Learning to Rank using Linear Regression

Gagan Suneja
School of Management
University at Buffalo
Buffalo, NY 14260
gagansun@buffalo.edu

Abstract

The report presents implementation of the machine learning problem LeToR(Learning to Rank) dataset using closed-form solution and Stochastic Gradient Descent. The clustering has been implemented using K-means clustering algorithm. Further to this, analysis has been done by fine-tuning the hyper-parameters like the number of clusters, the regularization parameter and the learning rate (for gradient descent).

1 Introduction

The problem of regression involves predicting a real valued output. Linear Regression is an approach of supervised learning to model the relationship between a dependent variable and one or more independent variables. A simple linear regression model involves a linear combination of the input variables to evaluate a dependent variable. For this problem, an extension of linear regression using non-linear functions of the input variables has been used as a simple linear input of variables imposes limitations on the model. Its relationship is given as

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

where w = (w_0, w_1,…,w_{M-1}) is a weight vector to be learnt from training data and ϕ=(ϕ_0, ϕ_1,… ϕ_{M-1}) is a vector of M basis functions or clusters. For this project, Gaussian radial basis functions have been used. Assuming ϕ_0(x)=1 and w_0 as the bias term, each of the basis function ϕ_j(x) convert the input variable x to a scalar value.

$$\phi_j(x) = e^{-\frac{1}{2}(x-\mu_j)\Sigma^{-1}(x-\mu_j)} \tag{2}$$

LeToR Dataset

The LeToR dataset used for this problem is used in Information Retrieval for Learning To Rank problem. The dataset comprises of input vector derived from a query-URL pair and the target value is relevance value (0,1 or 2). Larger the value of the relevance variable, better the match between query and the document.

2 Implementation

The project has been implemented using two methods- closed-form solution and Stochastic Gradient Descent. The three major implementation parts are- *Data pre-processing*, *Closed-form solution implementation* and *Stochastic Gradient Descent implementation*.

Data Pre-Processing

This step involves

i. Transforming the raw data (MGS 2007 dataset) to a CSV file (Querylevelnorm_t.csv for training output and Querylevelnorm_X.csv for training input).
ii. Removing the columns with 0 values (for this dataset, column no-6,7,8,9,10) as they are all 0 values due to which the variance comes out to be zero.

iii. Segregating the data(Size=69623) into Training(N=55699), Testing(N=6961) and Validation(N=6962) data.
   a. Training Data - The pre-processed data (80% of raw data) used to train the model
   b. Validation Data - The data (10% of raw data) used to tune the weights and prevent overfitting. This data is used to compare the predicted output and the actual output for this data and accordingly tune weights in case of error.
   c. Testing Data - The data (10% of raw data) on which the trained and validated machine learning model has to be applied in order to predict the output.

Closed Form Solution Implementation

A closed form solution solves a given problem in terms of functions and mathematical operations from a given generally accepted set. The weights without regularization are calculated as-

$$w_{ML} = (\phi^T \phi)^{-1} \phi^T t \tag{3}$$

and not as $w_{ML} = \phi^{-1} t$ is not possible as the ϕ is not a square matrix and is thus non-invertible

Weights with regularization, $w^* = (\lambda I + \phi^T \phi)^{-1} \phi^T t$ (4)

where t={t_0,t_1,…t_N} and t is the output of the training data and ϕ is the design matrix. With M as the number of basis functions and N as the input training data size,

$$\phi(x) = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) \ldots & \phi_M(x_1) \\ \vdots & \vdots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \phi_M(x_N) \end{bmatrix}_{\mathrm{N\,x}\,M} \tag{5}$$

Following are the sequence of steps followed to obtain a closed form solution

i. Clustering and calculation of μ matrix: This step involves calculation of μ matrix (1xM) using a clustering algorithm. Here, k-means clustering algorithm has been because of the simplicity and its performance on huge dataset.

$$\mu = [\mu_0, \mu_1, \ldots \mu_{M-1}]_{1\mathrm{xM}} \tag{6}$$

ii. Evaluate Big Sigma matrix, also called variance matrix

$$\sum = \begin{bmatrix} \sigma_{11}^2 & 0 & 0 \\ 0 & \sigma_{22}^2 \;\cdots & 0 \\ & \vdots & \\ 0 & 0 & \sigma_{41\,41}^2 \end{bmatrix}_{41\mathrm{X}41} \tag{7}$$

