Linear Regression with Basis Functions

Name: **Mimanshu Shisodia**

UBIT Name: **mimanshu**

UBIT Number: **50133318**

**Problem Overview**

The aim of this project is to perform **supervised learning** task and implement Linear Regression with **Basis Function** over the given dataset which is Microsoft provided **LeToR** (Learning to Rank) data. This dataset has **46** features some of which may be tf-idf values, uplinks, downlinks on the page, colors, fonts etc. The whole dataset is normalized hence the data was already cleaned.

The **features x** are considered as input vectors with corresponding **target** values **t**. We need to define a basis function Phi(x) using which we will train the model to find the corresponding weights (parameters) by mapping the input features into this model. The model equation is a linear equation in terms of weights w and linear combination of Basis functions.

The Generalized Equation is:

http://latex.codecogs.com/gif.latex?y%28x%2Cw%29%20%3D%20w%5ET%20%5Cphi%28x%29

ϕ(x) 🡺 also called **Design Matrix**

The **aim** is to learn weights **w** which can be used in the above mentioned predictive model to predict the future unknown value.

**Methods to be implemented:**

1. Maximum likelihood with closed form solution
2. Maximum likelihood with Gradient Descent

**Dataset Properties:**

1. Number of input parameters: **46**
2. Number of data samples: **69623**
3. Normalized between 0 and 1.
4. Output is labeled in 0, 1 and 2, which is a relevancy score, 2 being the most relevant and vice versa.

**Selecting the Basis Functions:**

It is important to select the best basis function to train the model. There are different basis function such as **Polynomial, Gaussian, and Sigmoidal etc**. Here, in this problem we will be using Gaussian Basis function. The advantage of using this are:

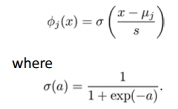
* It can represent the input features in a non-linear relationship with the target values.
* It is computationally effective since its value is non zero only for a small interval and zero otherwise. Results in a very sparse design matrix.

**Polynomial vs Gaussian vs Sigmoidal: Why Gaussian?**

**Polynomial:** 

**Gaussian:** http://latex.codecogs.com/gif.latex?phi%28x%29%20%3D%20exp%28%28x-%5Cmu%29%5ET%20%28x-%5Cmu%29%29/2*%28s*s%29

**Sigmoidal:**



We will work on **Gaussian** because

* It can represent the input features in a non-linear relationship with the target values.
* It is computationally effective since its value is non zero only for a small interval and zero otherwise. Results in a very sparse design matrix.
* Polynomial distribution is prone to high variance across all the basis function if there is a slight change in the values of Features.

**Terminology and Tuning Parameters**

In order to find the best fit, we need to tune some parameters of out equations. These can be classified as Parameter and Hyper parameters.

* M (Model Complexity) : This is the number of basis function used in the model function y(x,w)
* (Mean): Randomly selected mean.



* http://latex.codecogs.com/gif.latex?%5Clambda (Lambda): This is called the regularization coefficient and is responsible to reduce over fitting and under fitting. Also balances the bias and variances.
* s (Standard Deviation) : Randomly selected variance of the basis function.
* http://latex.codecogs.com/gif.latex?%5Ceta (Learning Rate): Use is gradient descent method. Stochastic analysis.

**Start Up Procedure:**

1. Parse the data and stored in a .mat file. I used **textscan** for parsing the data and segregated afterwards in the **Input Matrix** and **Output Vector**.

**Method 1: Maximum likelihood with closed form solution**

**Learning Process:**

* **Step-1:** Tune different M and divide the **design matrix** into

1. Training set.(80% of the data)
2. Test set. (10% of the data)
3. Validation set. (10% of the data)

An iteration was run different value of M ranging from M = 5 to 50. Hence 5 to 50 **random means** were selected from the collection. The dimension of the **Mean Vector** is **1 X46.** For the different Model Complexity, Phi(x) was calculated and were stored on the disk in a .mat file. Since we are tuning M at this point of time, we have selected a random value of standard deviation **s.**

Therefore, **s = 0.5 at this time.**

(Since the Data is normalized between 0 and 1, we will take the mean of min and max value of collection to define the variance at this point and tune the M)

**Result of Step 1:**

* We will get the **Phi(x)** Design Matrix after every loop iteration over M.
* Hence the **dimension** of this **Design Matrix** will be **69623 X M.**
* We will append a column of **ones vector** as the first column to get the zeroth **Phi vector** for zeroth parameter **wo**
* The final dimension of our Phi(x) is **M+1**.
* Design Matrix is divided into 3 data sets, **training, validation and testing** together with the output vector.
* **Step-2:** Iterating each Phi(x) design matrix with the different values of lambda (detailed explanation in Step 5) parameter to calculate the weights parameters, using the formula:

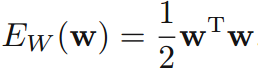


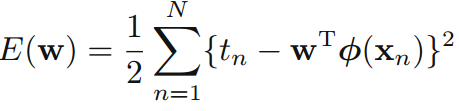
This is the closed form equation of the maximum likelihood solution of the Linear Regression with the basis function which is obtained by differentiating sum of squares function w.r.t. each parameters **w.**

