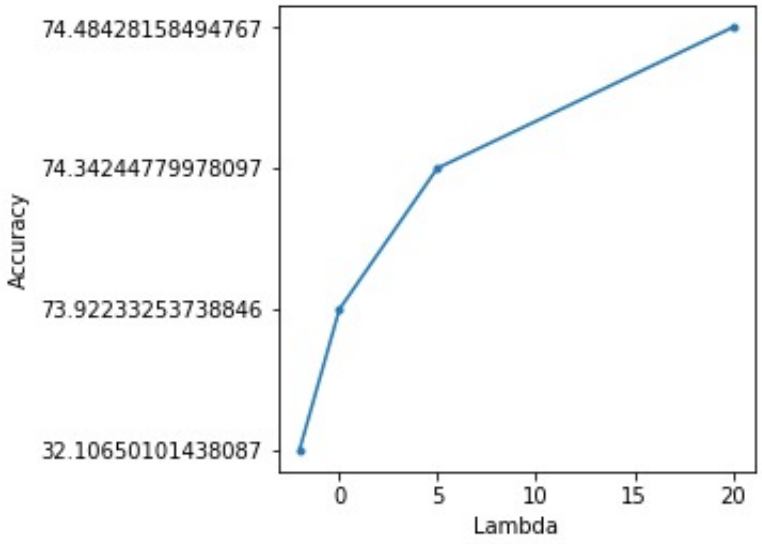

Figure 8: Accuracy Vs Lambda

5 Conclusion

1. The change in the number of clusters is not affecting the Erms or Accuracy by a big margin. However we can observe that there is slight decrease in both Erms and accuracy.

2. We can see from the graph that the increase in lamba decreases both the Erms and accuracy. However the values are stabilizing after a certain threshold.

6 References

The resources that has helped me understand the concepts of the Linear Regression are:

- Linear Regression : http://gwansiu.com/2018/06/17/Linear-Regression/
- Modeling Data with Linear Combinations of Basis Functions : http://www.utstat.utoronto.ca/ radford/sta414.S11/week1b.pdf
- Learning to Rank : https://opensourceconnections.com/blog/2017/02/24/what-is-learning-to-rank/
- Andrew Ng Coursera : https://www.coursera.org/learn/machine-learning