Project 1.2: Learning to Rank using Linear Regression
Introduction to Machine Learning

Yash Narendra Saraf
UB Person No. 50290453
MS Computer Science and Engineering
School of Engineering and Applied Science
ysaraf@buffalo.edu

Abstract

The project is used to understand linear regression using the LETOR dataset. The dataset contains a total of 46 features and a target value. So the linear regression concepts are understood along with the Radial Basis Functions. As the feature space is very large. We create the model using the Gradient Descent method and closed form method.

1 LETOR Dataset

The LeToR training data consists of pairs of input values x and target values t. The input values are real-valued vectors (features derived from a query-document pair). The target values are scalars (relevance labels) that take one of three values 0, 1, 2: the larger the relevance label, the better is the match between query and document. Although the training target values are discrete we use linear regression to obtain real values which is more useful for ranking (avoids collision into only three possible values).

2 Linear Regression

Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. Its used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories (e.g. cat, dog). Here we have 46 feature vector. This feature space is reduced to 10 features using the Radial basis function.

3 Radial Basis Function

In multidimensional regression X = RD. The easiest way to attack the regression problem is to look for f in a finite dimensional space of functions spanned by a given basis. In other words, we specify a set of functions 0, 1, . . . , P from X to R and look for f in the form of a linear combination.

We can have multiple types of Basis functions like the simple linear regression, polynomial regression and the Gaussian RBF'S. Here we have used the Gaussian RBF's. The formula for the same is as follows -

$$\phi = e^{-1/2*(x-\mu)^T*\Sigma^{-1}*(x-\mu)}$$

Here μ denotes the mean matrix, x denotes the input matrix and the Σ denotes the covariance matrix.

4 Closed Form Solution

We know that the target can be evaluated using the equation.

$$t = w^t * \phi(x)$$

So to find out the weights vector it is necessary to findout the inverse of the design matrix. Since the design matrix has very high dimensionality and also the matrix is not a square matrix. We need to use the Moore Penrose Inversion i.e. the pseudoinversion of the matrix which gives generalization of the inversion matrix. So to make sure we can find inversion we need to make sure that none of the columns has zero variance. The zeros variance is an important task as the matrix is not invertible even if one of the rows or columns has zero variance. Using the following moore penrose formula we can obtain the weight matrix. This is then used to calculate the output of for all the three sets i.e. validation, testing and training. Here it does not make sense to have validation split seperately. Still a validation split has been maintained so as it is being used in the Stochastic Gradient Solution.

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

5 Stochastic Gradient Descent

This is a linear regression solution uses the Stochastic Gradient method to update the weights after each iteration. Over here the output is calculated based on some random predetermined weights, these weights are then being updated after every iteration. Here the output is being compared to the target value and and RMS value is calculated. To reduce the error value the gradient of the error is calculated and based on some constant factor the weights are then changed. The constant factor is known as the learning rate. This is a hyperparameter and should be adjusted carefully. The stochastic gradient equations are explained below.

$$w_i = w_i + \eta \Delta w_i$$
$$\delta E / \delta w_i = \delta / \delta w_i [1/2 * (target - output)^2]$$
$$\delta E / \delta w_i = -\phi(x_i)[target - w^T \phi(x)]$$

6 Regularization

The model when trained just by using gradient descent may start to overfit the testing data. To avoid overfitting various techniques are used i.e. using regulariaztion with weights and using the validation data split. The regularization term is added to the error and the sum is then reduced which makes sure that the weights also stay within the bounds and avoids overfitting. Validation is usually used for early stopping where we check the validation set after every few iterations of the training set. As the validation accuracy starts to decrease for few significant iterations, it means that the model has started to overfit the training set and the computations need to be stopped.

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

The above equation denotes the total error term.

$$E(w) = E_D(w) + \lambda E_W(w)$$
$$E_w(w) = 1/2 * w^T w$$

The regularization term has been added to the Error term to make sure the model does not overfit and the weights do not go out of bound. The regularization also keeps the weights to be minimum.

The closed form solution with regularizer looks like-

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

7 Applying Linear Regression to Letor Dataset

First all the data is loaded from the csv files into the numpy arrays. The variance of 5 rows is 0 as all the terms in those columns is 0. This basically implies that those features are not important and has the same value no matter what the output. So removing these features do not harm the dataset. It is necessary to remove the 0 variance columns as the closed form solution requires the inversion of the matrix which is not possible if an entire column has 0 variance.

Then the data is being split into multiple sets, i.e. the validation set, training set, and the testing set. The data split is necessary for different parts of the stochastic gradient process.

Then using the Kmeans clustering the cluster centers are found out for the 46 feature space. This is necessary to find out the Radial basis functions. Also the variance for all the features are calculated.

