Project 1.2

LeToR using Linear Regression

Illa Nuka Saranya
50248926

10-Oct-18
Saranya@buffalo.edu

Contents

1. PROBLEM DEFINITION 2
2. IMPLEMENTATION 2
2.1 Preprocessing of the dataset: 2
2.2 Fetching and Preparing the Dataset 2
2.3 Splitting the Dataset 3
2.3.1 Training Dataset 3
2.3.2 Validation Dataset 3
2.3.3 Test Dataset 3
2.4 Closed-Form Solution for finding optimal weights: 4
2.4.1 Results 6
2.5 Gradient Descent Solution for linear regression 13
2.5.2 Graphs 14
2.5.1 Tables 15
3. Conclusion 17

1. PROBLEM DEFINITION

The goal of this project is to use machine learning to solve a problem that arises in Information Retrieval, one known as the Learning to Rank (LeToR) problem. We formulate this as a problem of linear regression where we map an input vector x to a real-valued scalar target y(x, w).

There are two tasks:

1. Train a linear regression model on LeToR dataset using a closed-form solution.

2. Train a linear regression model on the LeToR dataset using stochastic gradient descent (SGD).

DATASET: In this project, the "QueryLevelNorm" version of Microsoft LETOR 4.0(MQ2007) is used as dataset which consists of 69623 query-document pairs (rows), each having 46 features.

2. IMPLEMENTATION

2.1 Preprocessing of the dataset:

- "QueryLevelNorm.txt" file is preprocessed in a way that it can be used to train and test the model in using python. We generate two files of csv format namely RawTarget.csv and RawData.csv.
- Querylevelnorm_RawTarget.csv consists of only the first column values of the given dataset that has relevance labels of the 69623 rows that takes one of the discrete values 0, 1 or 2.
- Querylevelnorm_RawData.csv consists of all other column values of all the 69623 rows of given dataset.

2.2 Fetching and Preparing the Dataset

- Convert the Querylevelnorm_RawTarget.csv file into a vector named RawTarget using GetTargetVector method that reads the csv file line by line to generate a list containing only the target values.

  The length of this vector or the list is 69623.

- Convert the Querylevelnorm_RawData.csv file into a data matrix named RawData that initially has 69623 rows and 41 columns which upon deleting 5 columns(that has variance equal to zero) and doing a transpose converts it into a matrix of 41 rows and 69623 columns.

  The final shape of this RawData matrix is (41, 69623).

2.3 Splitting the Dataset

Splitting of the dataset can be done in many ways but here we do the split of training, validation and testing data in continuous way using the percentage values to find the lengths.

2.3.1 Training Dataset: The sample of data used to fit the model from which it sees and learns.

- The entire dataset consists of 69623 query-document pairs (rows), each having 46 features. We use 80% of this dataset as Training dataset i.e 55699 rows of the given dataset are into training.
- TrainingTarget and TrainingDataMatrix are the two arrays that have Training target and target data values.
- TrainingTarget is a single dimensional array that has 80% of the RawTarget vector values from the beginning.
- TrainingDataMatrix is a (41, 55699) shaped array that has 80% of RawData columns keeping t he number of rows same.
- 5,6,7,8,9 columns of the given raw dataset matrix are deleted as the variance of those columns is zero and hence we can't compute inverse of the bigsigma matrix in future computation.

2.3.2 Validation Dataset: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper-parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

- We use 10% of our given dataset as validation data i.e 6962 rows of raw datatset are into validation.
- ValidationTarget and ValidationDataMatrix are the two arrays that have Validation data.
- ValidationTarget is a single dimensional array that has 10% of the RawTarget vector values from that begins from the end of TrainingTarget values.
- ValidationDataMatrix is a (41, 6962) shaped array that has 10% of RawData columns that begi ns from the end of TrainingData values keeping the number of rows same.

2.3.3 Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

- We use 10% of our given dataset as testing data i.e 6962 rows of the raw dataset are into testing.
- TestingTarget and TestingDataMatrix are the two arrays that have Testing data.
- TestingTarget is a single dimensional array that has 10% of the RawTarget vector values from that begins from the end of ValidationTarget values.

- TestingDataMatrix is a (41, 6962) shaped array that has 10% of RawData columns that begins f rom the end of ValidationData values keeping the number of rows same.

The next step is to train the linear regression model using the training dataset to find the optimal weights that gives the best fit of solution.

