CSE 474/574: Introduction to Machine Learning (Fall 2016)
Project 2: Learning to Rank using Linear Regression

UBitName: manishat

Person Number: 50207628

Tasks involved in the project:

- Train a linear regression model on LeToR dataset using a closed-form solution.
- Train a linear regression model on the LeToR dataset using stochastic gradient descent (SGD).
- Train a linear regression model on a synthetic dataset using a closed-form solution.
- Train a linear regression model on the synthetic dataset using SGD.

Data set #1: The LeToR Dataset

- The letor training data consists of a pair of input values x and target values t.
- LeToR is the short of Learning to Rank, here the dataset has been downloaded from MQ2007.
- Details about the data:
- The first column is the relevance label of the row. It takes one of the discrete values 0, 1 or 2. The larger the relevance label, the better is the match between query and document. Note that objective output y of our linear regression will give a continuous value rather than a discrete one– so as to give a fine-grained distinction.
- The following 46 columns are the features. They are the 46-dimensional input vector x for our linear regression model. All the features are normalized to fall in the interval of [0, 1].

Data set #2: Synthetic Dataset

- Each data point consists of 10 features.
- All the target values of the training data are from a set discrete {0, 1, 2}. The objective of the regression algorithm is to map the input to a real value in the interval [0, 2].
- The target values are discrete. But the regression algorithm learns a continuous value.
- In the solution we assume the output to be Gaussian distributed–which is a continuous distribution. Thus the learning algorithm will learn a continuous value; the mean of the output is going to be around 1 if outputs of 0,1, and 2 are equally likely.

- While most of the outputs will be in [0, 2], some outputs will lie outside of [0, 2] and some outputs could even be negative

<u>Extracting feature values and labels from the data:</u>
The data has been extracted using genfromtxt using numpy module.
<u>Data partition:</u>
I partitioned the data into **training, validation and testing set** depending upon the requirement to be 80%,10% and 10% respectively.
I specified the number of docs that have to be respectively into the training,validation and testing set and the appropriate division took place.

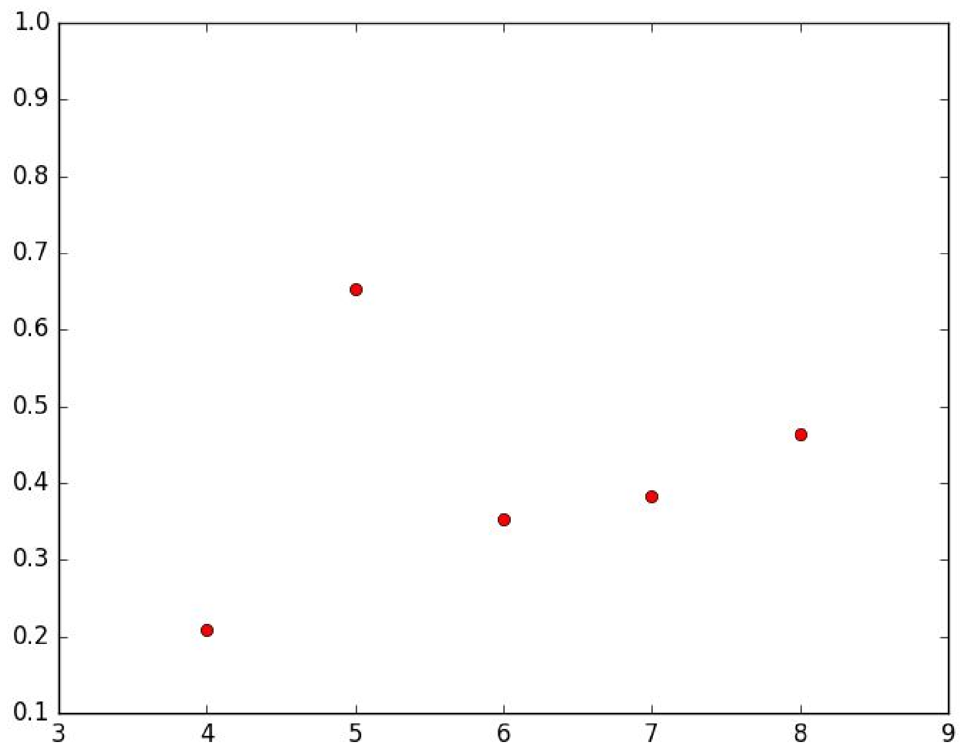
- As a part of the first step, I imported numpy, re and random from python.
- **<u>Closed-form solution:</u>**
- **<u>Train Model Parameter:</u>**
- Correspondingly, Y and X matrices for training, validation and testing were made such that the relevance values correspond to Y matrix and the feature vectors point to the X matrix.
- The next major task was to generate means and calculate variance matrix.
- For generating the variance matrix, we had to consider each column of X as the column matrix and fill in the corresponding values of index [i][i] in the variance matrix.
- The weight can be found out using the formula given, $w_{ML} = (\Phi^{\top}\Phi)^{-1}\Phi^{\top}t$.
- Next major task is to fill in the Φ matrix.
- The regularized weight can be found out from the formula $w^* = (\lambda I + \Phi^{\top}\Phi)^{-1}\Phi^{\top}t$. Where λ is a random uniform number generated using randint.
- **<u>Tune Hyper Parameters:</u>**
- While tuning Hyper Parameters, I observed the changes for different values of M ranging from 3 to 8, the values showed the best fit on an average for value of M = 4.
- **<u>Testing machine learning scheme on the testing set:</u>**
- The value of Hyper parameters are :
- M = 4
- **'W' array:**
- ([[0.67772272],
  [2.41152502],
  [-0.98007709],
  [-1.88241263]]))
- **'WR' array:**
- ([[0.68224029],
- [2.20309575],
- [-0.98878531],
- [-1.67582092]]))