Here, for i ≠ j, $\sigma_{\mathrm{ij}} = 0$ as we are only taking account variance of one feature with respect to itself only.

iii. Train the design matrix given by eqn (5) for N as 55699 and initial M as 10

iv. Calculate the weights as $w_{Tr} = (\phi^T \phi)^{-1}\, \phi^T t$

v. Calculate the error

Error = $\sqrt{\frac{2}{N}(\frac{1}{2}(t - t'\,))}$, where t' is the predicted value

E_{RMS} = $= \left(\frac{2E(w^*)}{N_v}\right)^{\frac{1}{2}}$

vi. If the values in ∑ matrix are small, scale them up by a factor of 200

Stochastic Gradient Descent Implementation

The gradient descent solution, with the learning rate ɲ, iteratively traverses the data points one at a time until a minima(values of weights where the E_{RMS} is minimum) is reached thereby updating the weights w^{τ} using the below relationship-

$$w^{\tau+1} = w^{\tau} + \Delta w^{\tau},\ where\ \Delta w^{\tau} = -ɲ^{\tau} \nabla E$$

where ƞ is the learning rate that directly affects how early the gradient descent reaches the minima. It decides how big the updated should be for each iteration. If ƞ is very high, the gradient descent will have drastic updates which will lead to divergent behaviors. If ƞ is too low, the algorithm will take a lot of time to reach the minima. Hence, optimum learning rate should be set.

Stochastic Gradient iteratively traverses through the data points, one at a time. For each of the iteration, the weights are updated by Δw . Below are the relationships that are used in this solution.

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

$$\text{Where } \nabla E_D = -\left(t_n - w^{(\tau)^T}\phi(x_n)\right)\phi(x_n) \quad \text{and } \nabla E_W = w^{(\tau)}$$

3 Analysis

The analysis has been done by fine tuning the number of clusters M and the regularization parameter λ and the learning rate ƞ.

Table 1: Findings on E_{RMS} and M for λ=0.03 for Closed Form solution

M	E_{RMS} Training	E_{RMS} Testing
10	0.5494	0.6279
15	0.5469	0.6273
20	0.5461	0.6261
30	0.5432	0.6229
40	0.5416	0.6207
60	0.5396	0.6192
80	0.5390	0.6187
90	0.5389	0.6191
100	0.5386	0.6186
110	0.5384	0.6187
120	0.5383	0.6187
130	0.5383	0.6184

Table 2: Findings on E_{RMS} and M for λ=2 and M=10 for Gradient Descent

ƞ	E_{RMS} Training	E_{RMS} Testing
0.01	0.54964	0.62372
0.05	0.55261	0.62384
0.10	0.58417	0.64879
0.20	0.5572	0.63515
0.25	20.78709	20.67601
0.30	12.36029	12.25598
0.40	Overflow	Overflow
0.50	Overflow	Overflow

Table 3: Findings on E_{RMS} λ for ƞ=0.01 for Gradient Descent

λ	E_{RMS} Training	E_{RMS} Testing
0.1	19.13901	18.9634
0.2	13.02633	12.9095
0.3	8.84469	8.76923
0.4	5.96406	5.91115
0.5	3.99004	3.95876
0.6	2.66727	2.65519
0.7	1.78277	1.78512
0.8	1.19949	1.21393
0.9	0.84544	0.88724
1.0	0.66502	0.73303
1.1	0.59434	0.67819
1.2	0.57919	0.67218
1.3	0.5797	0.65508
1.4	0.55707	0.63255
1.5	0.55007	0.62483
1.6	0.54947	0.62367
1.7	0.55061	0.62475
1.8	0.55044	0.62518
1.9	0.54971	0.62387
2.0	0.54964	0.62372
2.5	0.54957	0.62362
3..0	0.5496	0.6246

For closed form solution, the observations have been taken by keeping the regularization parameter (λ) fixed at 0.03 and increasing the number of clusters (M) from 10 to 130. The details are described in the Table 1 and Fig 1.

For Stochastic Gradient Descent, the observations have been taken for finding relation between the learning rate (ɳ) and E_{RMS} by keeping the regularization factor(λ) constant at 2 and for finding relation between the regularization parameter (λ) and E_{RMS} by keeping ɳ constant at 0.01.

