PROJECT 1 – LINEAR REGRESSION

Name: Rakesh Baingolkar

UB ID: 50097576

OBJECTIVE:

Project was to implement and evaluate supervised machine learning approaches to the task of linear regression. The main objective was to predict the target value t corresponding value of input vector x using a model

$$y(x.w) = w^T \phi(x)$$

where $w = (w_0, w_1, \ldots\ldots w_{M-1})$ is a weight vector to be learnt from the training samples and $\phi = (\phi_0, \phi_1, \phi_2 \ldots\ldots \phi_{M-1})^T$ is a vector of M basis functions. Each basis function $\phi_j(x)$, j=0….M-1 converts the input vector x into a scalar value.

Gaussian Basis Function:

$$\Phi_j(x) = \exp(-(x- \mu_j)^2 / 2s^2)$$

Where μ_j is a vector in feature space and s is an isotropic spatial scale.

The training data set consists of N exemplar vectors $X = (x_1, x_2, \ldots\ldots x_N)$ together with the corresponding values $t = (t_1, t_2, \ldots\ldots t_N)$

DATASET:

Data set used is the Microsoft LETOR Dataset.

Data contains N = 69623 query-document pairs and d = 46 dimensions.

The first column is the relevance label and the relevancy of the document depends on this value.

Second column is the query ID.

The following columns till 46 are the features normalized in 0, 1 range.

Last column is the details of the document.

MODEL DESCRIPTION:

- First I parsed the data and converted into a .mat file names parsed_r_file.mat which contains the entire data named as parsed_data obtained by parsing the given dataset. The .mat file contains two parts namely initial_data which is a matrix of all the data and the to_check_with which is the target vector.
- I take eighty percent of the total data which is used for training and then using the datasample function I take 20,000 random samples from the data.
- Random sampling is done Modular Complexity times and we get a mean vector at every sampling eventually getting a matrix containing mean vectors.
- Using this collection of mean vectors I calculate the Gaussian Basis Function

$$\Phi_j(x) = \exp(-(x- \mu_j)^2/ 2s^2)$$

  Where μ_j is a vector in feature space and s is an isotropic spatial scale.
- Then we find the quadratic regularizer extension

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

- This basis function is in turn used to minimize the sum of squares error.

$$E_D(w) = 1/2 \sum \{ t_n – w^T \Phi(x_n) \}^2$$

- Then we use the regularization parameter to prevent overfitting.

$$E(w)=E_D(w)+ \lambda E_w(w)$$

  Where $E_w(w)= ½ \sum |w_j|^q$

- Then we find the root mean square error.

