CSE 574 – PROJECT 1 REPORT

Linear Regression with Basis Functions

Name : Animesh Kumar

UBID : 50134753

Abstract:

Train a regression model based on query-url pair datasets , then predict the page relevancy labels for new coming queries.

Introduction :-

LETOR is a package of benchmark data sets for research on Learning To

Rank released by Microsoft Research Asia.

The latest version, 4.0, can be found at

<http://research.microsoft.com/en-us/um/beijing/projects/letor/letor4dataset.aspx>

For this project, one dataset of MQ2007 is used (supervised ranking):

”Querylevelnorm.txt” (69623 urls/samples in total)

Given text file was used to generate the Input and Target matrices for the project implementation.

During our implementation, we divide the dataset into three parts as follows :-

1) Training Set – It comprises of 80% of the total dataset. This data is used to create the learning algorithm.

2) Validation Set – It comprises of 10% of the total dataset. The learning algorithm created by the training algorithm is validated in validation step using the validation dataset and then various parameters are fine tuned to obtain better results.

3) Testing Set – It comprises of the last 10% of the dataset. Once the training algorithm is ready and validated, we test the algorithm on the testing data and calculate the error between predicted values generated by our algorithm with the expected target values provided in the testing dataset (read relevancy labels provided).

Approach :

We worked on two approaches for generating the machine learning algorithm:-

1) Closed-form Maximum Likelihood Solution

2) Stochastic Gradient Descent Maximum Likelihood Solution

In both the solutions, for the choice of basis functions, Gaussian distribution was used as Gaussian distribution works better in terms of accuracy for higher dimensions (46 in our case) due to it's quadratic non-linear nature of x. { (x-μ) 2}.

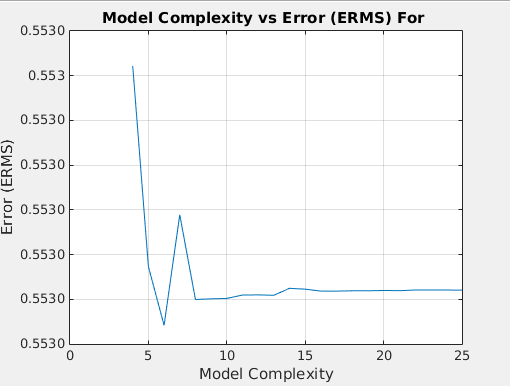
Closed-form Maximum Likelihood Solution

For the Gaussian Distribution, there are two parameters Mean (μ) and Variance (σ 2) which needs to be chosen and fixed.

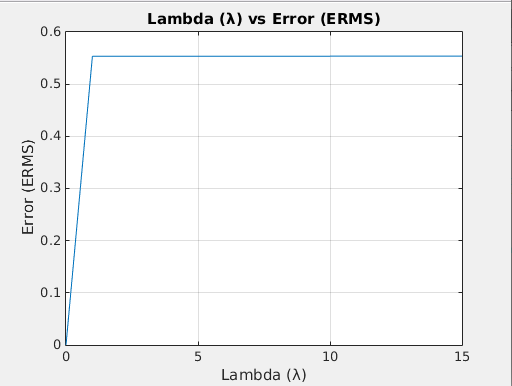
**Mean (μ)** – Mean is the deciding parameter for the location of the basis function that will be generated. For a given Model Complexity (M), M-1 datapoints (vectors) were chosen randomly from the training dataset and will be applied to M-1 basis functions (excluding the first basis function in M basis functions). Each of these M-1 vectors will have the dimensions of 1 x D (D=46).

**Variance (σ 2)** - Variance is the spatial scale of the gaussian distribution for the basis functions. Variance was calculated by the Matlab function **var(x(:))**. This gives an average variance of every column of the the dataset.

**Model Complexity (M)** – For model complexity we start with first start with a value M=4 and calculate the weights using the basis function generated and use the weights in the validation set to generate the error for the given complexity. This error is used to calculate the Root Mean Square error. Now the same step is repeated for Model Complexity upto 25. The resultant Root Mean Sqaure Errors are plotted on a graph against the Model Complexities and the Model Complexity with the least Root Mean Square Error is chosen. In our case, it came to be at M = 6. So we fix M=6 and try to tune other parameters.



**Regularization Parameter : Lambda (λ)** – This parameter is the regularization coefficent used to regularize the error and hence avoid over fitting. To calculate lambda, once the model complexity is fixed, we iterate over the Lamda values from 1 to 15 and calculate the weights. The weights are then used to generate the error. The resultant Root Mean Sqaure Errors are plotted on a graph against the Lamda values and the Lamda value with the least Root Mean Square Error is chosen. In our case, it came to be at λ = 1.



