Report On : Project - 1.2 Learning to Rank using Linear Regression

Neeraj Ajit Abhyankar **UB Person No. : 50290958** **UBID : nabyank**

Objective :

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).

Learning to Rank for Information Retrieval :

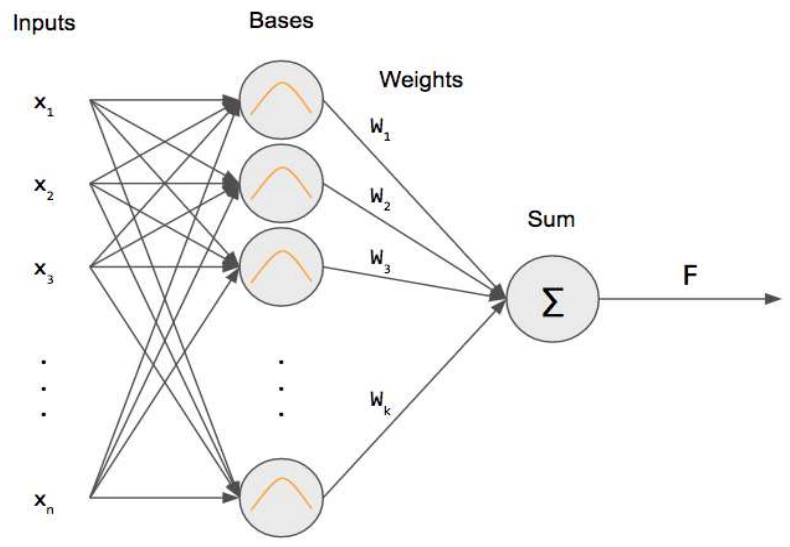
LETOR is a package of benchmark data sets for research on LEarning TO Rank, which contains standard features, relevance judgments, data partitioning, evaluation tools, and several baselines.

There are about 1700 queries in MQ2007 with labeled documents and about 800 queries in MQ2008 with labeled documents.

The LeToR training data consists of pairs of input values x and target values t. The input values are real-valued vectors (features derived from a query-document pair). The target values are scalars (relevance labels) that take one of three values 0, 1, 2: the larger the relevance label, the better is the match between query and document. Although the training target values are discrete we use linear regression to obtain real values which are more useful for ranking (avoids collision into only three possible values).

Radial Basis Function :

RBF's are very commonly used for regression or function approximation problems. We have some data that represents an underlying trend or function and want to model it. RBF nets can learn to approximate the underlying trend using many Gaussians/bell curves. An RBF net is similar to a 2-layer network. We have an input that is fully connected to a hidden layer. Then, we take the output of the hidden layer perform a weighted sum to get our output.



Linear Regression :

Linear regression is a **linear model**, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).When there is a single input variable (x), the method is referred to as **simple linear regression**. When there are **multiple input variables**, literature from statistics often refers to the method as multiple linear regression.

Stochastic Gradient Descent :

SGD also known as incremental gradient descent, is an iterative method for optimizing a differentiable objective function, a stochastic approximation of gradient descent optimization. It is called stochastic because samples are selected randomly (or shuffled) instead of as a single group (as in standard gradient descent) or in the order they appear in the training set.

Regularization :

One of the major aspects of training a machine learning model is avoiding overfitting. *The model will* have a low accuracy *if it is* overfitting. This happens because the model is trying hard to capture the noise in training dataset given. Noise is the data points that don't really represent the true properties of the data, but random chance. Learning such data points makes the model more flexible, at the risk of overfitting.

Task :

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).

Task 1 :

In Task 1 we are going to train the model by Linear Regression using the closed form solution. Our Linear Regression Function has the form $\mathbf{y(x, w) = w^T \Phi(x)}$ where $\mathbf{w = (w_0, w_1, ... w_{M-1})}$ is a weight vector to be learnt from training samples and $\mathbf{\Phi = (\Phi_0, ... \Phi_{M-1})^T}$ is a vector of M basis functions.

To find the output of the above linear regression function we have to find $\mathbf{\Phi j(x)}$ and calculate the noise model and objective function : $\mathbf{\Phi j(x) = exp(-1/2(x-\mu_j)^T \Sigma_j^{-1}(x - \mu_j))}$ where $\mathbf{\mu_j}$ is the center of the basis function and $\sum_j$ decides how broadly the basis function spreads.

