CSE – 574 – Introduction to Machine Learning

Project Report – Linear Regression with Basis Functions on Microsoft’s LeToR Data Set

Ankit Sablok

UB Person # - 5009-6348

Email-id – [ankitsab@buffalo.edu](mailto:ankitsab@buffalo.edu)

Problem Description

The project aims to implement and evaluate several supervised machine learning approaches to the task of linear regression. The objective is to learn how to map an input vector x into a target value t using the model. The mathematical generalization of the problem is to find a function ‘y’ which takes in the input vector ‘x’ and weight vector ‘w’ and uses the linear combination of the components of the vector x to spit out a result which is the expected value that we would receive when we see ‘x’. The function appears to be something like the following -:

*y(x,w) = t, t – target value*

w = (w0, w1, w2, ………………, wn) is the weight vector

x = (x0, x1, x2, …………………., xn) is the data input vector where each xi is a feature of the data point in n dimensional space.

In this problem we have been given Microsoft’s LeToR data set which is basically a file consisting of (query, url) pairs, wherein each query has a specific set of features associated with it and all those features are weighted, along with each (query, url) pair we have been given a quantity called as the relevance label of the pair which indicates how strongly does the url satisfy that particular query which in brief means how good a match that url is for that query. The following are the certain features of the LeToR dataset given to us -:

* Each query is described using a set of 46 weights, which represent different features of a query which acts as one of our data point.
* The first column in the data set given is the relevance label of the query/url pair, it’s a number that belongs to the set {0, 1, 2} more the number more the relevance.

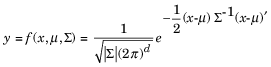
Toward a Solution

The solution to the problem described above comprises of 3 following stages -:

Data Parsing – the data given to us is just raw text, firstly we convert the data set given to us into a matrix having dimensions 69623x47, 69623 for the number of query/url pairs and 47 for the 46 features of each query in addition to the relevance parameter. Out of this data matrix we use 80% of the data matrix as our training set, the next 10% data as our validation set and the remaining as our test matrix. These are named as trainingSet, validationSet and testSet. All the data parsing is done in the file project1.m. Executing this file creates several mat files that will be used in further computations.

Model Training – using the training set obtained by executing “project1.m”, we train our model for the closed form and stochastic gradient descent model. Let’s see that in a bit more detail.

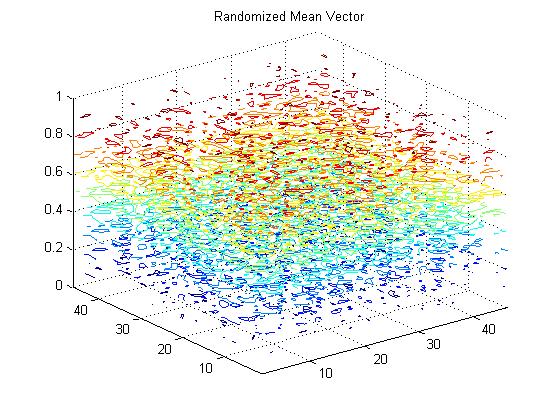
1. Closed Form Model – the closed form solution is implemented in the file “train\_cfs.m”. The closed form solution uses the Multivariate Gaussian Basis Function to evaluate the N x M design matrix.