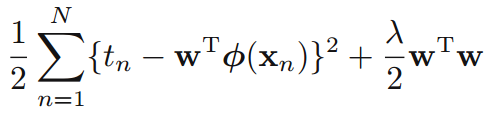
**Result of Step 2:**

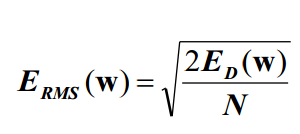
* We get the weights **w** for each design matrix of order M+1.
* **Step-3:** Using the weight we got in the step 3rd, we will be calculating the root mean square error. The formula used is:



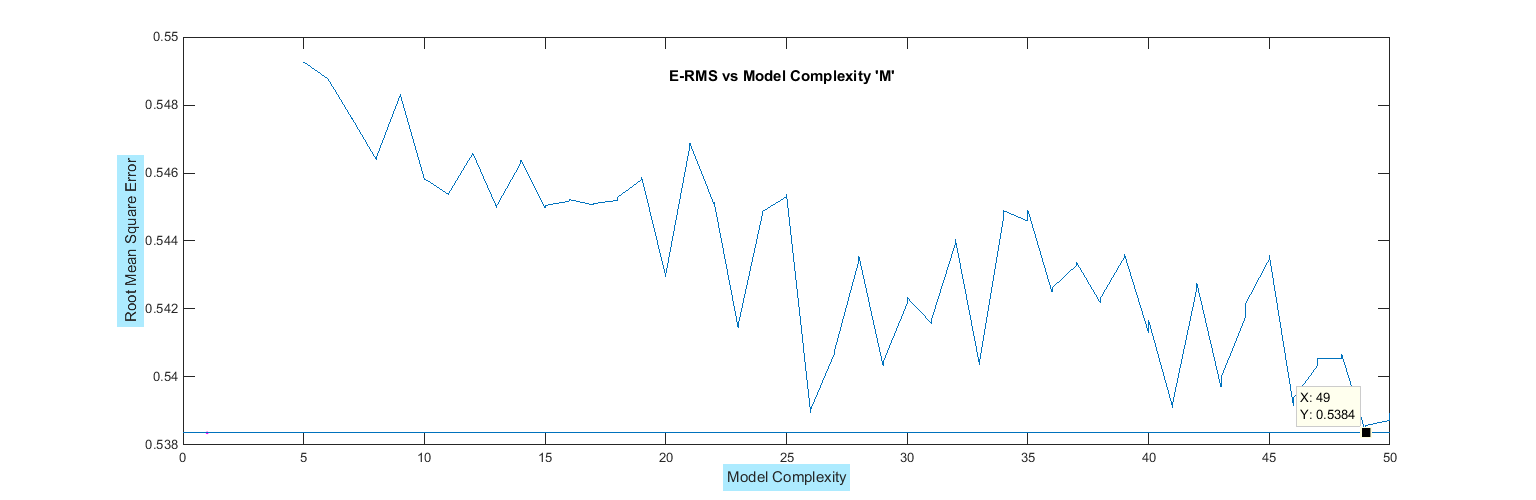




**Total Error:**  

**RMS Error:** 

**Error Calculation Procedure**: We will use the validation hence forward to calculate error. Also final error will be Root mean square error. All the calculations have considered the quadratic regularized error instead of lasso. Since the difference in the observed values of both was not that major.

A graph is plotted showing Error present for different Models. All the subsequent **Error Plots** are plotted on **Validation Data Set**

**Figure 1**

**Result of Step 3:**

As per as the figure we are getting minimum error for Model Complexity **M = 49** with minimum error of **0.5384**

* **Step-4:** Earlier we were tuning the M with a fixed value of variance parameter **s = 0.5.**

Now at this time, we will tune the variance parameter **s** for a different values over a range of **0.1 to 1.**

For this we will select Model Complexity on the basis of data obtained in **Figure-1.**