Also regularization is an important factor to avoid overfitting, we do this by adding a regularization term to the error function such that $E(\mathbf{w}) = E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$ where weight decay regularizer is $E_W(\mathbf{w}) = ½\, \mathbf{w}^T \mathbf{w}$.

Now that we have all the terms we need to find the closed form solution, our closed form solution is of the form Moore-Penrose pseudo-inverse of the matrix Φ as $\mathbf{w}_{ML} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{t}$ where $\mathbf{t} = \{t_1, .., t_N\}$ is the vector of outputs in the training data and $\mathbf{\Phi}$ is the design matrix of $\mathbf{\Phi_i (x_N) X \Phi_{M-1} (x_N)}$. The closed-form solution with least-squared regularization is $\mathbf{w}* = (\lambda \mathbf{I} + \mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{t}$.

Task 2 :

In Task 2 we are going to train the model by stochastic gradient descent algorithm which uses random initial value $\mathbf{w}^{(0)}$. Then it updates the value of **w** using $\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} + \Delta \mathbf{w}^{\tau}$ where $\Delta\mathbf{w}^{\tau} = \eta^{\tau} E$ is called the weight updates. It goes along the opposite direction of the gradient of the error. η^{τ} is the learning rate, deciding how big each update step would be. It has linearity and that's why we have to calculate $\nabla E = \nabla E_D + \lambda \nabla E_W$ where $\nabla E_D = -(t_n - \mathbf{w}^{(\tau)\mathrm{T}} \boldsymbol{\varphi}(\mathbf{x}_n))\boldsymbol{\varphi}(\mathbf{x}_n)$ and $\nabla E_W = \mathbf{w}$

We evaluate our solution based on the RMS error value :

ERMS = $\sqrt{(2E(\boldsymbol{w}*)/\ NV)}$ where $\mathbf{w}^*$ is the solution and N_V is the size of the test dataset.

Key Concepts :

1) Training Dataset : The model is initially fit on a training dataset, that is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the mode. In practice, the training dataset often consist of pairs of an input vector and the corresponding answer vector or scalar, which is commonly denoted as the target. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

2) Testing Dataset : The testing dataset is used to provide an unbiased evaluation of a final model fit on the training dataset. A test dataset is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier.

3) .csv File : CSV is a simple file format used to store tabular data, such as a spreadsheet or database. Files in the CSV format can be imported to and exported from programs that store data in tables, such as Microsoft Excel or OpenOffice Calc. CSV stands for "comma-separated values". Its data fields are most often separated, or delimited, by a comma. For example, let's say you had a spreadsheet containing the following data.

Key Imports :

1) from sklearn.cluster import KMeans - import clustering from sklearn.cluster using KMeans

2) import numpy as np - import numpy library as np

3) import csv - import csv import and export format for spreadsheets and databases

4) import math - import math library

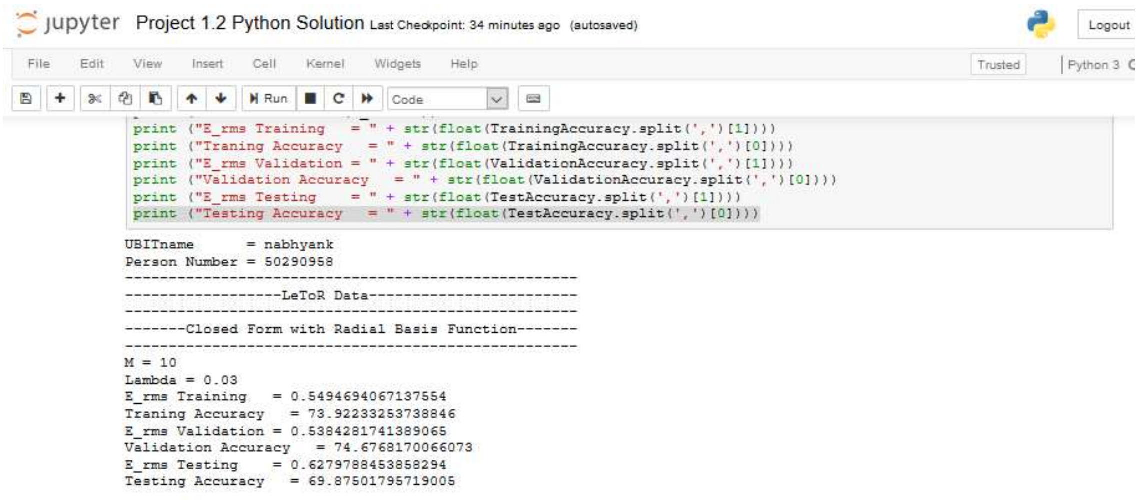
5) import matplotlib.pyplot - import pythons math plot library

