Project 1: Linear Regression with Basis Functions

Name : Warren Mendonca

UB Num : 501348345

1. Objective

This project is 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

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

where w={w0,w1,…..,w_{m-1}} is a weight vector to be learnt from training samples and phi = {phi0,phi1,……,phi_{m-1}} is a vector of M basis functions. Each basis function phi_j(x),j=0….M-1 converts input vector x into a scalar value. An example is the Gaussian basis function

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{(\mathbf{x} - \boldsymbol{\mu}_j)^2}{2s^2}\right)$$

where mu_j is a vector in feature space and *s* is an isotropic spatial scale. The training dataset consists of N exemplar vectors X={x1,…..,x_n} together with the corresponding target values t={t1,…..,t_n}

2. Parameters

M: is the number of basis functions used. The model's complexity increase with an increase in the value of M.

Phi: is the basis function used in the model. Each function has a different mean and a common isotropic spatial scale s. Phi is used to create a design matrix. The design matrix contains data on the independent variables in statistical models which attempt to explain observed data on a response variable in terms of the explanatory variables. In our case Phi is calculated using the following expression:

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{(\mathbf{x} - \boldsymbol{\mu}_j)^2}{2s^2}\right)$$

Lambda: Lamda is a regularization parameter. It is added in order to avoid overfitting. We use the training dataset to reach an effective value of lambda. A moderate value of lambda leads to lesser overfitting and minimizes the amount of error encountered. However an extremely large value for lambda is also not desirable.

3. Program Details

1. Run the train_cfs.m using the training dataset. Training dataset is saved in the .mat file project1_data.mat which is loaded in the program. The project1_data.mat file contains all the extracted feature vectors from the provided Microsoft LETOR Dataset file – Querylevelnorm.txt.
2. The training program is used to reach a good value for the M, Phi, Lambda.
3. Using the values for the parameters achieved from the training set we can use the prediction function. Use the test_cfs.mat file to use the prediction function to verify if the E_{rms} falls at the expected value.
4. Run the train_gd.mat file using the training dataset. Training dataset is saved in the .mat file project1_data.mat which is loaded in the program. The project1_data.mat file contains all the extracted feature vectors from the provided Microsoft LETOR Dataset file – Querylevelnorm.txt.
5. The training program is used to tune a value for the learning rate.
6. Using the value for the learning rate achieved from the training set we can use the prediction function. Use the test_cfs.mat file to use the prediction function to verify if the E_{rms} falls at the expected value.

Number of samples used in the various phases of the program are:

1. Training phase: 69623*0.8=55698. This is the first 55698 rows of the dataset provided. It is 80% of the entire dataset provided.
2. Validation phase: 69623*0.1=6962. This set of rows following the training set data which we used to train the test_cfs.m file. This is 10% of the dataset after training data.
3. Testing phase: 69623*0.1=6962. This is the last 6962 rows of the data set provided. 10% from the end of the validation set to the end of the dataset.

The equation used in the program test_cfs.mat uses the form of the quadratic regularizer extension which is given by the below equation:

$$\mathbf{w}_{ML} = (\lambda\mathbf{I} + \Phi^T\Phi)\ \ {}^{1}\Phi^T\mathbf{t}$$

For the stochastic gradient descent the program test_gd.m we use the equation:

$$\mathbf{w}^{(\tau+1)} - \mathbf{w}^{\tau} + \eta(t_n - \mathbf{w}^{(\tau)T}\phi_n)\phi_n$$

Using this we find the maximum likelihood solution which is the same as minimizing the squares error

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n-1}^{N}\{t_n - \ \mathbf{w}^T\phi(\mathbf{x}_n)\}^2$$

