Machine Learning CSE574

David Jegan Abishek Dominique
Person no 50290785
davidjeg@buffalo.edu

Abstract

The report provides a detailed insight on how Linear regression can be used to solve Learning to Rank problem. We leverage two methodologies., Closed form and Gradient Descent.

1 Introduction

1.1 Linear Regression

This is a supervised learning model, where a linear relationship between the input variables (features X) and target variable (t) is assumed.

$$Target = \sum_{j=0}^{K} (Weight_j * Input^j) + Bias$$

where, − K is the order of the polynomial, − Input $X = (x_1, x_2.., x_N)^T$, − Target output $t = (t_1, t_2.., t_N)^T$ and Bias is the noise added to the linear relationship.

1.2 Learning to Rank

The learning to rank algorithm, gives the closest match between a query and a document. The target is a relevance label of values 0,1 and 2, where 2 denotes the best relation between the query and the document. Though the values in the training target are scalar and discrete (0,1,2), we consider linear regression to obtain continuous value rather than a discrete on, to increase the difference factor between the documents and thereby ranking the pages.

1.3 Approach

There are three stages in this problem

1. Create training data
2. Closed form solution
3. Stochastic gradient descent solution

2 Methodology

2.1 Data Preparation

For the algorithm to learn we provide a training dataset. This data is crucial for the prediction as the model considers this data as Ground truth and tries to learn.

From the learning to rank (LeToR) dataset (Querylevelnorm.txt), we extract the relevance label and feature set of 46 values for 69623 entries. The feature set is normalized to be between [0,1] to mitigate computational complexity. We drop 5 of the 46 rows, because all the entries are zero, and it is not possible to determine the inverse of the covariance for these 5 rows.

2.1.1 Feature Data

We construct a design matrix for the three datasets. These three design matrices share the same μ and Σ, but the data entries are unique for each set.

2.1.2 Target Data

We divide the target values into 80%, 10% and 10% as training, validation and testing result set respectively, in such a way that the sets do not overlap. These values are then compared with the obtained results to compare the accuracy of our model.

2.2 Closed form solution

In the Closed form approach, we build a linear regression model, which leverages the entire calculated weight and phi (basis function) to determine the target.

Thus, the target can be determined using the equation,

$$t = w^T \Phi(x)$$

where $\Phi(x)$ is the basis function design matrix constructed using x (input set), w is the weight and t is the output (target).

2.2.1 Weight and Error function

Both the input and the target is provided in the training phase. So to optimize the model, we need to improve the accuracy of the weights, and weights are dependant on the error function. Thus inorder to change the weights, we reduce the error value between target and prediction for each input x.

Considering Φ as basis function design matrix, we have the formula for w as,

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

We calculate the weights based on the training dataset and run our model against the validation and the test cases, to check the accuracy of our model.

2.2.2 Basis Function

Basis function helps to define the non-linear relationship between the target and input. The weights are different for each basis function. For the input vector x, the basis function is expressed as $\phi_j(x)$

$$Target = \sum_{j=0}^{M-1} (Weight_j * Basisfunction(Input)^j)$$

here, M is number of the basis function, Weight $w = (w_0, w_1.., w_{M-1})^T$ and Basis function $\Phi = (\phi_0, \phi_1.., \phi_{M-1})^T$

2.2.3 Gaussian Radial Basis Function

$$\phi_j(x) = e^{(-0.5(x-\mu_j)^T \Sigma^{-1}(x-\mu_j))}$$

The Gaussian basis functions are chosen because it is non-zero over a very small interval and provide a sparse design matrix which helps in quick computation.

We build Σ (covariance) matrix with the help of BigSigma functionality. Similarly μ is formulated by leveraging k-means clustering algorithm and x is extracted from the input 41 feature (R^{41}) dataset. The final size of ϕ_j is M*N, where M is the number of basis functions and N is the number of samples.

The ϕ_j matrix is built for train, validation and test datasets.

2.2.4 Big Sigma Σ

The variance is a scalar value. So in order to make matrix multiplication feasible, we expand the variance vector (varVect) into $41x41$ matrix, with only has diagonal filled and rest is zero entries. Thus the bigsigma contains only diagonal entries (variance) and the rest are zero (covariance).