6) from matplotlib import pyplot as plt - import plot from math plot library

Hyper - Parameter Changes :

Parameter Set 1 :

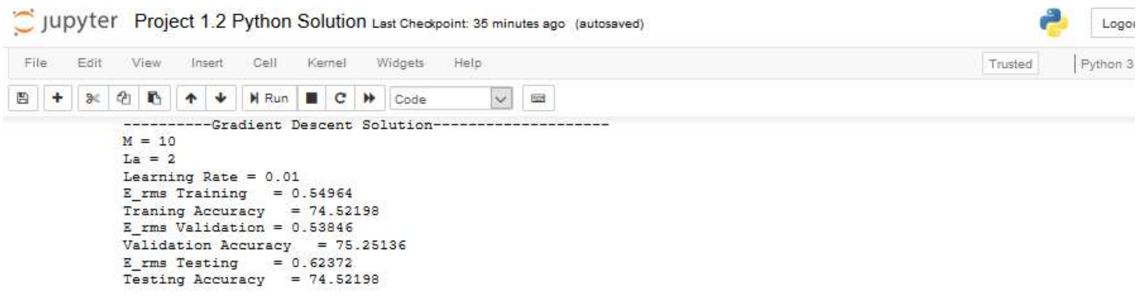
a) For Closed form : M= 10, Lambda = 0.03


jupyter
Project 1.2 Python Solution Last Checkpoint: 34 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

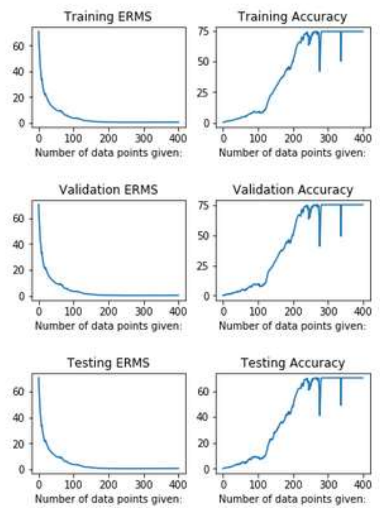
```
print ("E_rms Training   = " + str(float(TrainingAccuracy.split(',')[1])))
print ("Traning Accuracy   = " + str(float(TrainingAccuracy.split(',')[0])))
print ("E_rms Validation = " + str(float(ValidationAccuracy.split(',')[1])))
print ("Validation Accuracy   = " + str(float(ValidationAccuracy.split(',')[0])))
print ("E_rms Testing    = " + str(float(TestAccuracy.split(',')[1])))
print ("Testing Accuracy   = " + str(float(TestAccuracy.split(',')[0])))
```

```
UBITname      = nabhyank
Person Number = 50290958
----------------------------------------------------
------------------LeToR Data------------------------
----------------------------------------------------
-------Closed Form with Radial Basis Function-------
----------------------------------------------------
M = 10
Lambda = 0.03
E_rms Training   = 0.5494694067137554
Traning Accuracy   = 73.92233253738846
E_rms Validation = 0.5384281741389065
Validation Accuracy   = 74.6768170066073
E_rms Testing    = 0.6279788453858294
Testing Accuracy   = 69.87501795719005
```