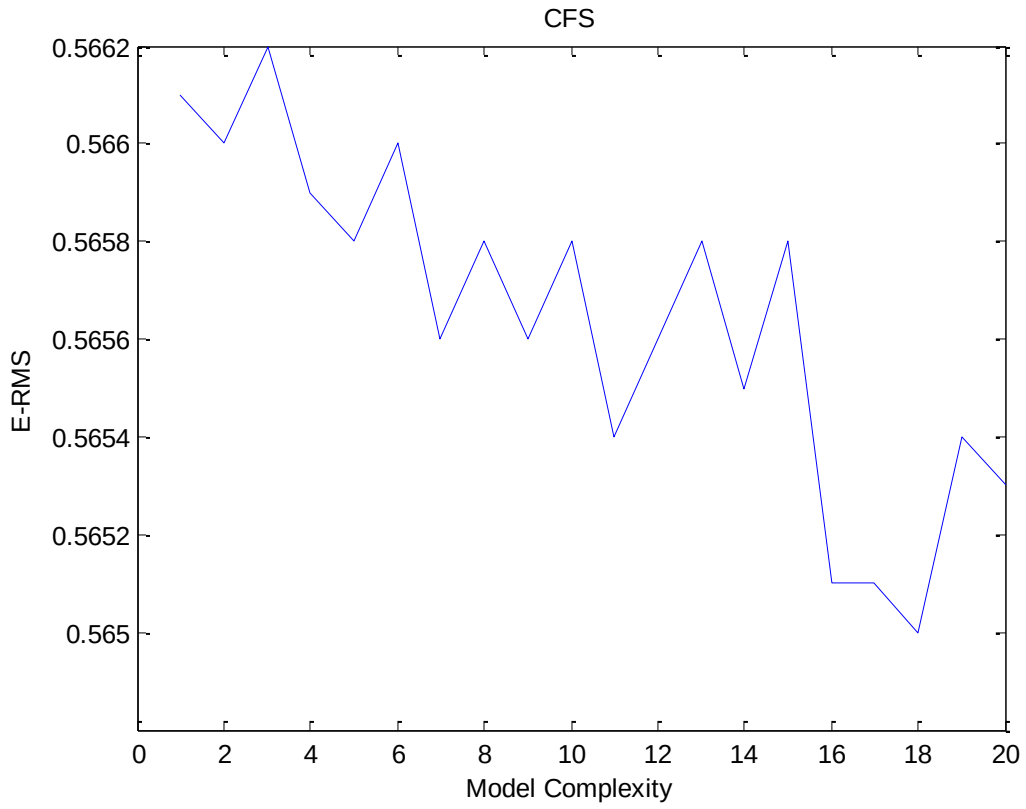
$$E_{RMS} = \sqrt{2} * E(w)/N$$

N = size of training set.

OBSERVATION:

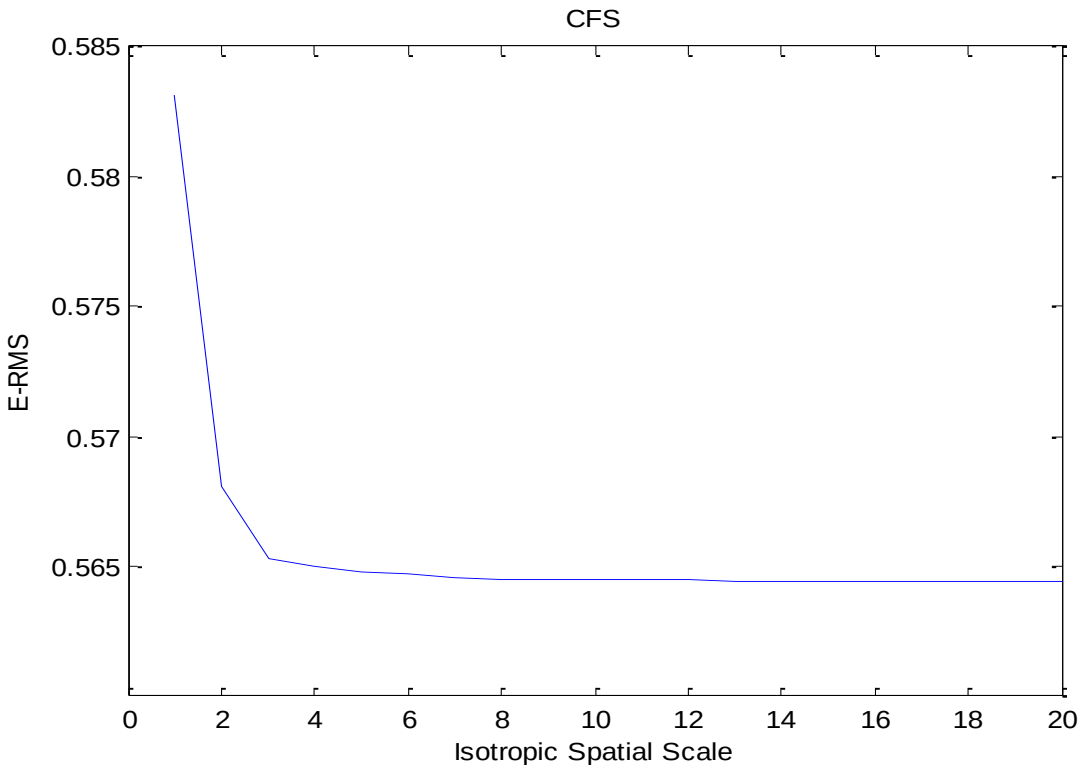
I took three sets of observations.

- First I maintained the spatial scale S and regularization parameter λ constant and changed the Model Complexity from the range 1 to 20 and observed the corresponding values of E_{RMS}. The graph of the observation is shown below.