2.2.5 Mu matrix μ

The mu matrix contains the centroid of the clusters (position of the basis functions). It consists of M centroids and M individual clusters. The centroids are determined using K-means clustering algorithm. Thus there are M*41 entries in the Mu matrix, where M is the number of the basis functions and 41 is the input feature count.

2.2.6 Design Matrix

In a design matrix of dimensions, each row corresponds to the N input and each column corresponds to the M basis function. The design matrix is tabulated for all three inputs (Train/validation and test)

2.2.7 Pseudo Inverse

We know that ϕ_j is not a square singular (invertible) matrix. So we use Moores penrose pseudo inverse, to find the inverse of the design matrix.

$$Inverse = (\Phi^T \Phi)^{-1} (\Phi^T)$$

An inverse of design matrix N * M is given by $\Phi_{nj} = \Phi_j(x_n)$

2.2.8 Regularization

In order to avoid overfitting, we consider a regularization parameter λ, This is done while finding the weights. The Lambda being a scalar, is expanded to an M*M matrix using an Identity matrix for facilitating matrix multiplication. The parameter reduces overfitting by adding bias to our estimate.

$$w^* = (\lambda I + \Phi^T \Phi)^{-1} \Phi^T t$$

Also, regularizer is responsible for reducing the weight values and maintaining an optimal weight. This change is consistent with respect to the weights.

2.2.9 E_{RMS}

This methods calculates the accuracy of our prediction with the actual target by identifying how different our model has deviated from the actual. If Erms equals 0 for train set, then it is overfitting. Erms is the square root of the sum of the difference square, divided by total predictions

2.2.10 Accuracy

Based on the number of matches of Actual and Predicted values, we find the accuracy of the model.

2.3 Stochastic gradient descent

In the Stochastic gradient descent approach, we build a linear regression model, which traverses the dataset incrementally, updating the values of weight and phi (basis function), calculating the minimal error function after each epoch to predict the target. Thus our algorithm moves towards lower error, updating weights accordingly to find the global cost minimum.

Thus, the target can be determined using the equation,

$$t = w^T \Phi(x)$$

where, $\Phi(x)$ is the basis function design matrix constructed using x (input set), w is the weight and t is the output (target).

2.3.1 Weight

Both the input and the target is provided in the training phase. So to optimize the model, we need to improve the accuracy of the weights, and weights are dependant on the error function. Thus in-order to change the weights, we reduce the error value between target and prediction for each input x. Therefore, the weight in next recursion is the sum of the weight in current recursion and an additive ∇ value.

Weights can be initialized randomly, so we consider the weights obtained from the previous stage. Considering W_{old} as weight obtained in a step i, we have the formula of W_{new} for step i+1 as,

$$W_{new} = W_{old} + \nabla W$$
$$\nabla W = -\eta \nabla E$$
$$W_{new} = W_{old} - \eta \nabla E$$

Here, η is the learning rate, and negative (-) sign denotes that the graph descends towards minima.

So we multiply the negative learning rate with ∇E and sum it with current weight (W_{old}) to obtain the new weight (W_{old})

2.3.2 Error function

Gradient descent iteratively minimizes the error function. The value of gradient on Error function gives us the direction of downhill. So we must differentiate our Error function to get desired solution.

The error function is defined as,

$$E = 0.5(t_i - w^T \phi(x_i))^2 + 0.5\lambda w w^T$$

where, t_i is the trainingtarget value for i, w is the current weight, $\phi(x)$ is the basis function design matrix for input x and λ is the regularization parameter.

We know that, differentiating E with respect to W, gives 0.

$$\nabla E = -(t_i - w^T \phi(x_i))\phi(x_i) + \lambda w$$

We calculate the weights based on the training dataset, and run our model against the validation and the test cases, to check the accuracy of our model.

This term is multiplied with the negate of learning rate, in each recursion.

2.3.3 Gaussian Radial Basis Function

Similar to the closed form solution, Basis function helps to define non-linear relationship between the target and input.

$$\phi_j(x) = e^{(-0.5(x-\mu_j)^T \Sigma^{-1}(x-\mu_j))}$$

On the same lines, we build Σ (covariance) matrix with the help of BigSigma functionality. μ is formulated by leveraging k-means clustering algorithm and x is extracted from the input 41 feature (R^{41}) dataset. The final size of ϕ_j is M*N, where M is the number of basis functions and N is the number of samples.