We find this using two approaches namely, Closed-form solution and Stochastic Gradient Descent Solution

2.4 Closed-Form Solution for finding optimal weights:

The closed-form solution with least-squared regularization, as defined as

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

λ - the least-squared Regularization and can be used as one of the hyper-parameters to tune the model.

t - {t1, .., tN } is the vector of outputs in the training data

Φ - is the design matrix

$\Phi_j(x)$ - Basis functions that converts the input vector x into a scalar value. In this project, you are required to use the Gaussian radial basis function

$$\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$$

- From the above equation, we need to get the phi matrix on training dataset called TRAINING_PHI to find the weights.
- Design matrix consists of scalar values produced by using radial Gaussian functions. The number of Gaussian functions we use is equal to the number of clusters obtained by using k means clustering of dataset.
- Determine the mean and the variance of the clustered datasets to compute the basis function values.

The formula can be modified to

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^\top \Sigma_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right)$$

- We find the variance of all the Training Dataset values on every feature.
- That is we have in total of 41 variances which will be the diagonal values of BigSigma. Calculate the inverse of BigSigma as the modified formula suggests.
- By having the mean and inverse BigSigma values, it is easy to compute scalar values of the design matrix using radial Gaussian functions.
- The shape of the design matrix is (59699, 10).
- Using the design matrix (**Φ**), target values (t) and the least squared regularization value (**λ**)**,** we
- compute the weight matrix

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

- This process of computing weights as described above is called as closed-form solution of weights.
- Using this weight matrix and design matrices (**Φ(x)) of validation and testing**, compute the output values on validation and testing dataset respectively.
- The regression line predicts the average y value associated with a given x value. It gives the measure of the spread of the y values around that average. To do this, we use the root-mean-square error (Root mean squared error).
- Calculate the root mean squared errors between actual target values of Validation stored in ValidationTarget with the computed output values (using validation Design matrix and the weight matrix obtained using closed form).

The possible hyper-parameters that can be changed here are:

- The number of basis functions (No of clusters used in K-Mean clustering) which will change the MuMatrix that has means of different clusters**. (μ**)
- The value of lambda (**λ**)
- The data split on training, validation and testing on the actual raw dataset.
- Scalar values by which co-variance matrix is multiplied.

2.4.1 Results

The following table gives values of accuracy on training, validation and testing datasets for different hyper-parameter values

Training percentage:70, Validation percentage: 15

Testing percentage : 15

We observe that the best fit occurs when we have divided the dataset into 7 clusters with the regularization rate of 0.03.

Training accuracy : 74.328, Validation accuracy: 73.714, Testing Accuracy: 70.944

Training Erms: 0.549, Validation Erms: 0.55, Testing Erms: 0.605

It is observed that as regularizing rate increases, the Erms value of the model decreases till some extent and increases later on. The value of regularization results in under-fitting and overfitting of data.

2.4.1.1 Tables:

Tables are plotted by changing

- **Different values of μ (Changing number of basis functions),**
- **Changing the different values of BigSigma inverse(changing the scalar value by which bigsigma is multiplied)**
- **Different values of regularization (lambda)**
- **Different values of M (Changing different values of no of clusters)**

Accuracy values

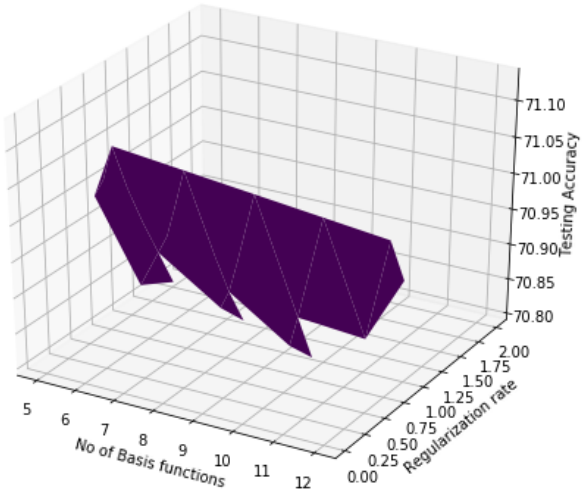
Clusters	Lambda	Accuracy Training	Accuracy Validation	Accuracy Testing	difference
5	0.03	74.641	74.05	71.136	0.592
5	0.05	74.641	74.05	71.136	0.592
5	1.0	74.732	74.05	71.165	0.682
5	2.0	74.814	74.184	71.222	0.63
7	0.03	74.328	73.714	70.944	0.613
7	0.05	74.336	73.676	70.925	0.66
7	1.0	74.393	73.58	70.944	0.813
7	2.0	74.494	73.647	71.05	0.846
10	0.03	74.046	73.284	70.781	0.763
10	0.05	74.098	73.284	70.896	0.814
10	1.0	74.364	73.533	70.877	0.832
10	2.0	74.442	73.6	71.002	0.843
12	0.03	74.065	73.102	70.868	0.963
12	0.05	74.09	73.13	70.801	0.959
12	1.0	74.467	73.6	71.088	0.867
12	2.0	74.467	73.676	71.069	0.791