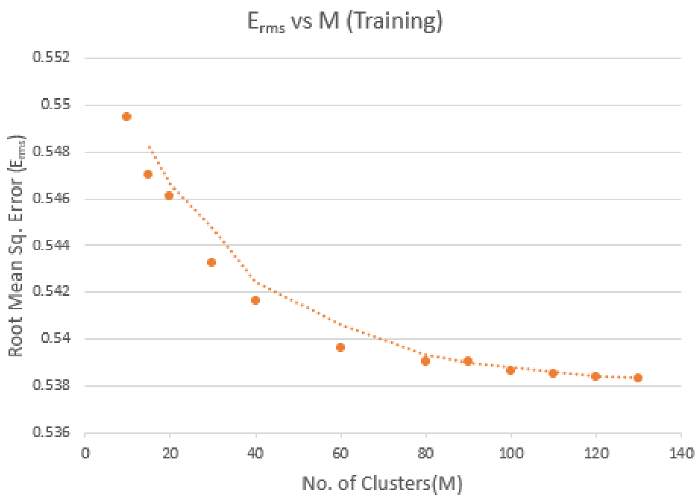

Fig 1: Relationship between E_{RMS} and M (Training)

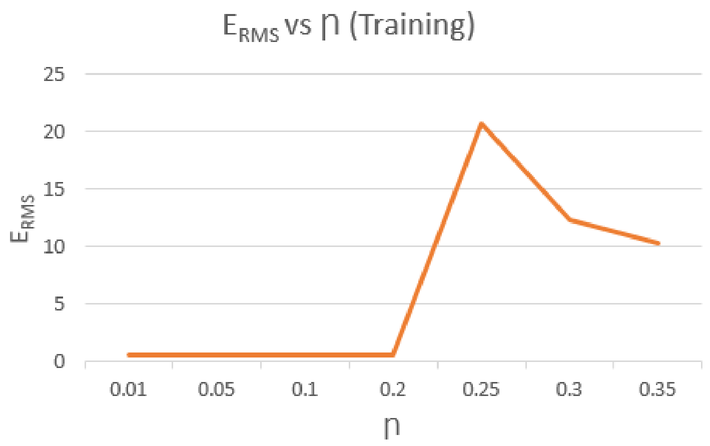

Fig 2: Relationship between E_{RMS} and λ

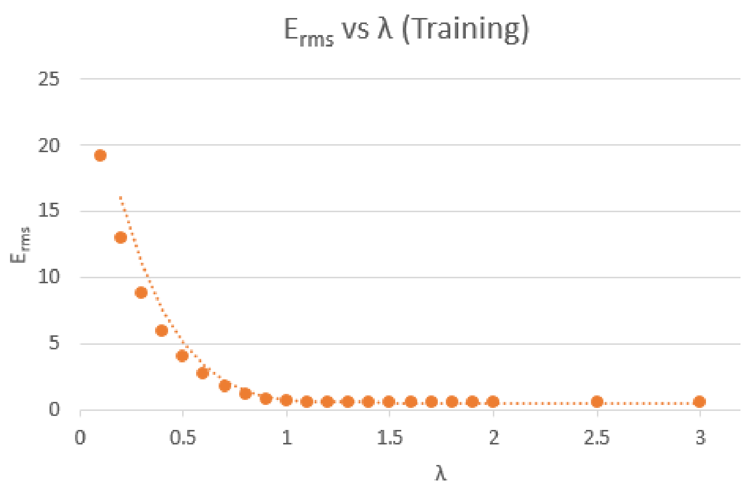

Fig 3: Relationship between E_{RMS} and λ

4 Inference and Conclusion

From Table 1 and Fig 1, it is evident that as the number of basis functions (or clusters -M) increase, the E_{RMS} decreases and reaches a near constant value after 100 clusters onwards. For λ=0.03, the minimum E_{RMS} of 0.5386 is achieved at M=100 with an accuracy of 73.1180.

From Table 2 and Fig 2, it is seen that on increasing the learning rate, after a certain value, the stochastic gradient descent behaves erratically. It can be inferred that optimum learning rate should be chosen.

From Table 3 and Fig 3, it is seen that low values of regularization factor will lead to high E_{RMS}. As we increase it, E_{RMS} decreases and reaches almost a constant value of ~0.62362 at λ=2.5 for η=0.01.