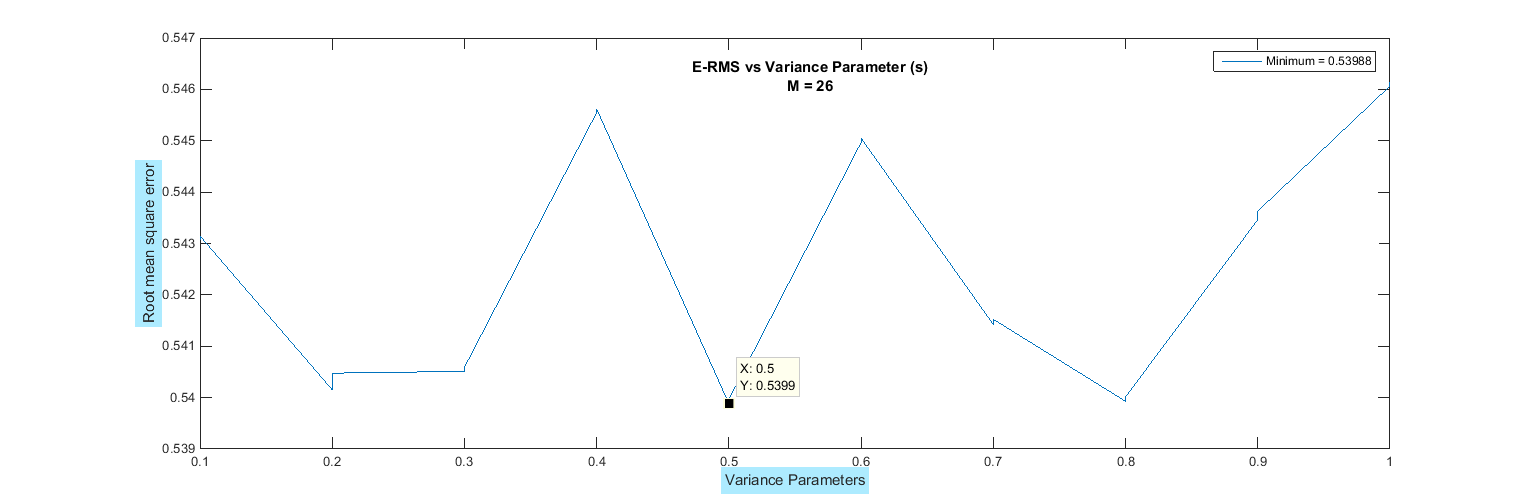
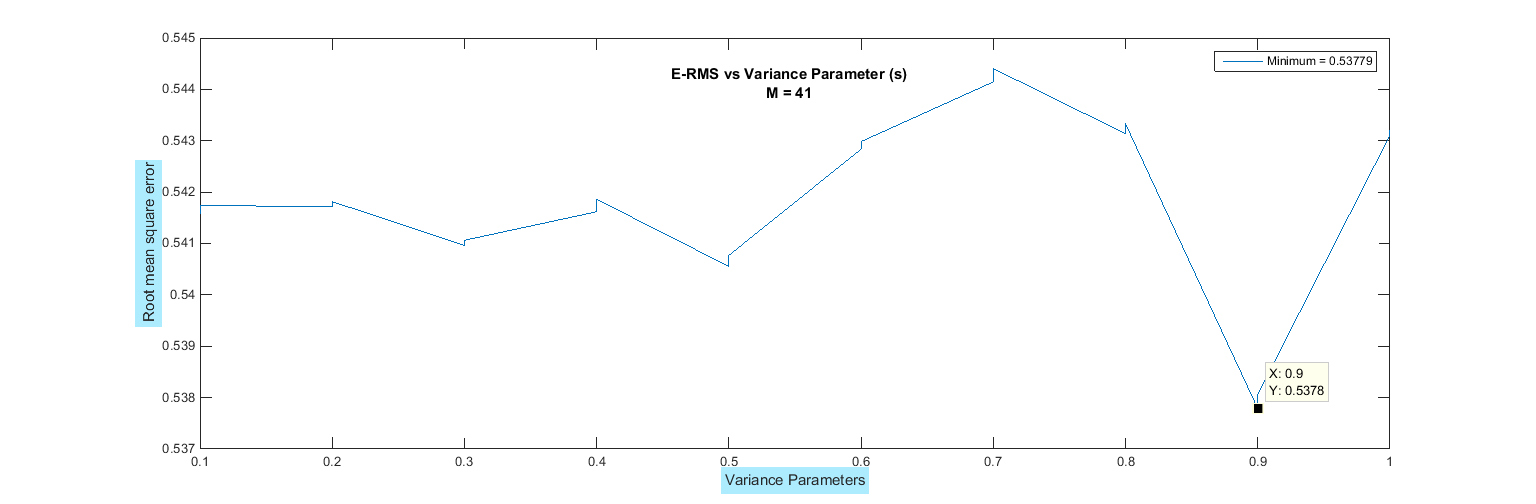
Selected M:

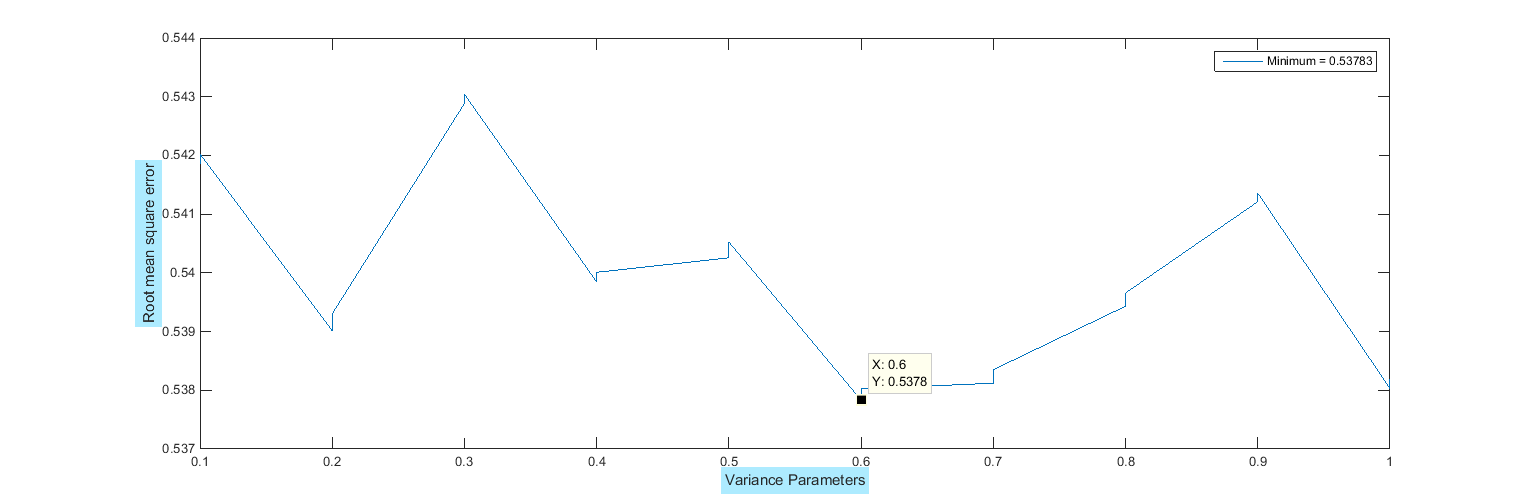
**M = 26**

**M = 41**

**M = 49**

Plotting the Error obtained with these Model Complexities w.r.t. to the different values of s in the range of **0.1 to 1.**