- **<u>'phi', array:</u>**
  ```
  ([[ 1.     , 0.77564431, 0.8624078 , 0.81841702],
  [ 1.       , 0.70138374, 0.72644175, 0.7186349 ],
  [ 1.       , 0.85348372, 0.79595823, 0.89364021],
  ...,
  [ 1.       , 0.63685177, 0.64015872, 0.66208188],
  [ 1.       , 0.73106398, 0.86979932, 0.75318124],
  [ 1.       , 0.83834519, 0.81507891, 0.88651583]]))
  ```
- For stochastic gradient, we try to find ∇E using the formulae, $\nabla E = \nabla ED + \lambda \nabla EW$
- The corresponding values printed were:
- ('h2', array([15575.48583931]))
- ('EDW', 7787.742919656318)
- ('ERMS', 0.55813055532397859)
- ('EDW', 7788.092316636833)
- ('Reg ERMS', 0.55814307544345054)
- ('ERMS2n',ERMS2i', 0.70317915588594837, 0.99412946634896737)
- ('ERMS2nr',ERMS2ir', 2.8776368517357951, 2.8903012074086951)

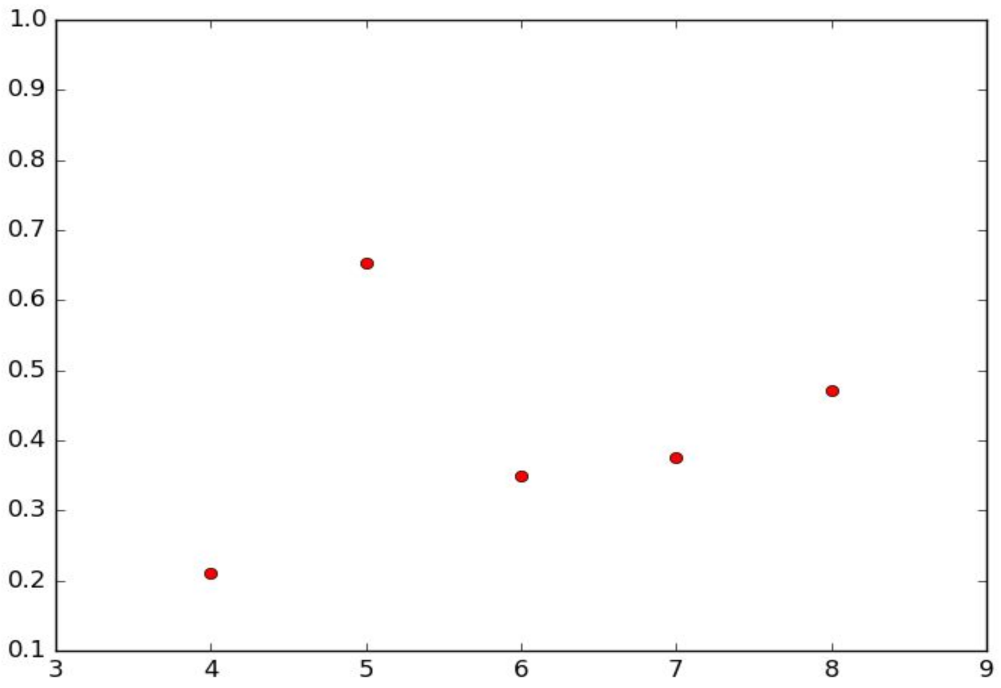
- The corresponding values being printed as a result of the training for **Microsoft Dataset** are:
- ('ERMS Closed form training set: ', 0.5541166558152979)
- ('Regularized ERMS Closed form training set: ', 0.55411665630252105)
- ('ERMS Closed form validation set: ', 0.22889947730293736)
- ('Regularized ERMS Closed form validation set: ', 0.22887267462245886)
- ('ERMS Closed form testing set: ', 0.22841662665775514)
- ('Regularized ERMS Closed form testing set: ', 0.22838955914962508)
- ('ERMS Stochastic form training set: ', 0.76661944553317707)
- ('Regularized ERMS Stochastic from training set: ', 0.76682009825754371)
- ('ERMS Stochastic form validation set: ', 0.81340418202375309)
- ('Regularized ERMS Stochastic form validation set: ', 0.81374695183586987)
- ('ERMS Stochastic form testing set: ', 0.98911947567175473)
- ('Regularized ERMS Stochastic form testing set: ', 0.9894729809915046)