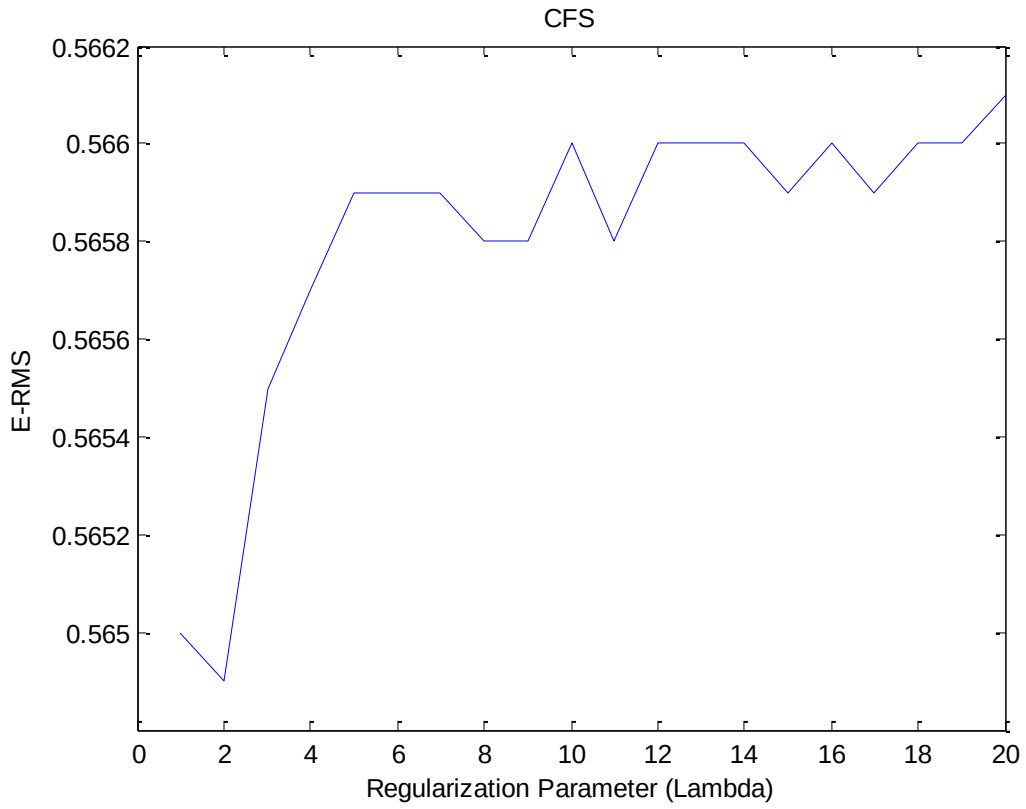
From the graph we can see that for the range 1 to 20 of model complexity the corresponding E_{RMS} values are in the range 0.565 and 0.5662 with model complexity at 18 gives the least E_{RMS}.

- Next I maintained regularization parameter λ and model complexity as constants and changed the spatial scale from 1 to 20. The graph for this is shown below.



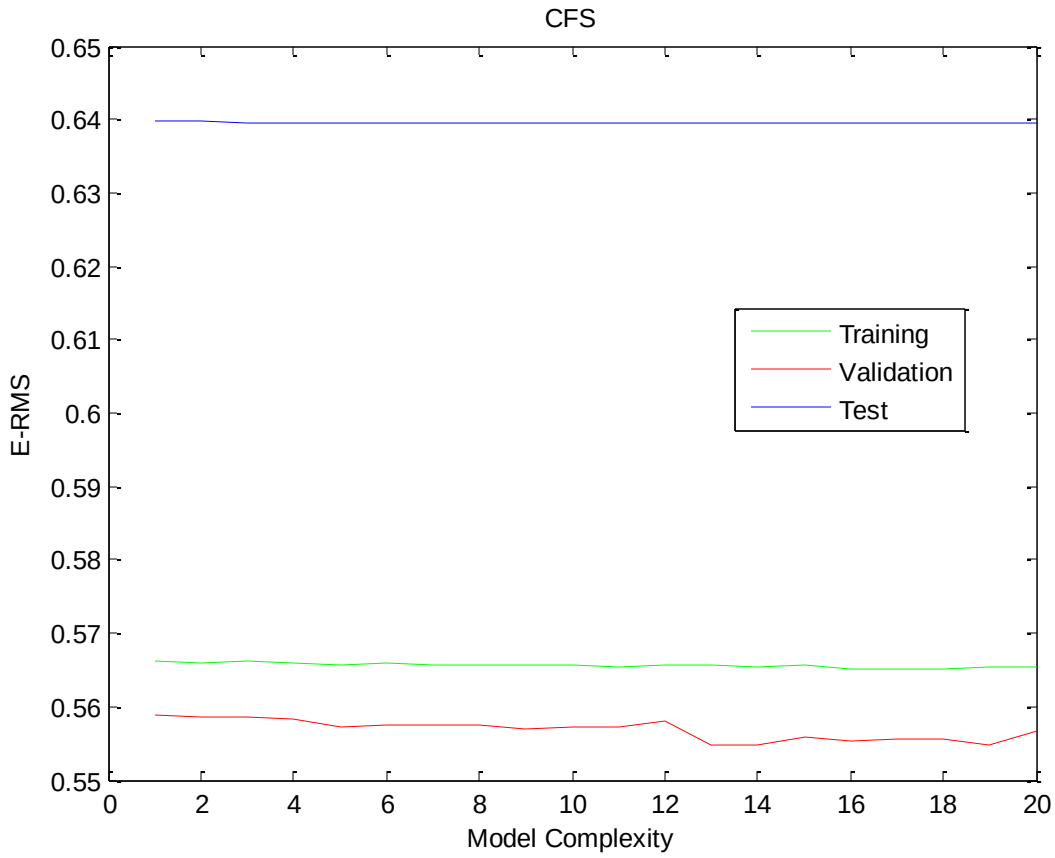
From the graph we can see that E_{RMS} is high in the initial phase i.e. when the spatial scale is 1-2 and then it gradually decreases and then it is constant at about 0.5654 which is the least error in the entire readings. Minimum E_{RMS} is when s= 3 to 20