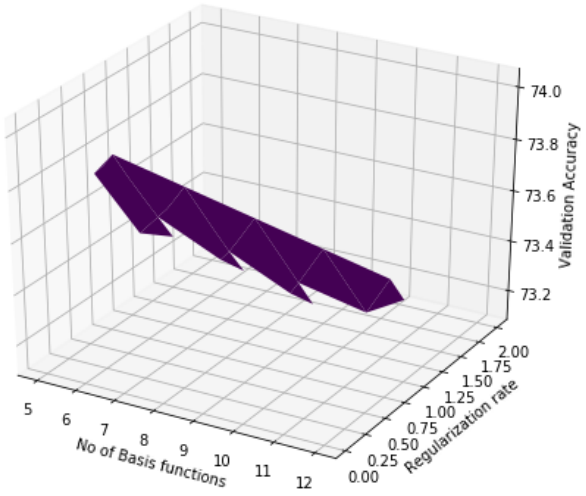
Erms Values

Clusters	Lambda	ErmsTraining	Erms Validation	Erms Testing
5	0.03	0.551	0.55	0.606
5	0.05	0.551	0.55	0.606
5	1.0	0.551	0.55	0.606
5	2.0	0.551	0.55	0.606
7	0.03	0.549	0.55	0.605
7	0.05	0.549	0.55	0.605
7	1.0	0.55	0.55	0.604
7	2.0	0.55	0.55	0.604
10	0.03	0.548	0.549	0.605
10	0.05	0.548	0.549	0.605
10	1.0	0.549	0.55	0.604
10	2.0	0.549	0.55	0.604
12	0.03	0.548	0.548	0.605
12	0.05	0.548	0.548	0.605
12	1.0	0.549	0.549	0.604
12	2.0	0.549	0.549	0.604

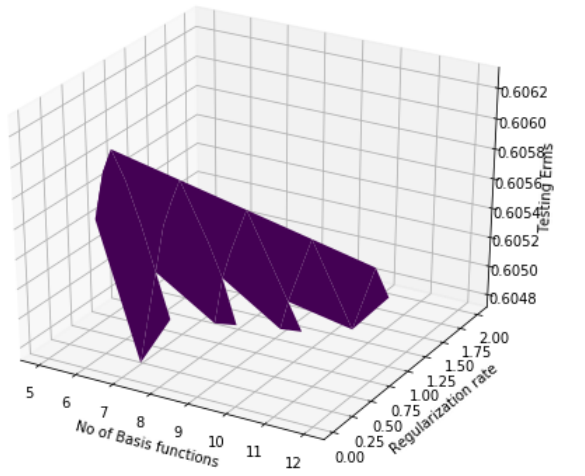
2.4.1.2 Graphs

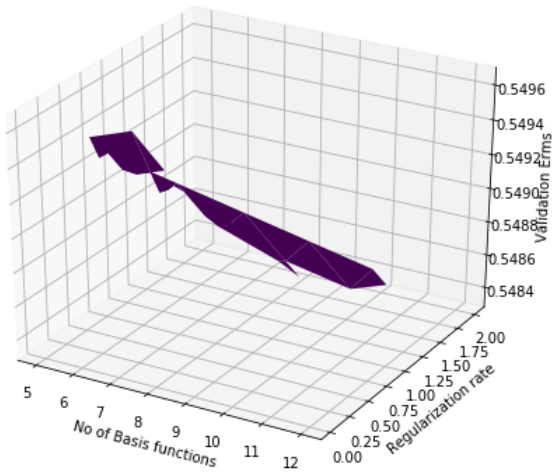
Graph plotted for Accuracy, basis functions and regularization rate