To this we add an additive regularization term to avoid over-fitting given by,

$$E_W(\mathbf{w}) = \frac{1}{2}\sum_{j-0}^{M\ \ 1}|w_j|^q$$

Thereby giving an error function of,

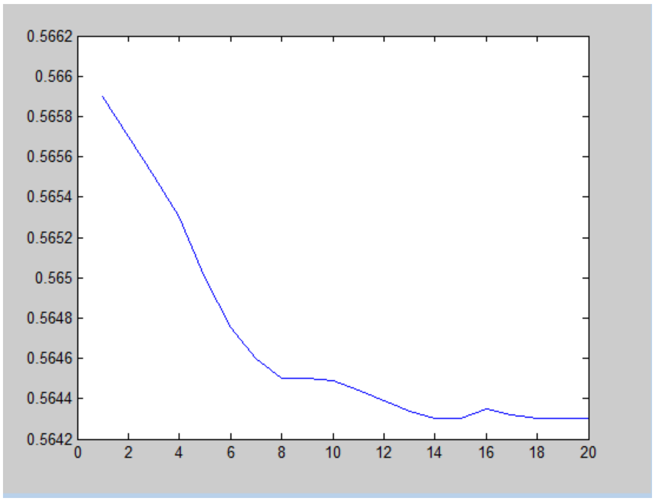
$$E(\mathbf{w}) = E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

4. Evaluation of model

I. CLOSED FORM MAXIMUM LIKELIHOOD SOLUTION

a. Determining the model complexity M

The graph of Model Complexity vs Erms for the training set is as shown below



We observe by varying all three parameters initially that we can get an estimated fixed point for lambda and s. The value of lambda is fixed at a value of 16 and s at 3 at which we get least error. We can now estimate the model complexity for by varying this quantity and viewing the corresponding Erms. From the above graph of plotted values we observer that the value for the error is least at the model complexity M=14.

The graph of Model Complexity vs Erms for the training set, validation set and the test set are as shown below

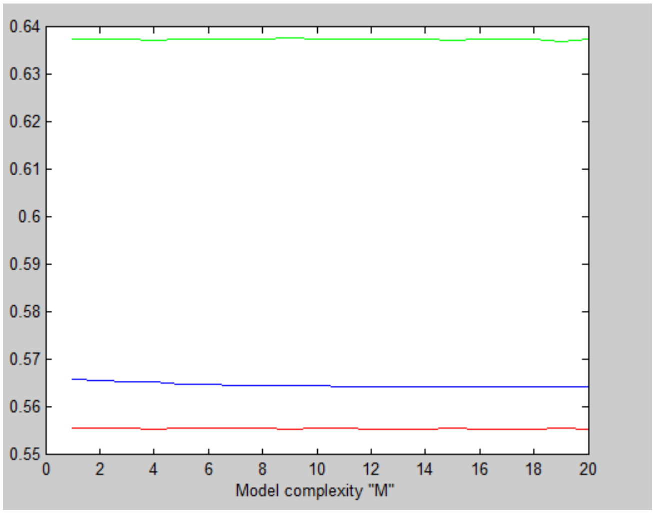


Fig: the training set is marked in blue, validation set in red and the test set is marked in green.

b. <u>Determination of the constant lambda</u>

During the training phase we try different values of M combined with the different values for lambda. Initially for a low value of M we see Erms increasing with an increase in lambda. However as the model complexity increase we notice that the value lambda at 16 provides a good decrease in the Erms value for all model complexities above this value. The graph of Regularization constant lambda vs Erms for the training set, validation set and the test set are as shown below

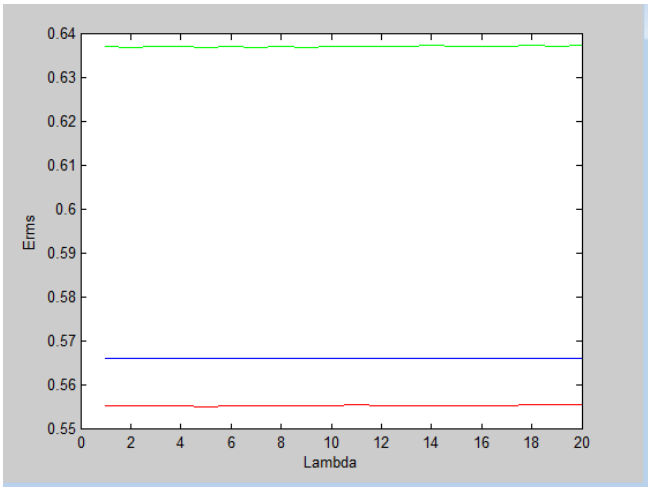


Fig: the training set is marked in blue, validation set in red and the test set is marked in green.