- For the third part I kept Model Complexity and the spatial scale constant and changed the regularization parameter from 1 to 20.
  The graph is as follows.



From the readings we can see that as we increase the lambda then the E_{RMS} increases. Minimum error is when the lambda is 2.

- From the readings the minimum value that I get from the readings is when s = 3, Model Complexity = 20 and lambda = 2 and the E_{RMS} = 0.5649.
- The next step is finding the E_{RMS} for validation and test sets. The values set for these readings are: lambda = 2 and s =2 and the readings obtained are shown in the graph below.



RESULT:

Final E_{RMS} validation = 0.5548 E_{RMS} test = 0.6394.

STOCHASTIC GRADIENT DESCENT:

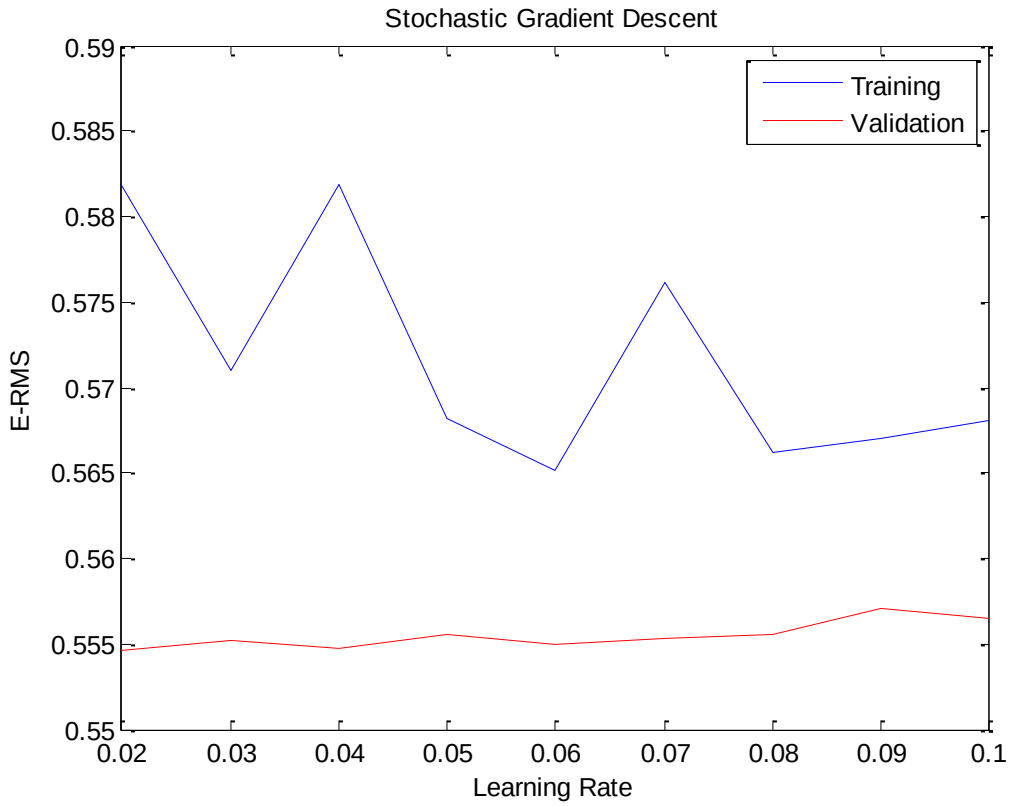
- Gradient descent is a method where we are at a particular position in a graph and our aim is to reach the local minima as rapidly as we can.
- One method to reach rapidly is to take one step at a time in the direction of local minima. The step we take is based on the learning rate which helps in the updating of the value.
- If the learning rate is large then our value will diverge instead of converging.
- Initial value of learning rate is a random value in range 0-1 and then according to the E_{RMS} we change the rate to be half the previous rate.
- Like the previous method I took 20000 random samples from the initial data set and calculated the Gaussian basis function.