* **For M = 26**
* **For M = 41**

**M = 49**

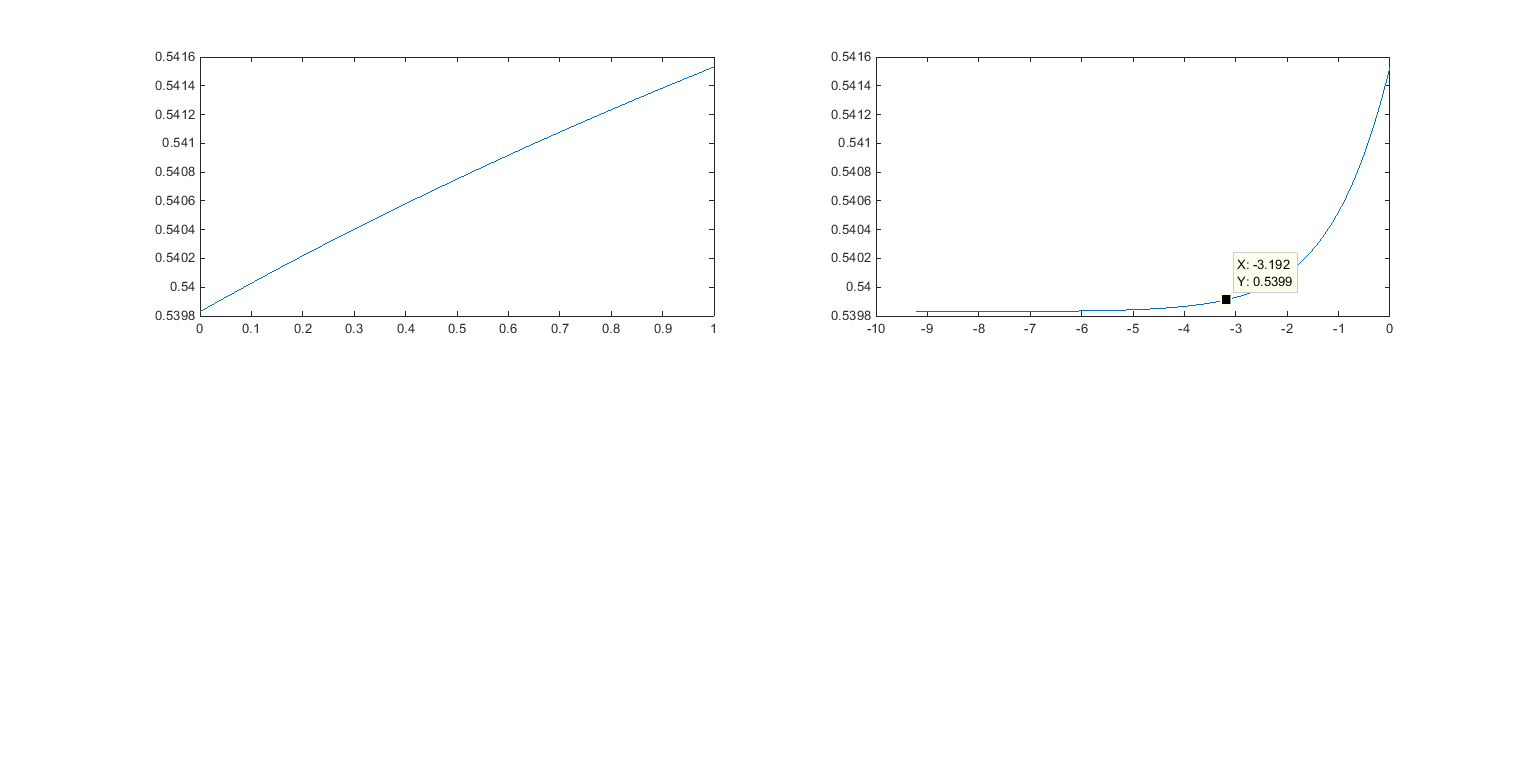
**Result of Step 4:**

* As can be observed that the minimum **error** coming is **0.5378** for both the Model Complexity **M = 41 and M = 49.**
* In order to choose the optimal Complexity and optimal value of variance parameter, we can say that having large variance can result is overlapping between the different Gaussian Basis Function. Hence for **M = 49** we have **s = 0.6** with error **0.5378.**
* There the observed optimal value of tuned Model Complexity and variance are:
  + **M = 49**
  + **s = 0.6**
* **Step-5:** Tuning the **Lambda parameter** known as **Regularization coefficient**.

Previous iteration were calculating the design matrix over the different values of lambda with a range of .0001 to 0.1.

Also I tried to run the values of lambda over a range of **0.1 to 100**. But an increasing curve was obtained. Therefore the lambda coefficient was then reduced to a range of **.0001 to 0.1.**

Since lambda is used to control the over fitting and under fitting issues in the Model, we will run a loop for **M = 49** over above mentioned range.

The plot obtained for log(lambda) and E RMS is:

Therefore the **lambda** decided is : **0.001.** Since at the point around -3 in the log lambda graph we have observed a sudden jump. So in order to reach an optimal tradeoff value, we will select this point.

**Result of Step 4:**

We decided our optimal solution by tuning the parameters as follow:

**M = 49**

**S = 0.6**

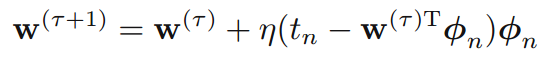
**Lambda = 0.001**

**Minimum error: 0.5378**

**Method 2: Stochastic Gradient Descent**

This model calculates the weights parameter w in a sequential manner. These are also known as on-line algorithms.

We use to assume model parameters initially to some value and consider data points one each at a time while updating the model parameter at each iteration. The algorithms that we are going to use is called stochastic gradient descent, also known as sequential gradient descent. The formula to calculate weight is:



Tau : it is iteration parameter

* : it is the learning rate that controls the values of w parameters

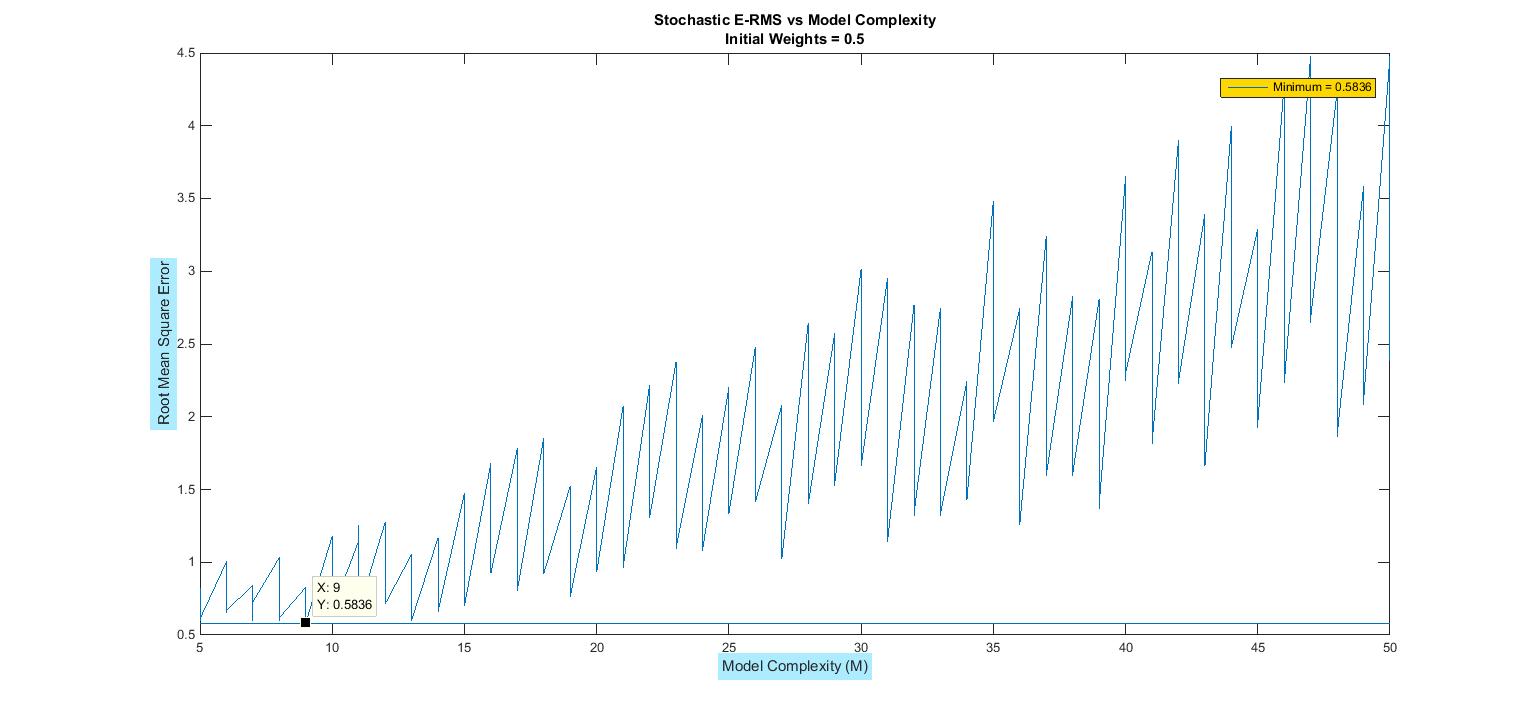
In this approach we will move in a direction with a step that is defined by this learning parameter in order to find the local minima with respect to the visible space of the current location. While finding the local minima we tend to converge a minimum point that could be the global minima of the input space with a very probability.