The ϕ_j matrix is built for train, validation and test data sets.

2.3.4 Learning Rate

The steepness of the downhill is decided by the learning rate. We consider $\eta = \frac{1}{\sqrt{i}}$ because we want our model to descends fast initially, and gradually slow down its desent.

2.3.5 Regularization

Inorder to avoid overfitting, we consider a regularization parameter (La), This is done while finding the weights. We have assumed the La variable randomly. The parameter reduces overfitting by adding bias to our estimate.

2.3.6 E_{RMS}

This methods calculates the accuracy of our prediction with the actual target. Erms is the square root of the sum of the difference square, divided by total predictions

2.3.7 Accuracy

Based on the number of matches of Actual and Predicted values, we find the accuracy of the model.

3 Findings

3.1 Closed form

3.1.1 Changing the number of basis function (M)

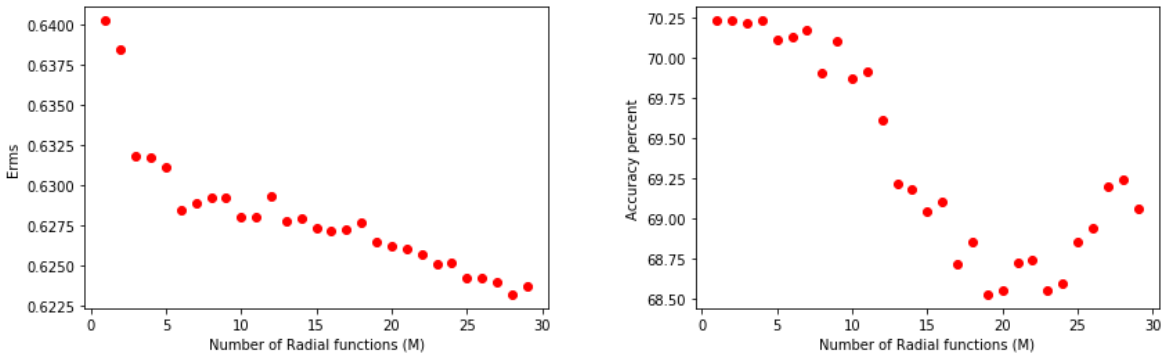

Figure 1: Error value (left) and Predication accuracy (right) for M

We observe that increasing the number of radial basis functions decreases the Error rate to aullar very small degree, and the impact of accuracy of test dataset is minimal.

With more number of Basis functions, the number of clusters increase, variance and Big sigma reduces. Thus the basis function value increases and has a negative impact on the accuracy of the algorithm.

3.1.2 When the Regularization parameter changes

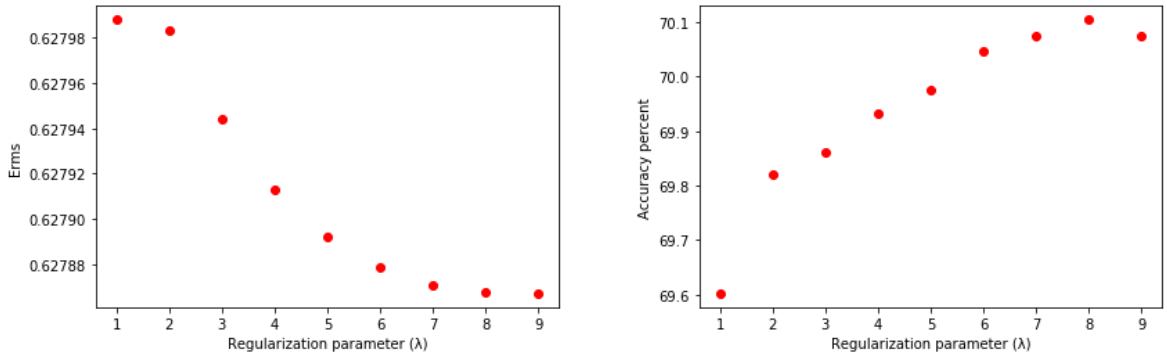

Figure 2: Error value (left) and Predication accuracy (right) for Lambda

We observe that by increasing the value of regularization parameter, the error rate decreases and the accuracy of test dataset prediction increases.

With increase in regularization parameter leads to underfitting. Thus there would be a decrease in the prediction accuracy for large Lambda values. Also, small values of lambda leads to overfitting and gives low accuracy.