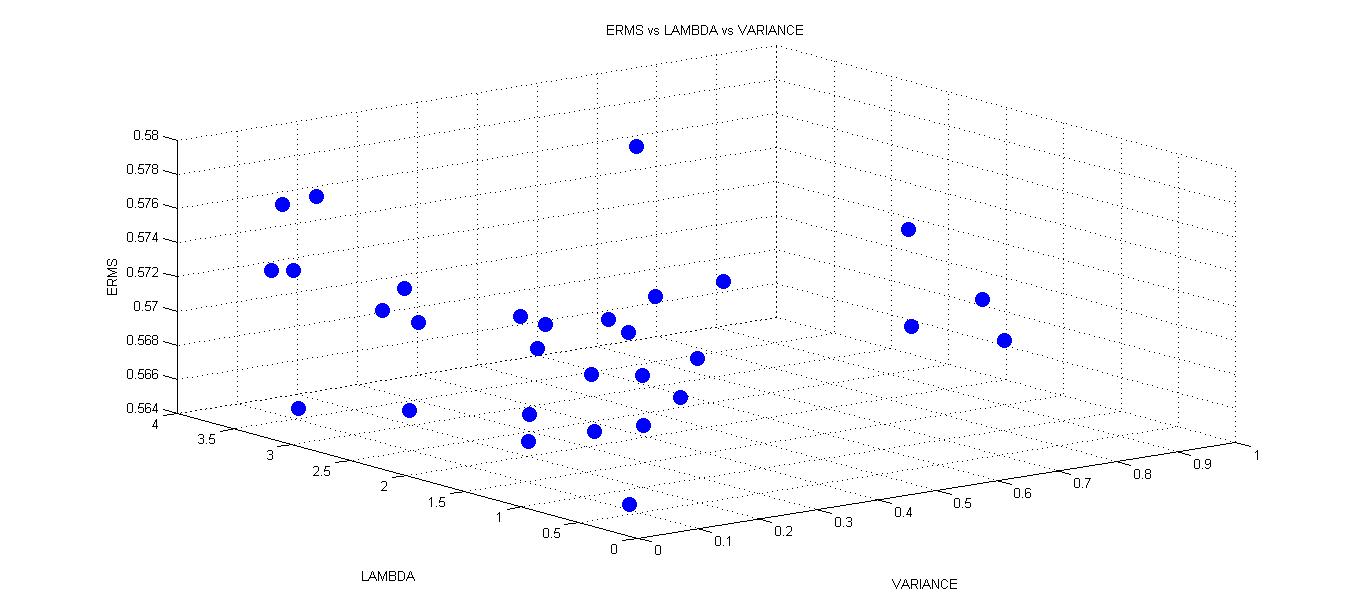


Using this design matrix, we evaluate the weight vector using the regularization factor lambda and the φ(x) as follows -:

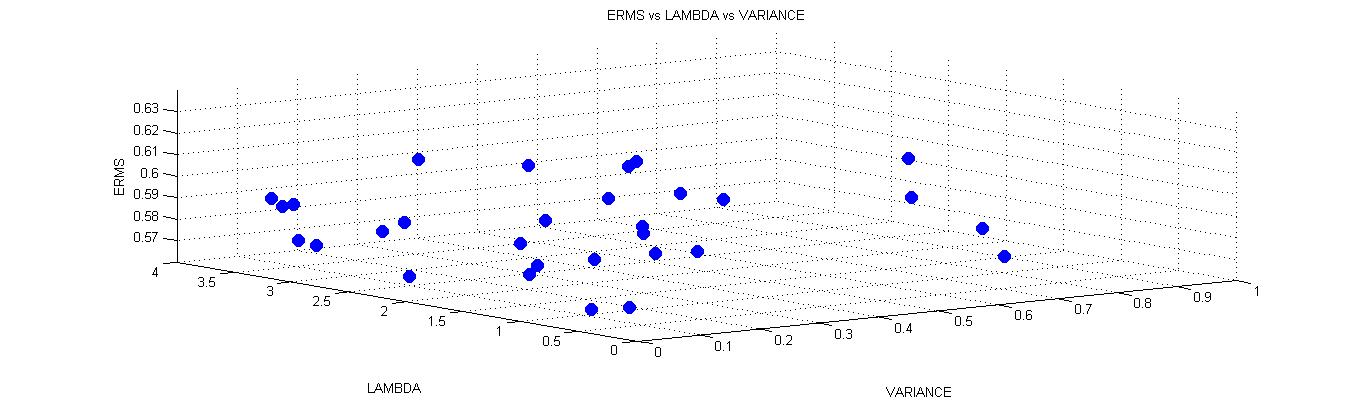
weightMatrix = (inv(lambda\*identityMatrix + phiX' \* phiX)) \* (phiX') \* trainingSetRelevance;

After doing the above we use the weight vectors and compute the root mean square errors for the training, validation and test set and we obtain the following results along with the graphs. The mean vector used in performing all the computations looks like the following -:

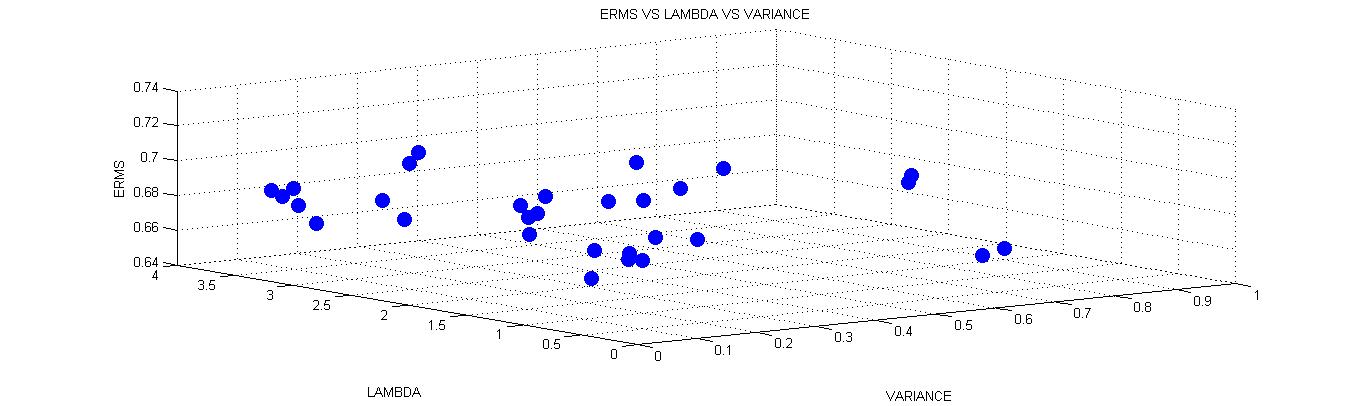




After plotting the above scatter plot of ERMS vs LAMBDA vs VARIANCE for the training set, we find that the minimum value of ERMS turns out to be 0.5648 for lambda = 0.4144 and variance = 0.0661.



After seeing the above plot of ERMS vs LAMBDA vs VARIANCE for the validation set, we find the minimum value of ERMS turns out to be 0.5681.

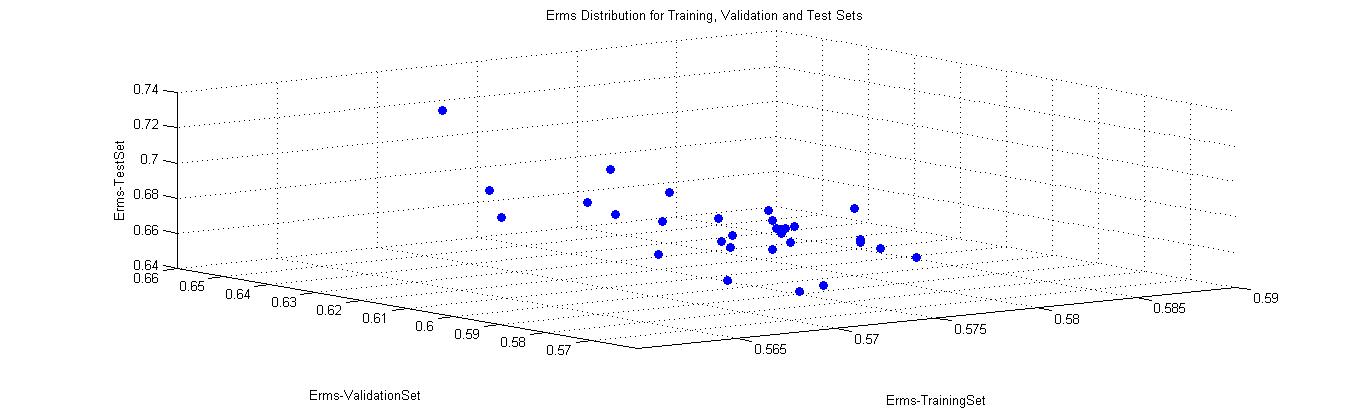


After seeing the above plot of ERMS vs LAMBDA vs VARIANCE for the testing set, we find the minimum value of ERMS turns out to be 0.6560. This can be also seen in the mesh represented by the figure.

In conclusion we have the following results -:

1. The ERMS value for the training set obtained is **0.5648**.
2. The ERMS value for the validation set obtained is **0.5681**.
3. The ERMS value for the testing set obtained is **0.6560**.

The cumulative distribution of ERM’s values can be seen through the following scatter plot -:



In performing all the above values, 46 Gaussian basis functions were used and different parameters like standard deviation, mean and lambda were varied to get the best weight vectors and best root mean square error for the data set which in turn gave the optimal values for mean vector, standard deviation and lambda.