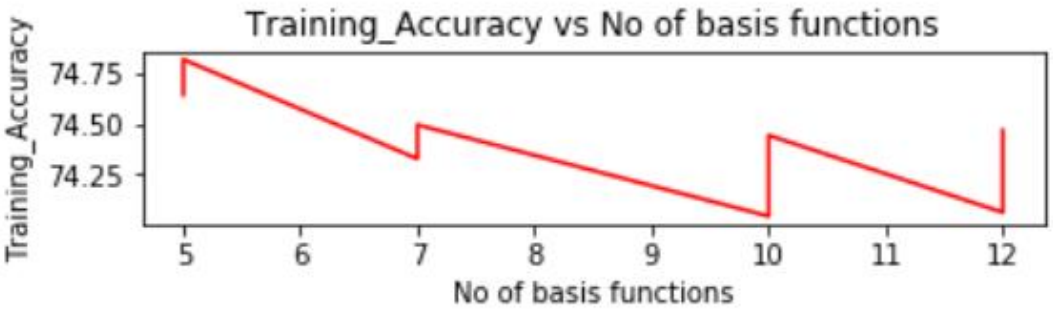


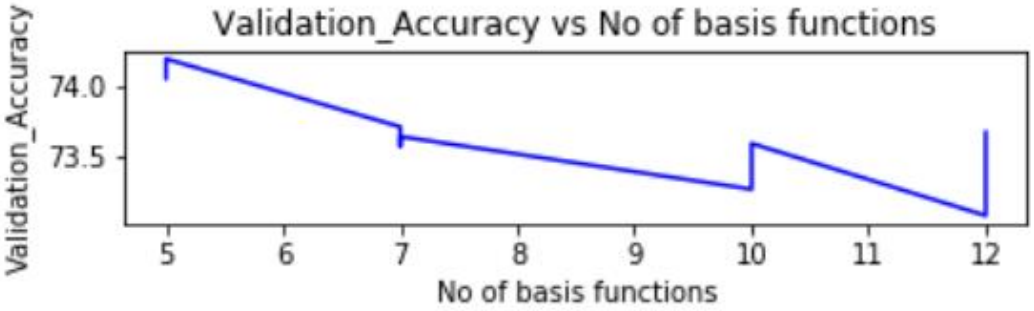


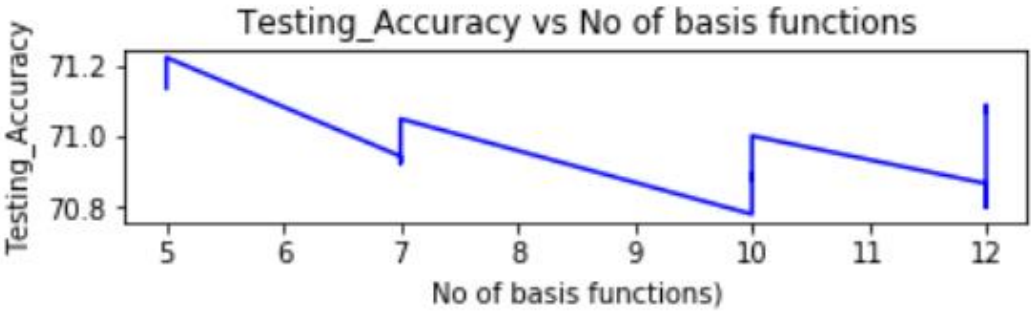
Graph plotted for Erms values, basis functions and regularization rate