3.1.3 When Variance matrix value changes

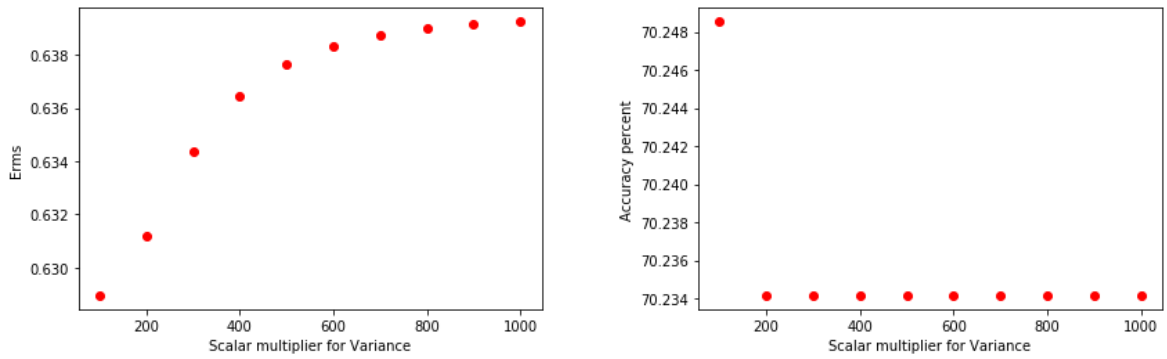

Figure 3: Error value (left) and Predication accuracy (right) for variance scalar multiplier

From the graph, we find that as the Scalar multiplier increases, the Error rate increases. This is because the randomness increases drastically as the multiplier increases, and thereby reduces the accuracy.

3.2 Stochastic gradient descent solution

3.2.1 When Learning rate change

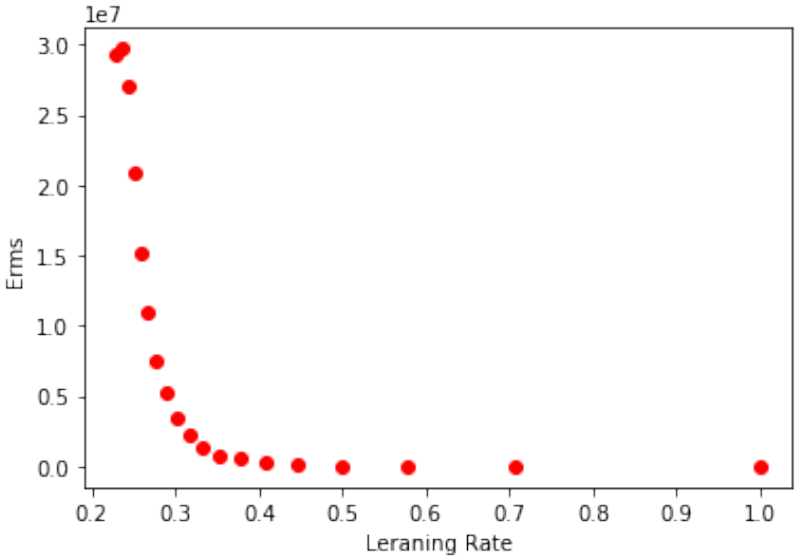

Figure 4: Predication accuracy for Learning Rate

In our algorithm, we consider learning rate to be the inverted square root of each iteration. So initial the learning rate was high and so is the error rate. But as the iteration increases, the learning rate decreases, and the error rate decreases accordingly.