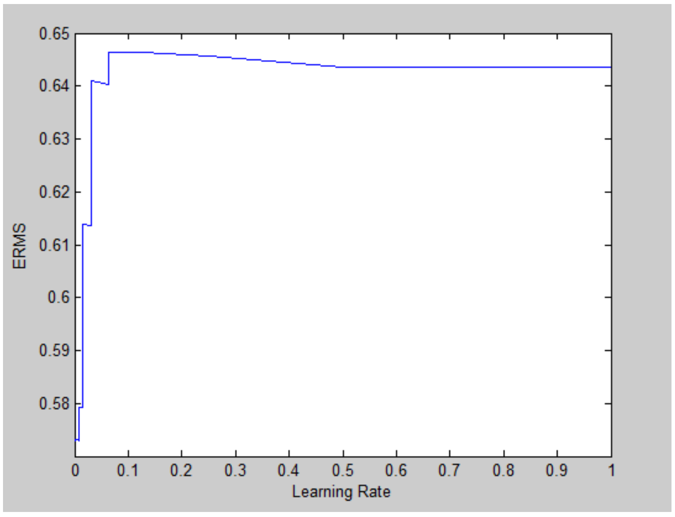
The value of Model complexity is fixed at 14 and s at 3 for which achieve least error.

II. STOCHASTIC GRADIENT DESCENT

a. Determining the learning rate:

For tuning the learning rate we vary the rate of the learning rate depending on the value of the calculated Erms. Stochastic Gradient descent algorithm starts off at a high rate of Erms and by tuning the value of the learning rate over a number of iterations we can find a good value for it. It can be observed that Erms started at a high value and as we continued tuning the value for learning rate we notice that the Erms jumps around this value till it reaches a point where the Erms is almost constant for every change in the value of the learning rate. The model complexity is kept constant at 4 for learning.

The graph of the learning rate to the Erms is shown below:



We notice the initial value of learning rate is quite high. But as we modify the learning rate for every successive iteration we notice that it eventually reaches a point where the Erms is generally constant for a value of learning rate. We can therefore mark the first such value as a good estimate for the learning rate. [learning rate = 7.812500e-03]

b. <u>Determining the model complexity for stochastic gradient:</u>

We can determine a good value for model complexity M by keeping the value of alpha constant as learned in the previous step and varying the model complexity. For all the range of values we determine an optimal value for the model complexity. The graph of model complexity vs the obtained values of Erms is shown below:

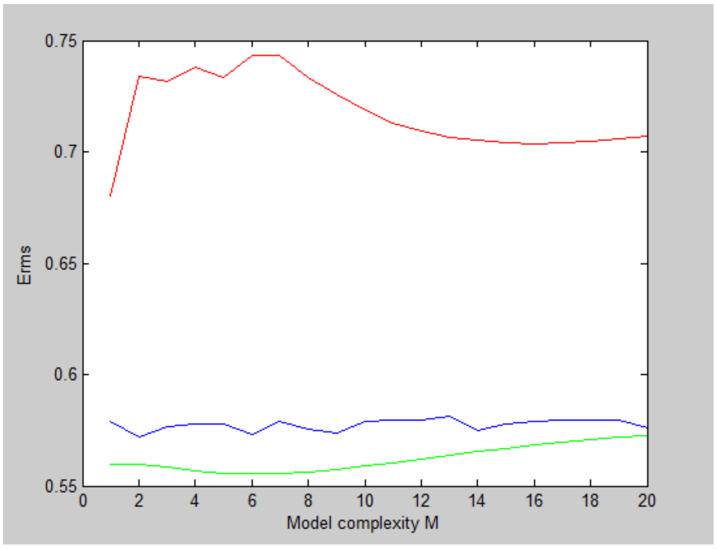


Fig: The values for Erms vs model complexity for the training phase is show in blue, green for the validation phase and red for the test phase

From the graph for the training set we have marked the value for Model complexity at the point where we get the lowest value of Erms (M=6)

c. Validation phase for Stochastic Gradient :

During the validation phase we use data 10% of the data given to check if the learning rate falls in the expected range. When we run the program valid_gd.m we notice that the learning rate falls in approximately the same range as the one we achieved in the training phase. We arrive at this conclusion by viewing the range of Erms values with respect to different values of the learning rate. The plot of these quantities with respect to each other is displayed below.