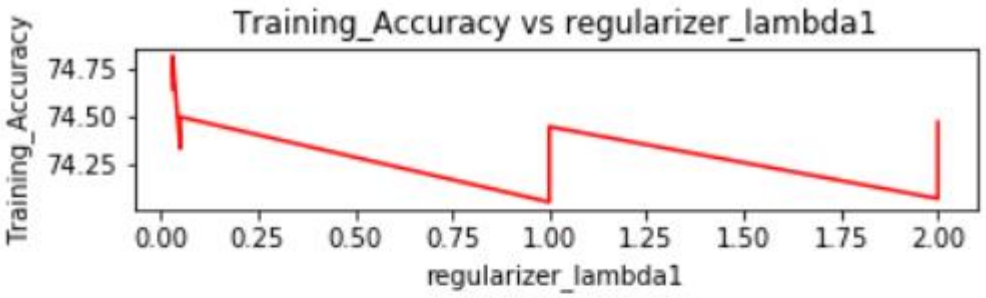


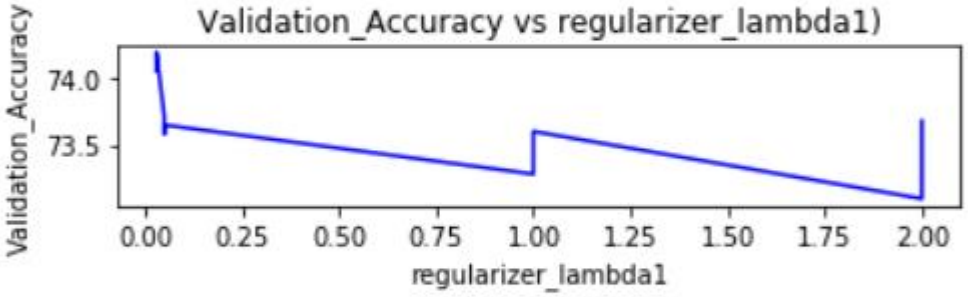


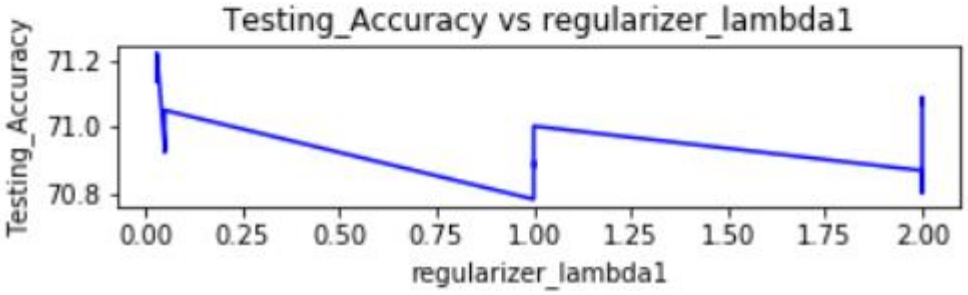

Training_Accuracy vs No of basis functions
Training_Accuracy
74.75
74.50
74.25
5
6
7
8
9
10
11
12
No of basis functions

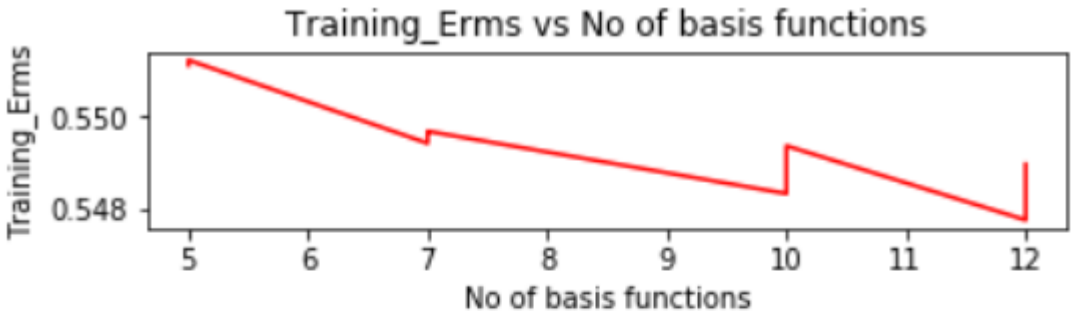

Validation_Accuracy vs No of basis functions
Validation_Accuracy
74.0
73.5
5
6
7
8
9
10
11
12
No of basis functions

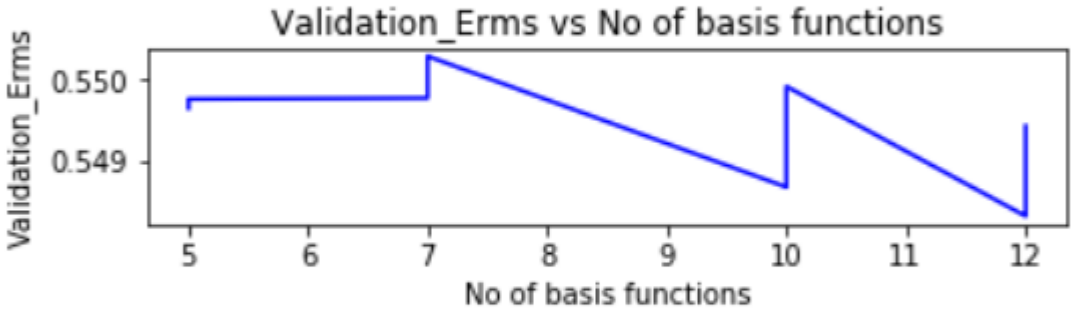

Testing_Accuracy vs No of basis functions
Testing_Accuracy
71.2
71.0
70.8
5
6
7
8
9
10
11
12
No of basis functions)

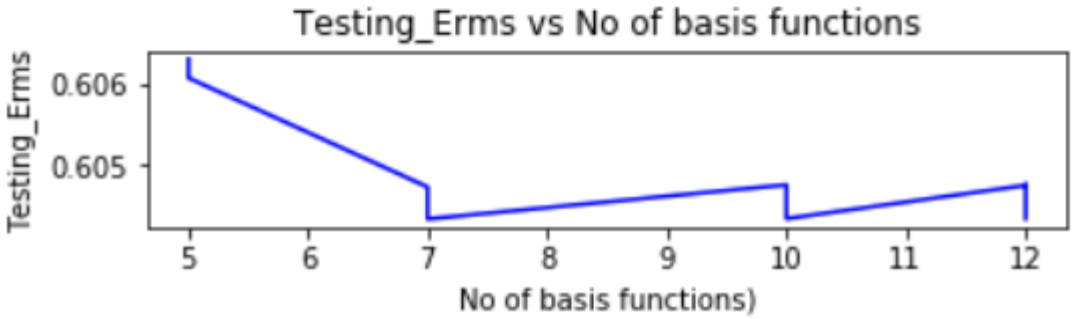

Training_Accuracy vs regularizer_lambda1
Training_Accuracy
74.75
74.50
74.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
regularizer_lambda1