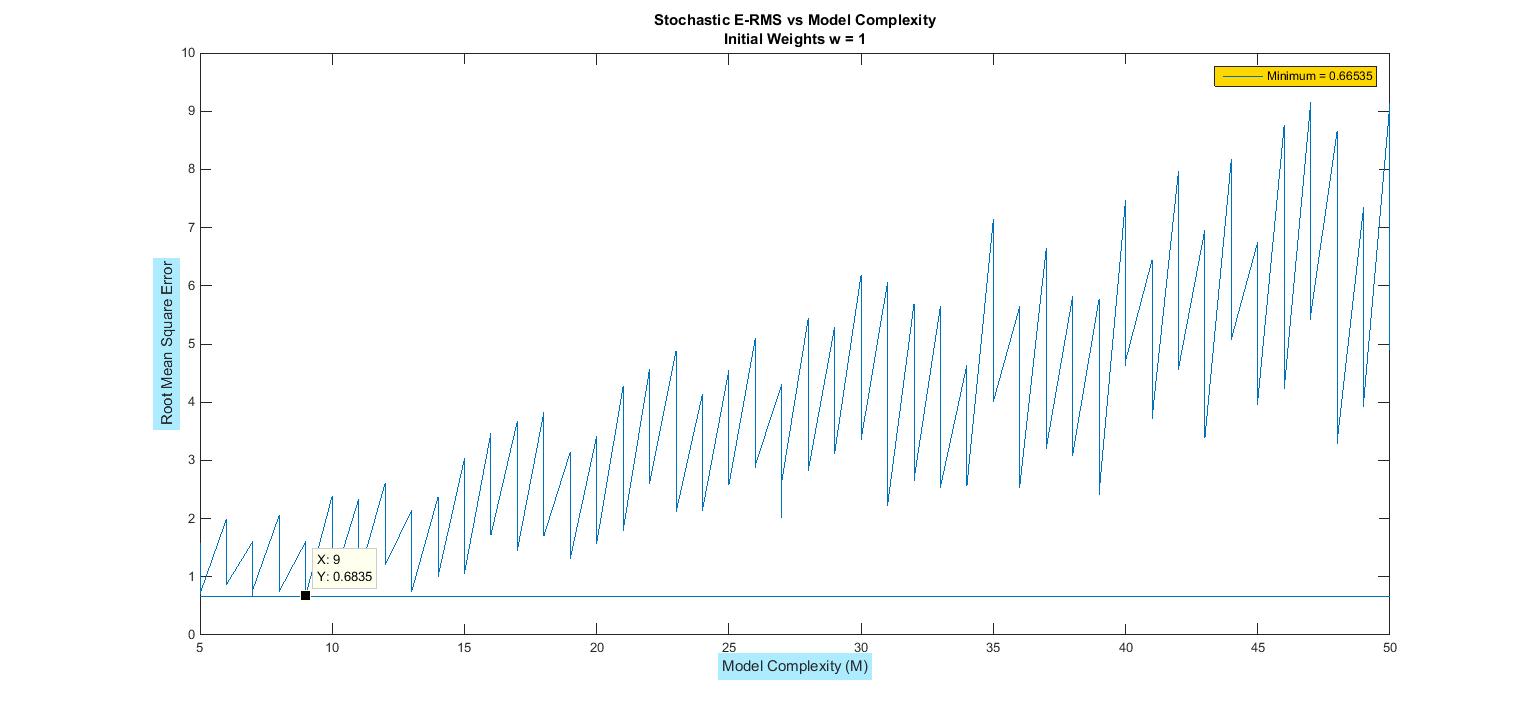
Hence we need to keep our learning parameter in a very confined range so that while taking the steps we should not miss the local minima point and led to altogether a different answer.