b) For Gradient Descent : M= 10, La= 2, Learning Rate = 0.01

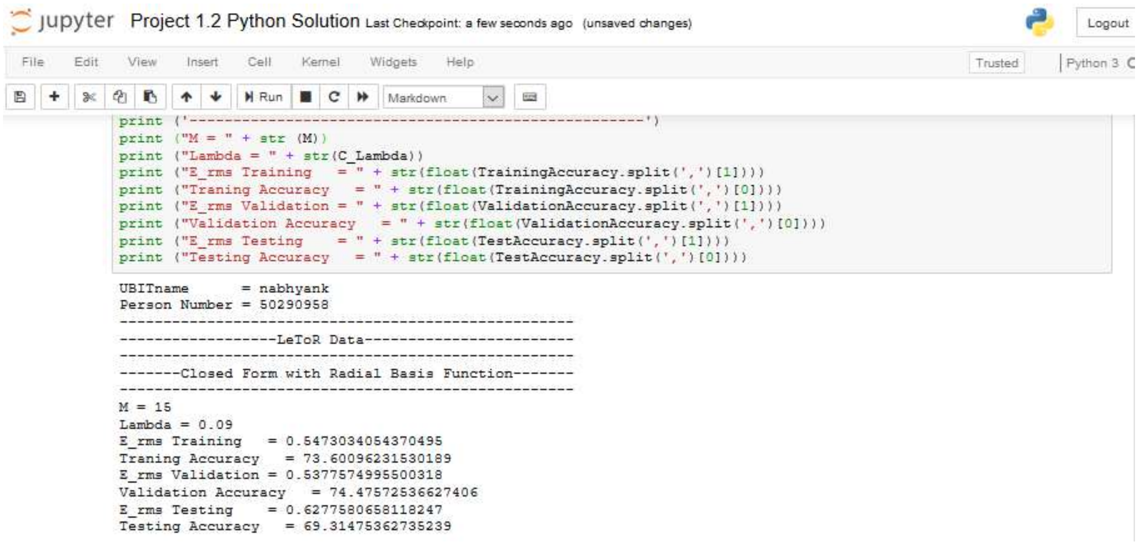

jupyter
Project 1.2 Python Solution Last Checkpoint: 35 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
----------Gradient Descent Solution--------------------
M = 10
La = 2
Learning Rate = 0.01
E_rms Training   = 0.54964
Traning Accuracy   = 74.52198
E_rms Validation = 0.53846
Validation Accuracy   = 75.25136
E_rms Testing    = 0.62372
Testing Accuracy   = 74.52198
```



Parameter Set 2 :

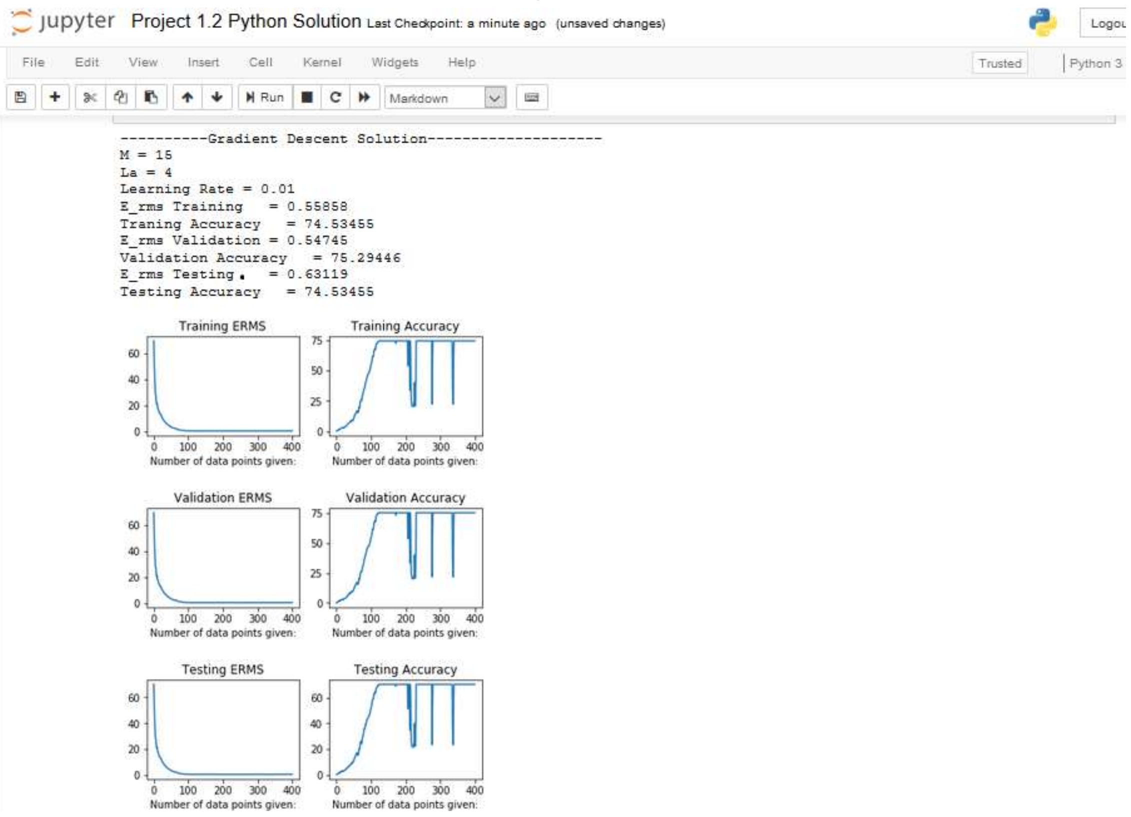
a) For Closed form : M= 15, Lambda = 0.09


jupyter Project 1.2 Python Solution Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