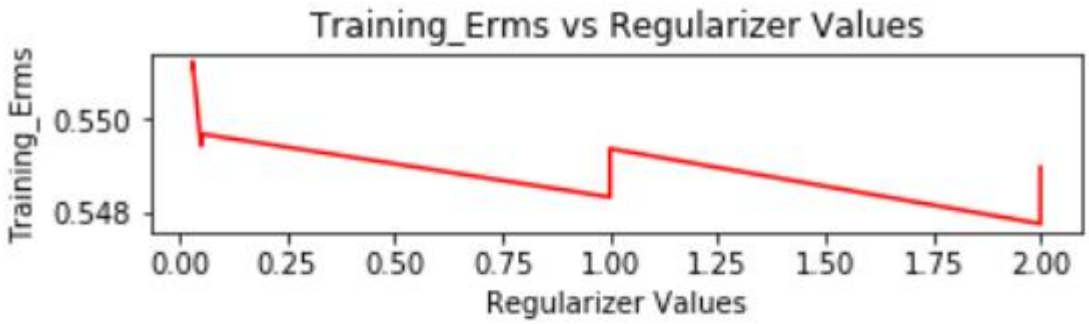

Validation_Accuracy vs regularizer_lambda1)
Validation_Accuracy
74.0
73.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
regularizer_lambda1

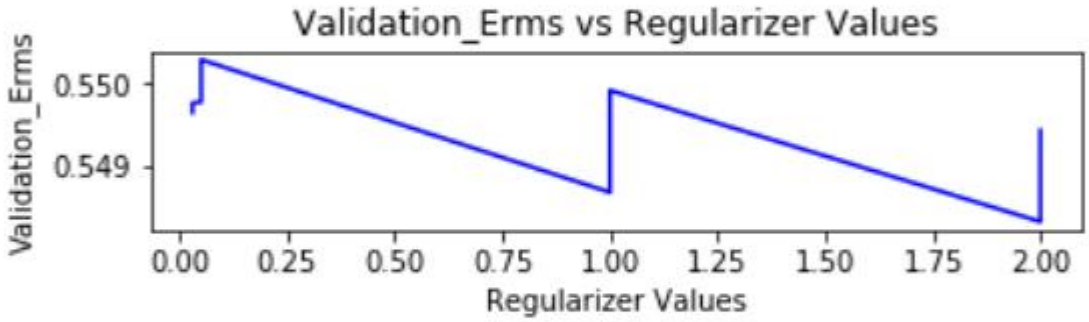

Testing_Accuracy vs regularizer_lambda1
Testing_Accuracy
71.2
71.0
70.8
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
regularizer_lambda1

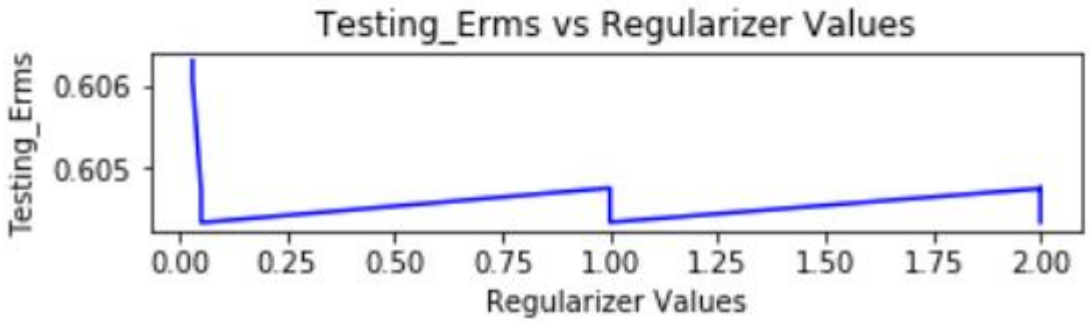

Training_Erms vs No of basis functions
Training_Erms
0.550
0.548
5
6
7
8
9
10
11
12
No of basis functions


Validation_Erms vs No of basis functions
Validation_Erms
0.550
0.549
5
6
7
8
9
10
11
12
No of basis functions


Testing_Erms vs No of basis functions
Testing_Erms
0.606
0.605
5
6
7
8
9
10
11
12
No of basis functions)


Training_Erms vs Regularizer Values
Training_Erms
0.550
0.548
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Regularizer Values