**Implementation to the current problem:**

1. Model complexity have been tested over a range of 5 to 50 with design matrix calculated with the randomly generated mean from the collection.
2. **Choose the initial valued of w**: two cases have been considered with values **0.5** and **1.**
3. Values of the **learning rate parameter** have been defined in a **range** of **.01 to 1**
4. RMS error is calculated at each step after the calculation of w parameters.
5. **Stop Condition:** When the change in the error values become less to constant, then the loop should be break.

**Tuning the Model Complexity:**

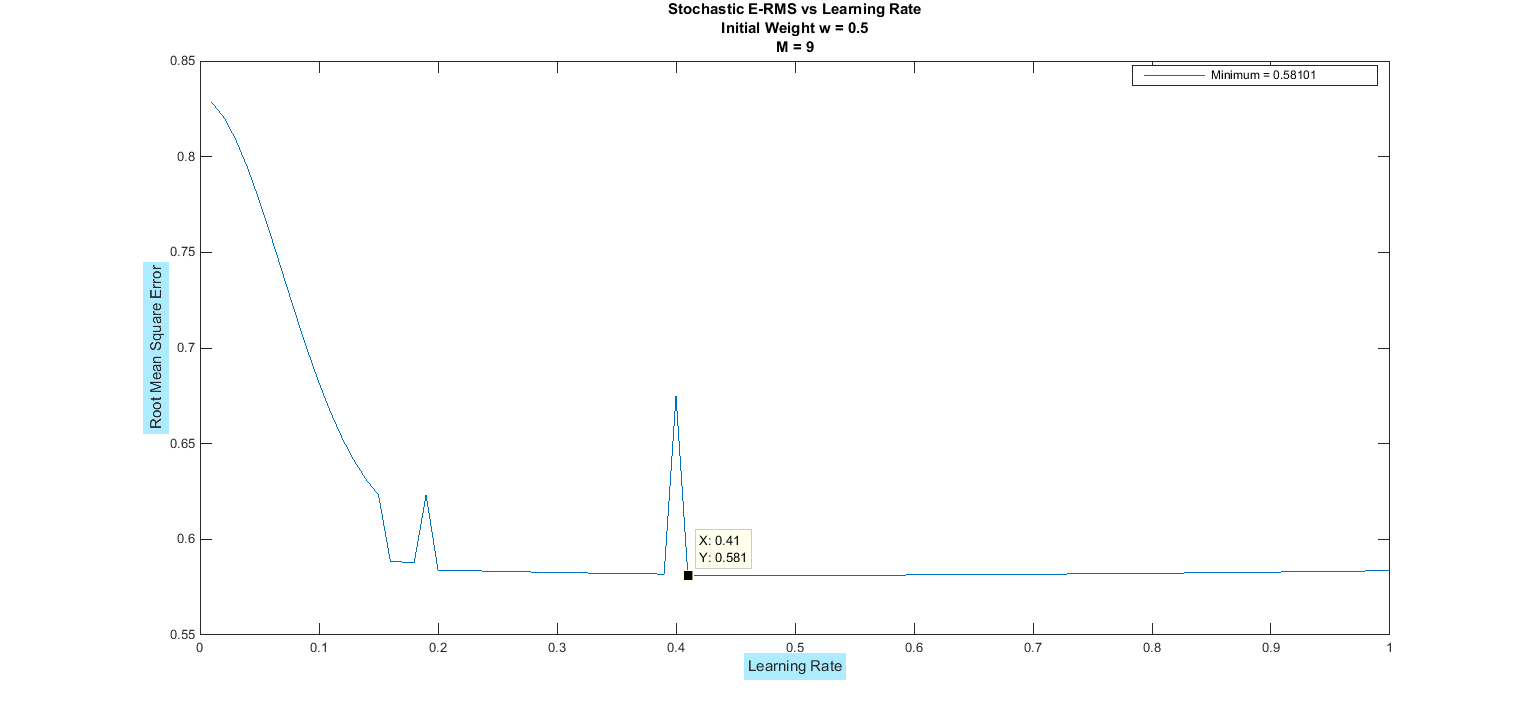
Following plots were obtained when the model is trained over the range of 5 to 50 Model complexity.