Using the variance matrix and the cluster centers the new design matrix is created for all the sets, i.e training set, testing set, and the validation set. These design matrices act as the new dataset on which simple linear regression is applied. This was done to reduce the feature space so that the computations are not expensive.

Now since we have the design matrices the closed form solution for the testing data has been found out using the moore penrose inversion. This returns the weights values. Now these weights can be used to find the outputs and accuracy for all the datasets, also the ERMS values for all the datasets has been calculated. This concludes the part of the closedform solution.

Now we start with the stochastic gradient descent solution. In this first we initialize the weights to some random value. Also some other constants like learning rate and the regularization constant has been defined.

Now we start the execution of the for loop for multiple datapoints. Here we first calculate the gradient value. The equation for the gradient is as follows.

$$\delta E / \delta w_i = -\phi(x_i)[target - w^T \phi(x)]$$

Then we find out the regularization term. Now the gradient and the regularization terms are added together. Now we have the entire delta. The weights are then updated by adding the scaled value of delta to the old weights. The scaling factor used here is called as the learning rate.

$$\delta = \delta E / \delta w_i + \lambda w$$
$$w_{new} = w_{old} - \eta \delta$$

In each iteration we are also finding the ERMS and accuracy for the training and validation dataset. Now this is to make sure that the model is getting trained. Also the early stopping concept can be applied if the model starts to overfit the given training dataset.

$$\sqrt{2E(w^*)/N_v}$$

where w* is the solution and N^v is the size of the test dataset

After the training over several datapoints the model is tested on the testing dataset and the plots for accuracy and ERMS are made. Accuracy is calculates by rounding off the values predicted using the regression value and comparing with the discrete values.

8 Observations

8.1 Closed Form Solution

For the closed form solution varying the regularization term is not causing a huge change in the accuracy of the model. The observations for multiple values are shown below. The model

Table 1: Tabulation of results for Closed Form Solution, First the Regularization constant is varied, then the Number of Basis functions and then the Learning Rate, Here M denotes number of basis functions, Lambda denotes the regularization constant and the eta denotes the learning rate.

				ERMS			ACCURACY	
Lamda	M	Eta	Test	Train	Validation	Test	Train	Val
0.01	10	0.01	0.626877	0.548765	0.537224	69.54461	73.53992	74.27463
0.03	10	0.01	0.626924	0.548775	0.537262	69.6308	73.55428	74.18845
0.3	10	0.01	0.627157	0.548957	0.537798	69.90375	73.81641	74.40391
3	10	0.01	0.627672	0.550013	0.539397	70.0905	74.30115	74.96409
0.03	1	0.01	0.639412	0.564271	0.553602	70.23416	74.52198	75.17955
0.03	5	0.01	0.631065	0.552359	0.540945	70.01868	74.38015	75.05027
0.03	10	0.01	0.626924	0.548775	0.537262	69.6308	73.55428	74.18845
0.03	40	0.01	0.622165	0.537877	0.542855	68.95561	73.08569	73.39845
0.03	100	0.01	0.618975	0.538585	0.535592	69.09927	73.16828	73.28354
0.03	10	0.001	0.626924	0.548775	0.537262	69.6308	73.55428	74.18845
0.03	10	0.01	0.626924	0.548775	0.537262	69.6308	73.55428	74.18845
0.03	10	0.1	0.626924	0.548775	0.537262	69.6308	73.55428	74.18845
0.03	10	1	0.626924	0.548775	0.537262	69.6308	73.55428	74.18845
0.03	10	10	0.626924	0.548775	0.537262	69.6308	73.55428	74.18845

provides an accuracy of 73.83 percent when no regulariaztion term has been used. Using a small regularization value of 0.2 increases the model accuracy to 73.98 i.e. not a large increase.

So to test for extremes, the regularization term when kept to very large number like 1000000 gives accuracy of 74.52198423670085 on the training dataset i.e. the number of zeros in the training dataset. Over here the model is predicting zeros for all the inputs. Since the percentage of zeros in the training data is high we are getting the accuracy to be 74.52198423670085 percent.

Varying the number of clusters is also not affecting the accuracy by a huge margin. Increasing the number of clusters should have a positive impact on the accuracy but it doesn't. Here the accuracy remains almost the same. Since the dataset is so skewed with most of the terms as 0's. When we use the number of clusters as one the model give the accuracy of 74.52198423670085 i.e. predicting 0 for all the inputs. So to have a cluster size of 10-20 seems to be ideal, looking at the results of the predicted and target values and the accuracy of the model.