Validation_Erms vs Regularizer Values
Validation_Erms
0.550
0.549
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Regularizer Values


Testing_Erms vs Regularizer Values
Testing_Erms
0.606
0.605
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Regularizer Values

2.5 Gradient Descent Solution for linear regression

The stochastic gradient descent algorithm first takes a random initial value of weights and keeps on updating the values of it using the following formula

$$w^{(\tau+1)} = w^{(\tau)} + \Delta w^{(\tau)}$$

$w^{(\tau)}$ -- current weights

$w^{(\tau+1)}$ -- updated weights

$\Delta w^{(\tau)}$ -- $-\eta^{(\tau)} \nabla E$ where η – learning rate, ∇E - gradient error

Root Mean Square (RMS) error, defined as $E_{RMS} = \sqrt{(2E(w^*)/N_v}$ where w^* is the solution and N_V is the size of the test dataset

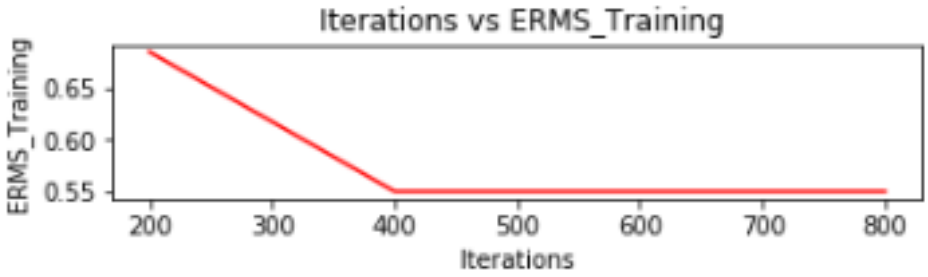
- In the process of finding optimal weights using gradient descent solution, set the initial weights randomly.
- **Error regularization factor** (lambda) can be treated as a hyper-parameter.
- **Learning rate decides** how big each update statement would be and choosing it too big could lead to divergence and choosing it too small could lead to intolerably slow convergence. Update the weights in iterations to find next set of weights using the current set of weights in a way as to minimize the E_{RMS}.
- We choose the number of iterations using the concept of early stopping to avoid overfitting. **Early stopping** is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as **gradient descent**. It works on the assumption that the performance on data can be improved only till certain number of iterations but after that there will be no significant improvement as such.

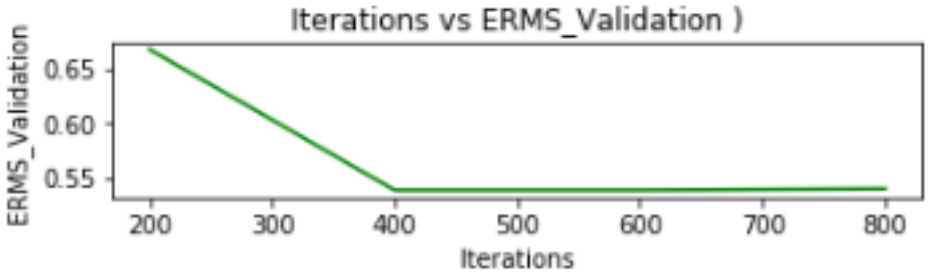
For every iteration, calculate the Training, validation and testing output by using the design matrix and updated weights.

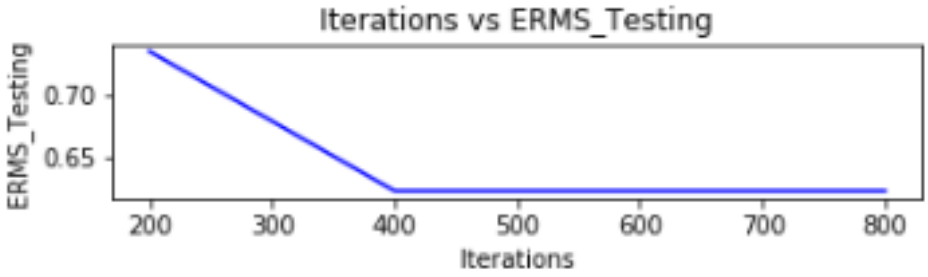
Also, compute the E_{RMS} values for each of the data split and append these values to the list to find the minimum value.