```
print ('---------------------------------------------------')
print ("M = " + str (M))
print ("Lambda = " + str(C_Lambda))
print ("E_rms Training   = " + str(float(TrainingAccuracy.split(',')[1])))
print ("Traning Accuracy   = " + str(float(TrainingAccuracy.split(',')[0])))
print ("E_rms Validation = " + str(float(ValidationAccuracy.split(',')[1])))
print ("Validation Accuracy   = " + str(float(ValidationAccuracy.split(',')[0])))
print ("E_rms Testing    = " + str(float(TestAccuracy.split(',')[1])))
print ("Testing Accuracy   = " + str(float(TestAccuracy.split(',')[0])))
```

```
UBITname      = nabhyank
Person Number = 50290958
----------------------------------------------------
------------------LeToR Data------------------------
----------------------------------------------------
-------Closed Form with Radial Basis Function-------
----------------------------------------------------
M = 15
Lambda = 0.09
E_rms Training   = 0.5473034054370495
Traning Accuracy   = 73.60096231530189
E_rms Validation = 0.5377574995500318
Validation Accuracy   = 74.47572536627406
E_rms Testing    = 0.6277580658118247
Testing Accuracy   = 69.31475362735239
```

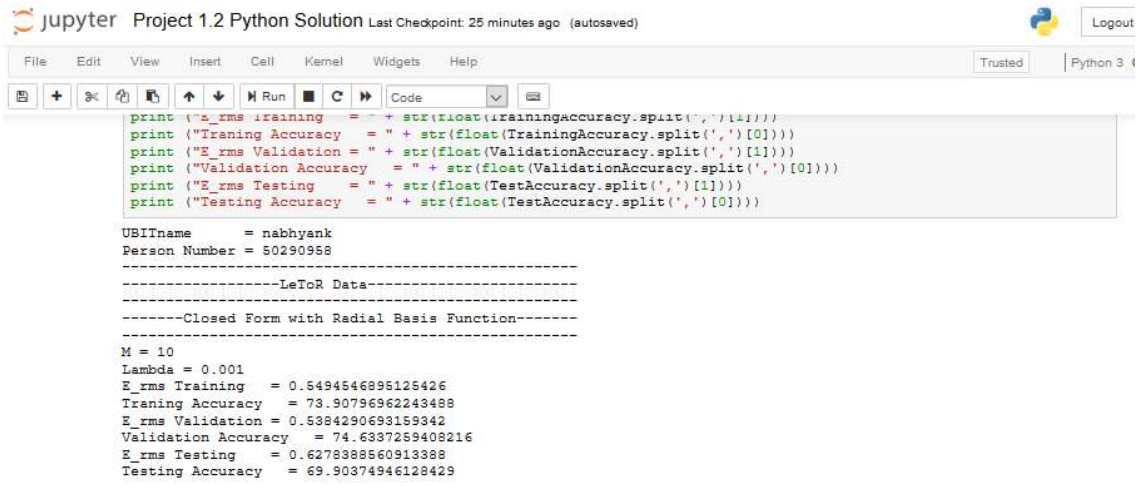
b) For Gradient Descent : M= 15, La= 4, Learning Rate = 0.04


jupyter Project 1.2 Python Solution Last Checkpoint: a minute ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
Training ERMS
Training Accuracy
Validation ERMS
Validation Accuracy
Testing ERMS
Testing Accuracy
Number of data points given:

```
----------Gradient Descent Solution--------------------
M = 15
La = 4
Learning Rate = 0.01
E_rms Training   = 0.55858
Traning Accuracy   = 74.53455
E_rms Validation = 0.54745
Validation Accuracy   = 75.29446
E_rms Testing.   = 0.63119
Testing Accuracy   = 74.53455
```

Parameter Set 3 :

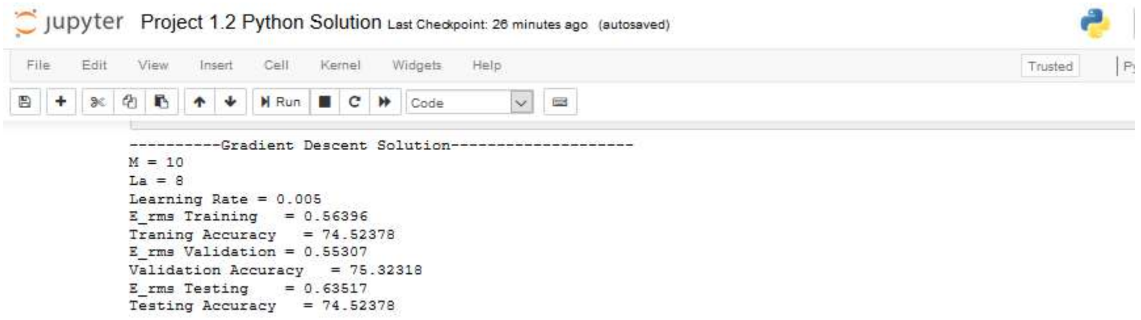
a) For Closed form : M= 10, Lambda = 0.001


Jupyter Project 1.2 Python Solution Last Checkpoint: 25 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

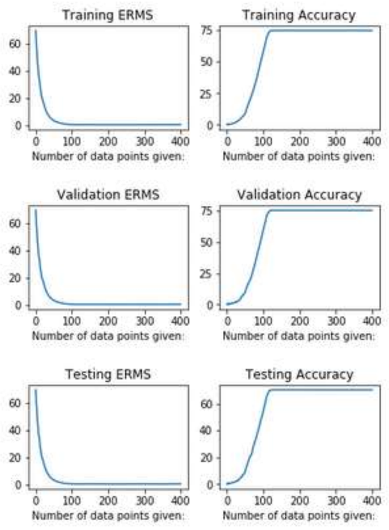
```
print ("E_rms Training   = " + str(float(TrainingAccuracy.split(',')[1])))
print ("Traning Accuracy   = " + str(float(TrainingAccuracy.split(',')[0])))
print ("E_rms Validation = " + str(float(ValidationAccuracy.split(',')[1])))
print ("Validation Accuracy    = " + str(float(ValidationAccuracy.split(',')[0])))
print ("E_rms Testing     = " + str(float(TestAccuracy.split(',')[1])))
print ("Testing Accuracy   = " + str(float(TestAccuracy.split(',')[0])))
```

```
UBITname      = nabhyank
Person Number = 50290958
----------------------------------------------------
------------------LeToR Data------------------------
----------------------------------------------------
-------Closed Form with Radial Basis Function-------
----------------------------------------------------
M = 10
Lambda = 0.001
E_rms Training   = 0.5494546895125426
Traning Accuracy   = 73.90796962243488
E_rms Validation = 0.5384290693159342
Validation Accuracy    = 74.6337259408216
E_rms Testing     = 0.6278388560913388
Testing Accuracy   = 69.90374946128429
```

b) For Gradient Descent : M= 10, La= 8, Learning Rate = 0.005


Jupyter Project 1.2 Python Solution Last Checkpoint: 26 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code

```
----------Gradient Descent Solution--------------------
M = 10
La = 8
Learning Rate = 0.005
E_rms Training   = 0.56396
Traning Accuracy   = 74.52378
E_rms Validation = 0.55307
Validation Accuracy    = 75.32318
E_rms Testing     = 0.63517
Testing Accuracy   = 74.52378
```



Graph Analysis :

Although the hyper parameters were changed there was no significant change in the shape or the accuracy of the graphs. The Accuracy was between 69 – 74 % for closed form and 74 - 75.40 % for the gradient descent form. The current regression model is running on 400 iterations and the dataset consists of approximately 69623 data points. The dataset is mostly populated more with 0's and there are lesser values of 1's or 2's thus the model gets overfit for 0's and messes with accuracy. Thus it is seen that changes in the hyper parameters does not change the graph plots or the Accuracy and ERMS values.

Conclusion :

It is observed that there are not many changes in the output ERMS or Accuracy of the regression based learning model even after changes in hyper-parameters were made. This is due to the regression model overfitting for 0's as the dataset contains more 0's than 1's and 2's. Also the graph plots of the model for different parameter changes show that there are no significant changes in the output graphs.

References :

https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/

https://pythonmachinelearning.pro/using-neural-networks-for-regression-radial-basis-function-networks/

https://machinelearningmastery.com/linear-regression-for-machine-learning/

https://en.wikipedia.org/wiki/Stochastic_gradient_descent

https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

https://en.wikipedia.org/wiki/Training,_test,_and_validation_sets

https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/