The closed form solution is providing almost 95-100 percent accuracy when it comes to zeros classes whereas the accuracy of around 10 percent for 1's but accuracy of strictly 0 for 2's. So the net accuracy of about 70 - 74.52198423670085 percent is observed on the training dataset. The testing accuracy is about 70 percent for all the variations.

8.2 Stochastic Gradient Solution

Here some changes have been made to the code i.e. added the bias along with the 10 RBF features, and varied the parameters like learning rate, number of basis function and the regularization term. The effect of each of the parameters is discussed over here.

The bias term did not cause any change. As we had multiple gaussian curves to plot our regression curve the bias term hence did not cause a lot of difference. Bias was added as the extra feature to make sure that there is a factor independent from the input features.

The second parameter varied is the number of the basis function.Since the target values for most of the data, approximately 75 percent is 0. So the model gets saturated at 74.52198423670085 percent for the training dataset as the number of zeros in the training data is that much of the total. After checking for individual classes I noticed that all the predictions are 0, i.e. the accuracy thus is depending upon the percentage of zeros of the dataset.

Also the learning rate is varied between the 0.001 to 10 logarithmically. At higher values i.e. greater than 1 the ERMS is diverging instead of converging and after a few hundred iterations reaches a value of order of 10 to power 150 and the loop breaks as this is outside the bounds of math library. So the model starts to overshoot for higher values of learning rate even if the accuracy is still maintained

Table 2: Tabulation of results for SGD Solution, First the Regularization constant is varied, then the Number of Basis functions and then the Learning Rate, Here M denotes number of basis functions, Lambda denotes the regularization constant and the eta denotes the learning rate.

			ERMS			Accuracy		
Lambda	M	Eta	Test	Train	Val	Test	Train	Val
0.01	10	0.01	0.725204	0.56516	0.55308	70.23416	74.52198	75.17955
0.03	10	0.01	0.734636	0.56134	0.55087	70.23416	74.52198	75.17955
0.3	10	0.01	0.730047	0.55948	0.54907	70.23416	74.52198	75.17955
3	10	0.01	0.739713	0.5646	0.55373	70.23416	74.52198	75.17955
0.03	1	0.01	0.682516	0.56569	0.55483	70.23416	74.52198	75.17955
0.03	5	0.01	0.716559	0.56596	0.55471	70.23416	74.52198	75.17955
0.03	10	0.01	0.72228	0.5589	0.54826	70.23416	74.52198	75.17955
0.03	40	0.01	0.711564	0.56594	0.55333	70.20543	74.51839	75.17955
0.03	100	0.01	0.742828	0.57159	0.56151	70.23416	74.52198	75.17955
0.03	10	0.001	0.638624	0.57058	0.561	70.23416	74.52198	75.17955
0.03	10	0.01	0.757422	0.57365	0.56399	70.23416	74.52198	75.17955
0.03	10	0.1	0.727704	0.55765	0.54647	70.23416	74.52198	75.17955
0.03	10	1		Starts	To	Overshoot		
0.03	10	10		Starts	To	Overshoot		

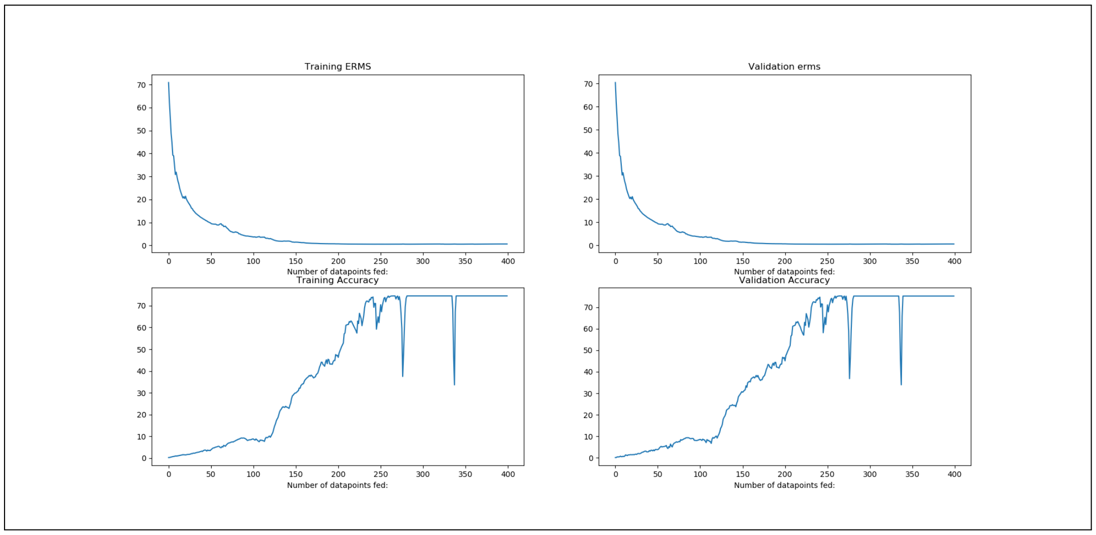

Figure 1: Summary - a-Training ERMS Vs Number of Datapoints, b-Training Accuracy Vs Number of Datapoints, c-Validation ERMS Vs Number of Datapoints, d-Validation Accuracy Vs Number of Datapoints