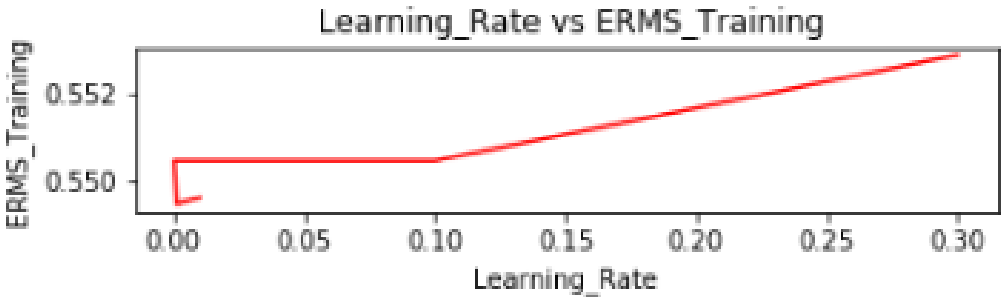
2.5.2 Graphs

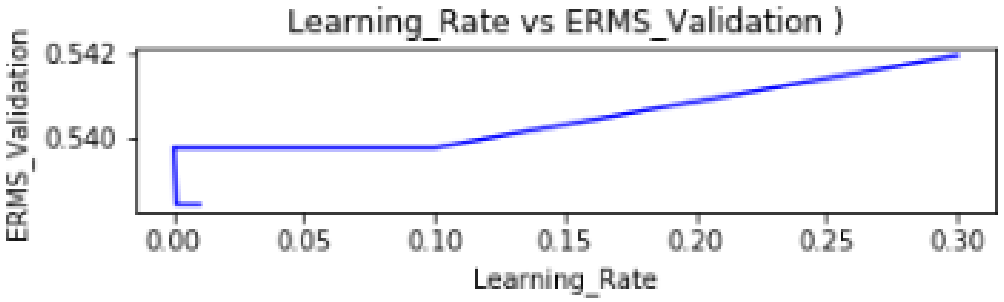
We observe that the weights become stagnant after 400 iterations and there is no much difference in the updated weights. Hence we do early stopping at 400 for optimal model

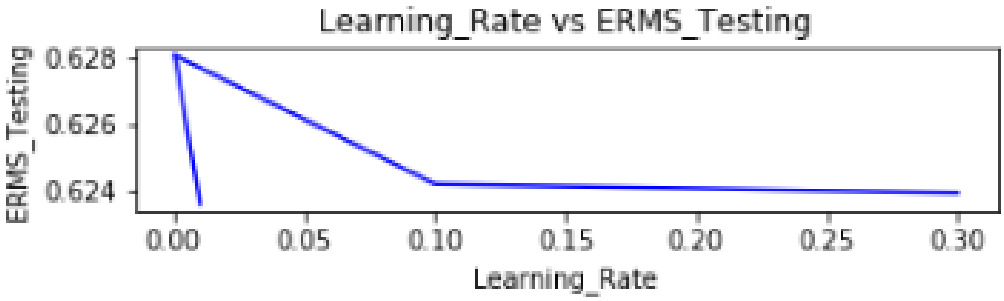












2.5.1 Tables

Tabulating the E_{RMS} values for different values of lambda, learning rate and M

Training percentage: 70

Validation percentage: 15

Testing Percentage: 15

Scalar by which BigSigma is multiplied:400

No of Clusters = 7,Regularization factor1= 2.0 ,Regularization factor2 = 2,Learning Rate = 0.01

Iterations	E_rms Training	E_rms Validation	E_rms Testing
200	0.68455	0.66878	0.73361
400	0.54962	0.53849	0.6236
600	0.54962	0.53849	0.6236
800	0.54962	0.53849	0.6239

- If the early stopping value is very less, the model is still underfit and the Erms value will be high.
- As the number of iterations increases, the Erms values decreases to some extent and the Erms values increases later on which means that the model overfits for more number of iterations.

Keeping the number of iterations = 400 and Lambda = 2, changing different values of learning rate we get following table.

Learning Rate	E_rms Training	E_rms Validation	E_rms Testing
0.01	0.5496	0.5384	0.6236
0.001	0.54945	0.5384	0.62783
0.0001	0.55044	0.53976	0.62783
0.1	0.55044	0.53976	0.6242
0.3	0.55286	0.54196	0.62393

- From the above table we observe that, the Erms value decreases with the increase in the learning rate to some extent and later on increases.
- The higher the learning rate, the faster the convergence of the model and vice versa. Too higher or too lower values of learning rate underfit or overfit the model.

M = 7
Lambda = 3
eta=0.01
E_rms Training = 0.54978
E_rms Validation = 0.55029
E_rms Testing = 0.6022
Accuracy Testing = 71.37522

3. Conclusion

We observe that stochastic gradient descent model has little higher accuracies than the closed form solution. We observe that there is no significant difference when we change the hyper parameters. The given model is not so accurate and gives only a maximum accuracy of 74.8 % on training and 70 % on testing data.The given model behaves almost the same for different hyper parameters and in both closed- form and stochastic gradient solution.