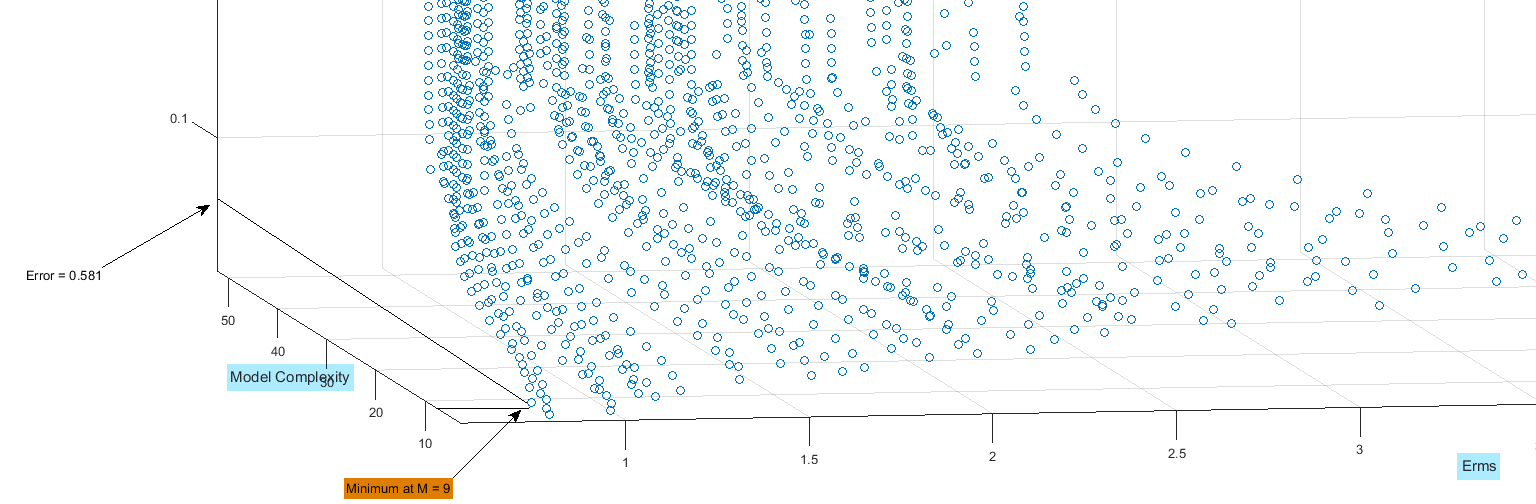
* It can be observed that the minimum error is being observed at the Model complexity of **M = 9** in both the plots with **w = 0.5** and **w = 1** as initial values of w.
* Therefore final tuned value for M will be taken as **M = 9.**
* Minimum **error** observed is **0.5836** for **w = 0.5**

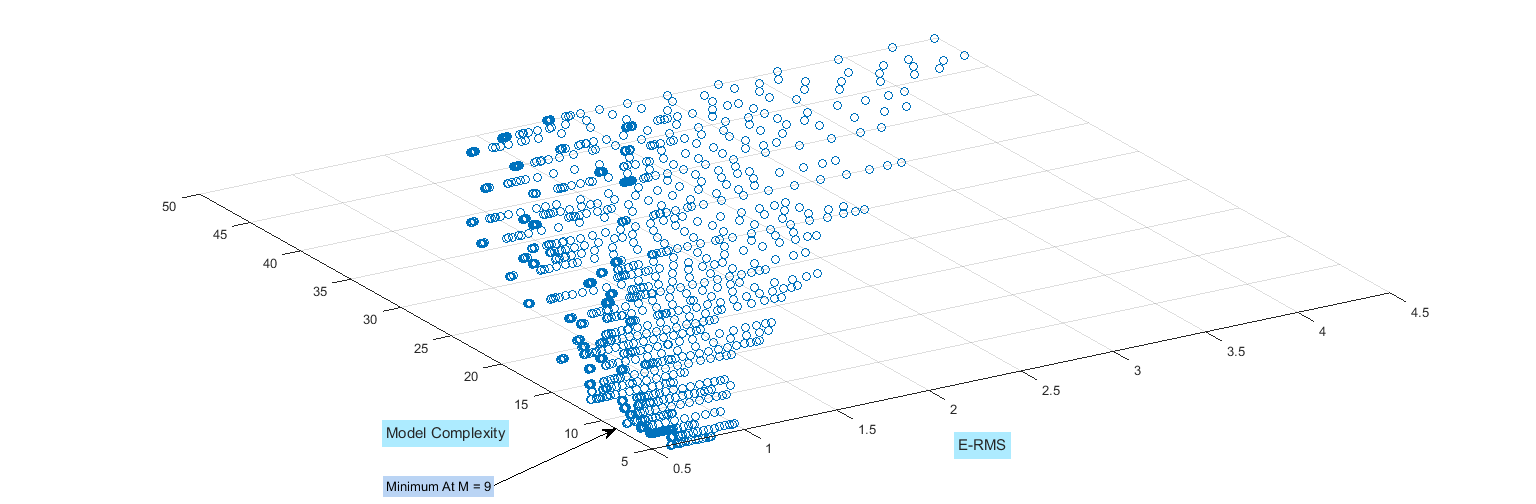
**Tuning the Learning Rate:**

As mentioned above, that the learning rate has been moved in the **range** of **.01 to 1.**

Therefor we will observed below plot for **M = 9**

As we can observe that the Minimum error coming is **0.581** at the learning rate value at **0.41**

****



**Optimal Solution to the Method 2:**

The final tuned values for this learning model is

* **= 0.41**

**M + 1 = 10**

**For initial w = 0.5**

**RMS Error Observed = 0.581.**

**Comparison between both the Methods:**

The error we got in Stochastic Model is in sync with what we got in 1st Method of Closed form Likelihood solution. Hence the **range of Error** is within **0.5 to 1.**