1. Stochastic Gradient Descent – This is yet another method for solving the problem of predicting the values of new data elements based on past data. The gist of the stochastic gradient descent algorithm can be explained as follows, we start off with an initial weight vector which we continue to optimize using some random samples from the training set until we converge to a specific value of the error rate at which point we stop the process of optimization and terminate execution. The training procedure for the stochastic gradient descent algorithm used the following steps -:

* Use a starting random weight vector.
* Evaluate the square error using this randomized weight vector.
* Iterate for a subset of samples for this project 100 samples were used and in each iteration we find the value of the new weight vector obtained from the previous weight vector using the learning parameter “ita” which we start off with 1.
* Then we find the value of the new square error using the new weight vector and update the value of “ita” according to the rule if new error is bigger than previous error then ita = ita/2.
* At the end of the loop we find that the root mean square error converges to a specific value, and this means we finally have landed upon the weight vector.

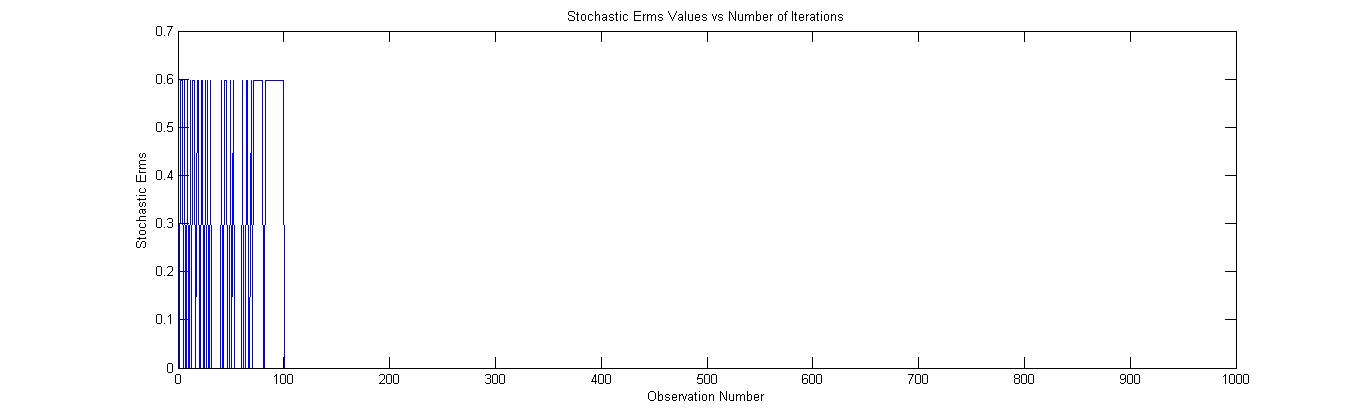
The formula used for computation is the following -:

newEdW = newEdW + (train\_gdTrainingSetRelevance(j,1) - train\_gdPhiX(j,:)\*newW)^2;

The formula used for validating the data set and testing the data set using the Stochastic Gradient Descent algorithm is -:

test\_gdValidationPhiX(i,j) = exp((-0.5)\*(test\_gdValidationSetWithoutRelevance(i,:) - test\_gdValidationMeanVector(j,:))\*(test\_gdValidationSetWithoutRelevance(i,:) - test\_gdValidationMeanVector(j,:))');

For performing the aforementioned procedure a 46 x 46 randomized mean vector was used, which is also used for evaluating the Gaussian design matrix. In the end all the root mean square errors are saved. Let’s take a look at how the computation of stochastic root mean square error varies with the number of iterations as we compute the gradient. The minimum error found while executing the weight vectors found using the test set is 0.6903.



After performing all the experiments we come to the following conclusion -:

1. The minimum root mean square error for the training set is 0.5892.
2. The minimum root mean square error or the test set is 0.6903.

**Which one’s a better model?**

After performing all the experiments I found the Linear Regression with Gaussian basis functions as an efficient approach towards performing predictions on new data set. Linear regression is also computationally efficient as compared to stochastic gradient descent algorithm. In the stochastic gradient descent algorithm there is a good probability that it mislead us to some value which is minimum as per the algorithm but might not be a global minimum and hence is prone to error. Hence I would prefer the linear regression with radial basis function model over the stochastic gradient descent algorithm.