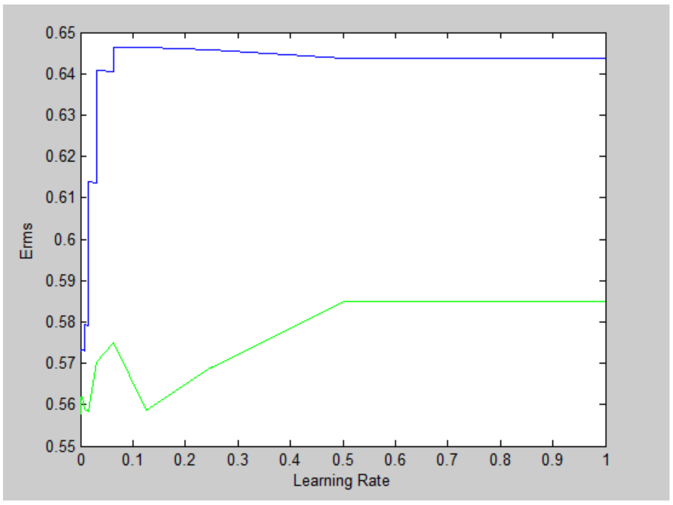
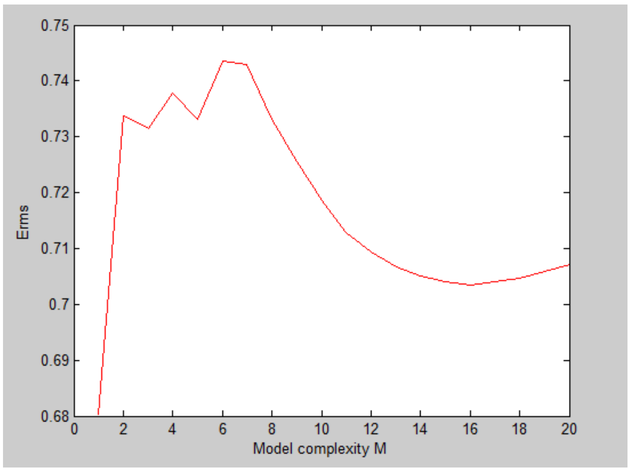


Fig: Values for raining data set is marked in blue and those for the validation phase are marked in green.

d. Value range for the test data

When we use the value of alpha against the different model complexities we find the predicted lowest range of values for the error. Below is the plot of Erms vs the model complexity (range from 1-20).



5. Comparison of models

For comparing both the models we draw a graph of the values for Erms determined by the two models vs the different model complexities we see. The graph of the two models drawing the comparison has been shown below:

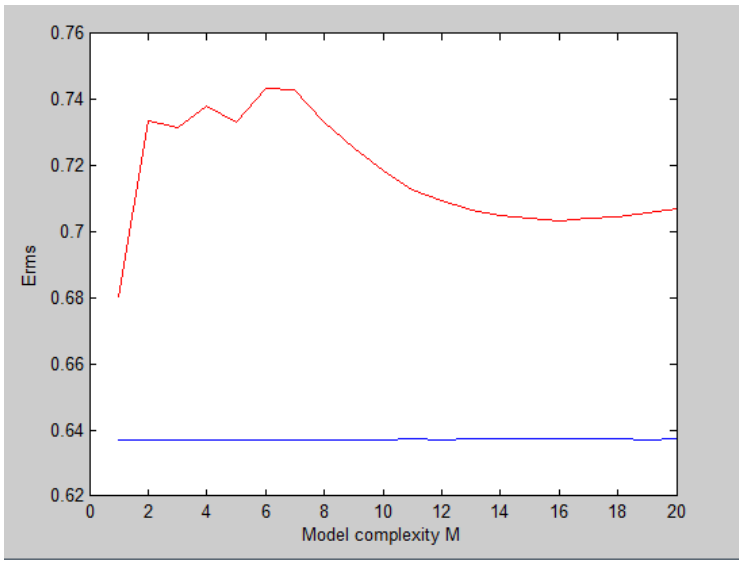


Fig: The values of Erms for the closed form of maximum likelihood solution is marked in blue whereas those achieved from Stochastic Gradient descent is marked in red.

From the graph we can conclude that for the given set of data the Erms value is lowest when we use the closed form for maximum likelihood solution.