**Evaluation (Root Mean Square Error from Testing) :**

The final weights, Lambda value, Model Complexity, Mean and Variance values are then passed to the testing set. A Gaussian Distribution is generated on the test set using the provided Mean and Variance. Once the Gaussian distribution is generated, using the available value of Lambda and weights, the error values are calculated. This error value is the final ERMS value. In our case, it comes to be 0.64 on the test set.

Stochastic Gradient Descent Maximum Likelihood Solution

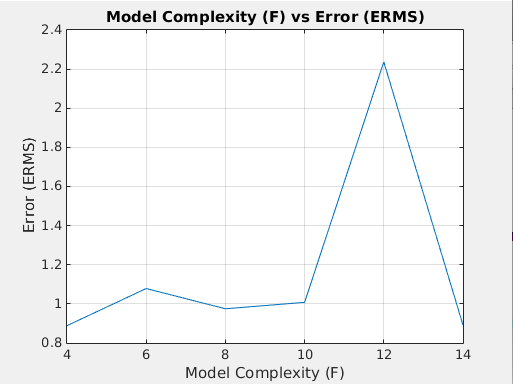
For Stochastic Gradient Descent, most of the steps other than calculating the weights are same.

For the Gaussian Distribution, there are two parameters Mean (μ) and Variance (σ 2) which needs to be chosen and fixed.

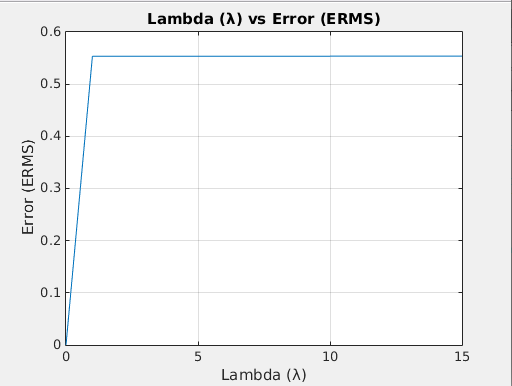
**Mean (μ)** – Mean is the deciding parameter for the location of the basis function that will be generated. For a given Model Complexity (M), M-1 datapoints (vectors) were chosen randomly from the training dataset and will be applied to M-1 basis functions (excluding the first basis function in M basis functions). Each of these M-1 vectors will have the dimensions of 1 x D (D=46).

**Variance (σ 2)** - Variance is the spatial scale of the gaussian distribution for the basis functions. Variance was calculated by the Matlab function **var(x(:))**. This gives an average variance of every column of the the dataset.

**Model Complexity (M)** – For model complexity we start with first start with a value M=4 and calculate the weights using the basis function generated and use the weights in the validation set to generate the error for the given complexity. In this method, for calculation of weights, we first choose an initial weight randomly and then at each iteration, this weight it improved using a new tuning parameter called Learning weight. We need to generate a minimum error possible by tuning these values. At each step error is calculated against the values of target vector, and if error is decreases, the error is accepted and we proceed to next iteration otherwise, the learning rate parameter is modified and the weights are calculated again till the error is decreasing. If the error starts decreasing continuously and the the error difference reduces to very low values, we stop the process. This error value is chosen for the given parameter. This error is used to calculate the Root Mean Square error. Now the same step is repeated for Model Complexity upto 25. The resultant Root Mean Sqaure Errors are plotted on a graph against the Model Complexities and the Model Complexity with the least Root Mean Square Error is chosen. In our case, it came to be at M = 4. So we fix M=4 and try to tune other parameters.



**Regularization Parameter : Lambda (λ)** – This parameter is the regularization coefficent used to regularize the error and hence avoid over fitting. To calculate lambda, once the model complexity is fixed, we iterate over the Lamda values from 1 to 15 and calculate the weights. The weights are then used to generate the error. The resultant Root Mean Sqaure Errors are plotted on a graph against the Lamda values and the Lamda value with the least Root Mean Square Error is chosen. In our case, it came to be at λ = 1.



**Evaluation (Root Mean Square Error from Testing) :**

The final weights, Lambda value, Model Complexity, Mean and Variance values are then passed to the testing set. A Gaussian Distribution is generated on the test set using the provided Mean and Variance. Once the Gaussian distribution is generated, using the available value of Lambda and weights, the error values are calculated. This error value is the final ERMS value. In our case, it comes to be 0.88 on the test set.

**Final Thoughts :-**

1) Performance : Both solutions give almost similar error rate but the Closed Form solution produced a lower error than the Stochastic Gradient Solution ((0.64 for Closed Form Solution and 0.88 for Stochastic Gradient Descent).

2) Also due to large number of iterations to bring the error to a considerable and acceptable state, the running time of the algorithm increases a lot.

2) The Model complexity for Closed Form Solution came to 6 whereas for Stochastic Gradient descent solution it was 4. Hence, the model complexity for Stochastic Gradient Descent Solution was better.