at 74.52198423670085 percent. This accuracy does not mean anything as the model is now blindly predicting 0 for any inputs. At the lower learning rates the ERMS curve saturates after a few hundred iterations and also the accuracy settles at about 75 percent.

When we vary the regularization split no significant change is noticed. As the model is always predicting 0 no matter the input the regularization term is not causing any change. For numerous values of regularization constant the outputs are noticed. There seems to be no change in the accuracies or the ERMS values.

The Figure 1 shows the validation and accuracy plot for the training and validation data. The weights get largely affected whenever the model encounters any 1 or 2 as target instead of a 1. So whenever there are multiple 1's or 2's together there is a dip in the graph for accuracy as the weights have been altered in such a way that the predictions are getting flawed. And when it is encountering only 0's the accuracy is still at 74.5 percent i.e the amount of 0's in the dataset.

```
——————————————————-
Shape of Raw feature data(40, 69623)
Shape of Raw Target data(69623,)
——————————————————-
Shape of Training Target data(55699,)
Shape of Training feature data(40, 55699)
——————————————————-
Shape of Validation target data(6962,)
Shape of Validation feature data(40, 6962)
——————————————————-
Shape of Test target data(6961,)
Shape of Test feature data(40, 6961)
——————————————————-
Shape of feature center matrix(10, 40)
Shape of Variance matrix(40, 40)
Training design matrix(55699, 11)
Shape of weights calculated(11,)
Validation design matrix(6962, 11)
Testing design matrix(6961, 11)
UBITname = ysaraf
Person Number = 50290453
——————————————————-
——————LeToR Data————————
——————————————————-
——-Closed Form with Radial Basis Function——-
——————————————————-
Number of Basis Function: 10
Regularizer Constant (Lambda) :0.03
E_rms Training = 0.5487753447995469
Training Accuracy = 73.55428284170273
0 acc: 96.16700395104559, 1 acc: 9.350279975113324, 2 acc: 0.0
E_rms Validation = 0.5372618841518344
Validation Accuracy = 74.18845159436944
0 acc: 96.67558272831486, 1 acc: 7.564841498559078, 2 acc: 0.0
E_rms Testing = 0.6269242202315218
Testing Accuracy = 69.63080017238903
0 acc: 96.17508692984251, 1 acc: 9.738079247817327, 2 acc: 0.0
Training Target: 0 ,Prediction:-0.0142679157816
Train Accuracy :74.52198423670085 ,Train ERMS:0.635422553111135
0 acc: 100.0, 1 Acc: 0.0, 2 Acc: 0.0
Val Accuracy :75.17954610744039 ,Val ERMS:0.6202428370856816
E_rms Testing = 0.7326933387224001
Testing Accuracy = 70.23416175836805
————-Gradient Descent Solution——————
Number of Basis Function: 10
Learning Rate used: 0.01
Regularization constant: 0.03
Train Accuracy :74.52198423670085 ,Train ERMS:0.635422553111135
0 acc: 100.0, 1 Acc: 0.0, 2 Acc: 0.0
Val Accuracy :75.17954610744039 ,Val ERMS:0.6202428370856816
E_rms Testing = 0.7326933387224001
Testing Accuracy = 70.23416175836805
E_rms Training = 0.56624
E_rms Validation = 0.5547
```

Conclusion

In this project we have been introduced to linear regression and the radial basis functions. Also the mathematical analysis has been done for both the closed form solution and the Stochastic Gradient Descent solution. The various hyperparameters importance is first studied theoretically and then based on the LETOR dataset problem the dataset is first understood and then both the models are applied. Based on the models the output is understood. The closed form solution provides a decent accuracy of about 75 percent and so does the Stochastic Gradient Decent model. As per the analysis of the model, I believe that the data being skewed is one of the reasons training is difficult. To train such a model the number of iterations should be increased to a large number and the learning rate to a very low value so that the weights get updated to an accurate value. The model is predicting zeros for almost all inputs, one of the outputs is as shown-

References

[1] Microsoft *LETOR: Learning to Rank for Information Retrieval.* `https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/` 2017

[2] Pattern Recognition And Machine Learning, *Book by Christopher Bishop*