- Then we calculate the quadratic regularizer extension

$$W_{T+1} = \quad W_T + \eta \, (t_n - W_T^T \, \Phi_n) \, \Phi$$

- And then we find the E_{RMS} as we calculated previously. If the Error is less than the previous iterations then it is the new error or else we update the learning rate as half of the previous one.

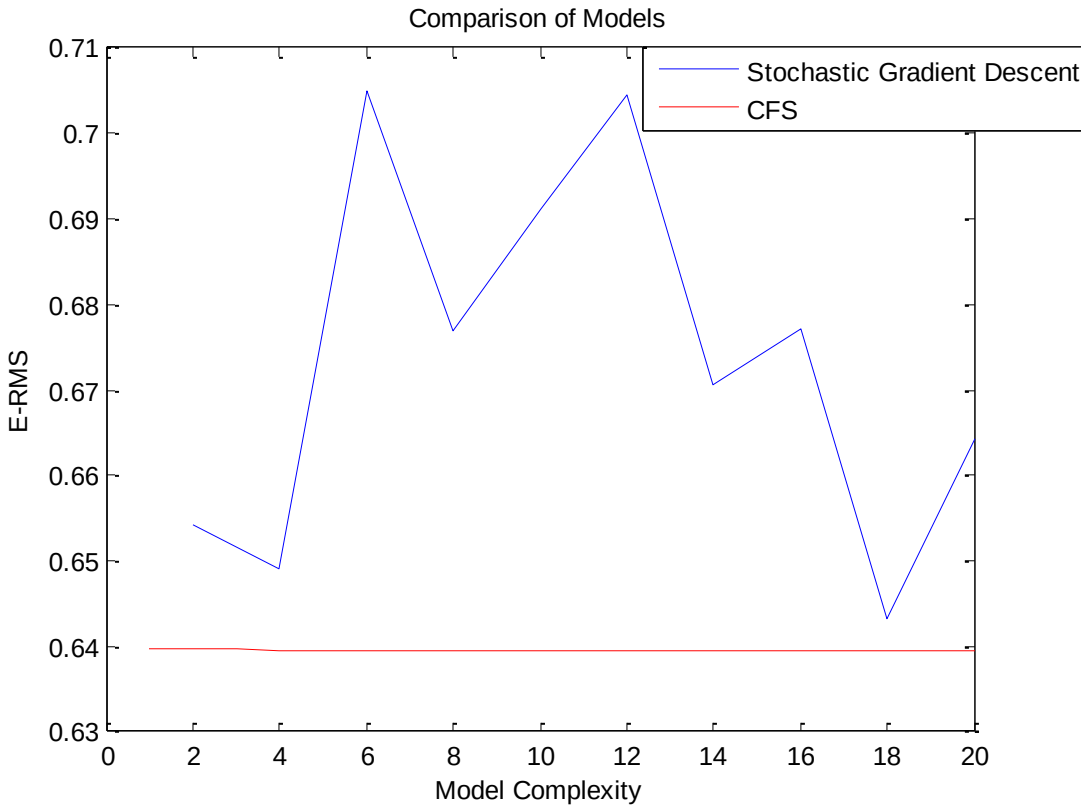
The graph below shows the change in the E_{RMS} with respect to the learning rate η.



The learning rate of 0.06 gives the least E_{RMS}.

- We fix the learning rate to 0.06 and vary the Model Complexity in Test Data.

Now comparing the two models we get the following graph.



From the above graph we can say that CFS model has less Error and thus from the results we can say that CFS Model is better.