3.2.2 When the Regularization parameter changes

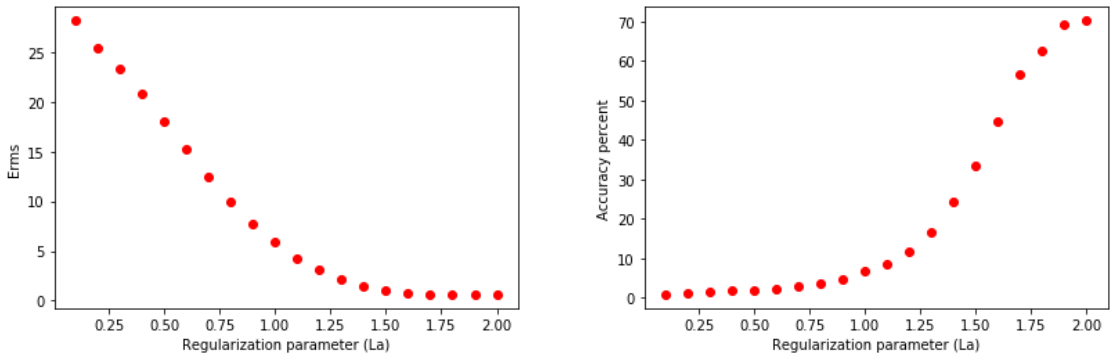

Figure 5: Error value (left) and Predication accuracy (right) for Lambda

We observe that by increasing the value of regularization parameter, the error rate decreases and the accuracy of test dataset prediction increases.

With increase in regularization parameter leads to underfitting. Thus there would be a decrease in the

prediction accuracy for large Lambda values. Also, small values of lambda leads to overfitting and gives low accuracy.

3.2.3 Changing the number of basis function (M)

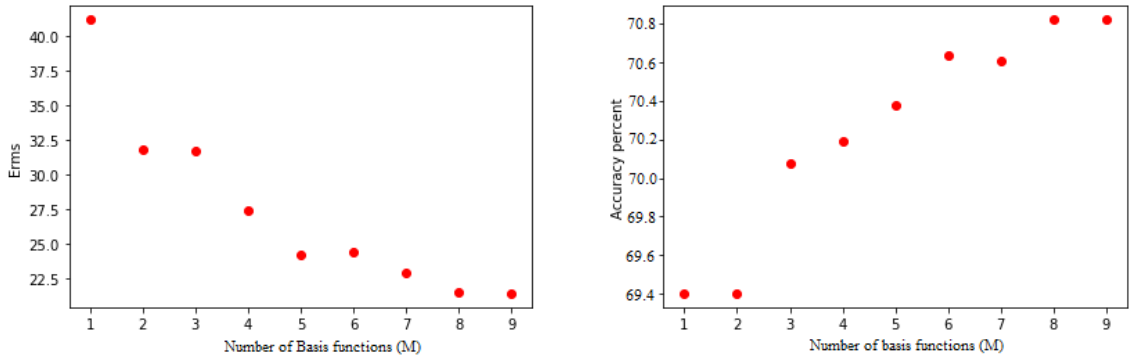

Figure 6: Error value (left) and Predication accuracy (right) for M

We observe that increasing the number of radial basis functions decreases the Error rate to a very small degree, and the impact of accuracy of test dataset is minimal.

With more number of Basis functions, the number of clusters increase, variance and Big sigma reduces. Thus the basis function value increases and has a negative impact on the accuracy of the algorithm.

3.2.4 Number of iterations

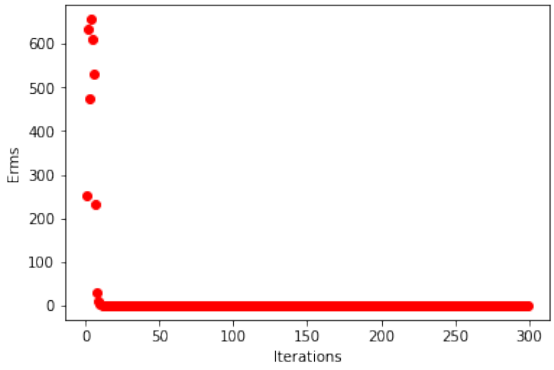

Figure 7: Error value based on iterations

We observe that the error value drastically reduces before the 50th iteration ie., our model has found the global cost minima. So we reduce the iterations to 300.

3.3 Consideration factor

- We find that closed form solution cannot hold good, when the input data size is too large, as calculating the phi and phi inverse is a computational overhead

- Since SGD is iterative, it can be a good choice when data is large. But when the training set is larger the computation increases and algorithm becomes slower.

References

1. https://cedar.buffalo.edu/ srihari/CSE574/Chap3/3.2-GradientDescent.pdf
2. https://cedar.buffalo.edu/ srihari/CSE574/Chap3/3.1-Regression-BasisFns.pdf
3. https://www.cs.toronto.edu/ rgrosse/courses/csc321_2018/readings/L02%20Linear%20Regression.pdf
4. http://cs229.stanford.edu/notes/cs229-notes2.pdf