- The corresponding values being printed as a result of the training for **Synthetic Dataset** are:
- ('ERMS Closed form training set: ', 0.55444034317820634)
- ('Regularized ERMS Closed form training set: ', 0.55444034335884818)
- ('ERMS Closed form validation set: ', 0.42224080636966449)
- ('Regularized ERMS Closed form validation set: ', 0.42223084833879376)
- ('ERMS Closed form testing set: ', 0.25217961916465925)
- ('Regularized ERMS Closed form testing set: ', 0.25218672592521468)
- ('ERMS Stochastic form training set: ', 0.76637385990335238)
- ('Regularized ERMS Stochastic from training set: ', 0.76656636017669577)
- ('ERMS Stochastic form validation set: ', 0.98773561926166864)
- ('Regularized ERMS Stochastic form validation set: ', 0.98807998392655694)
- ('ERMS Stochastic form testing set: ', 0.98917216172682554)
- ('Regularized ERMS Stochastic form testing set: ', 0.98951691096724348)

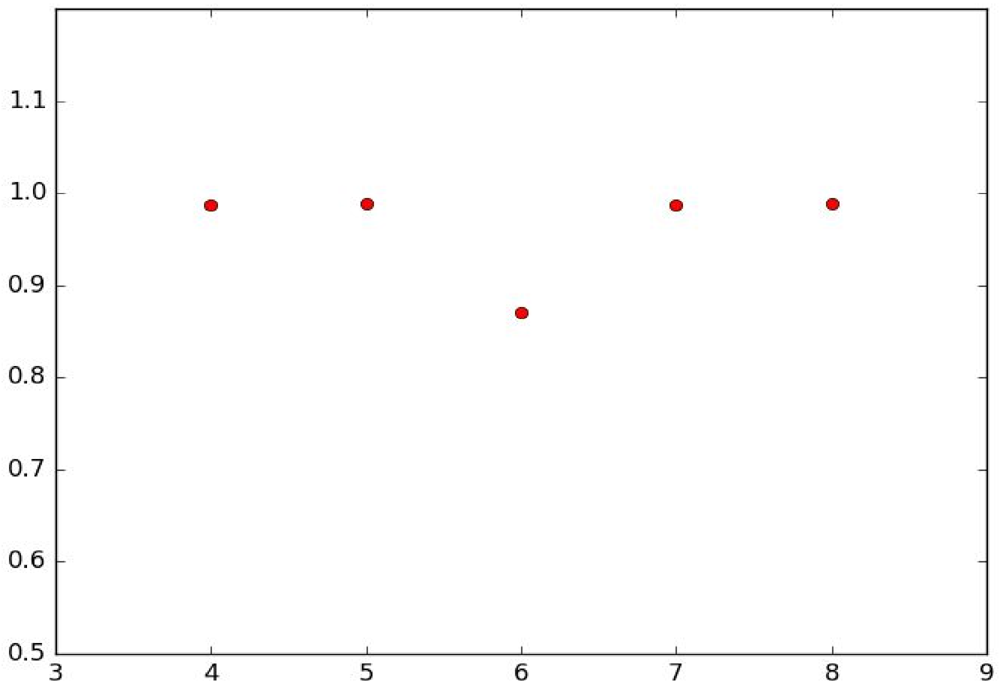
The corresponding plots observed are:



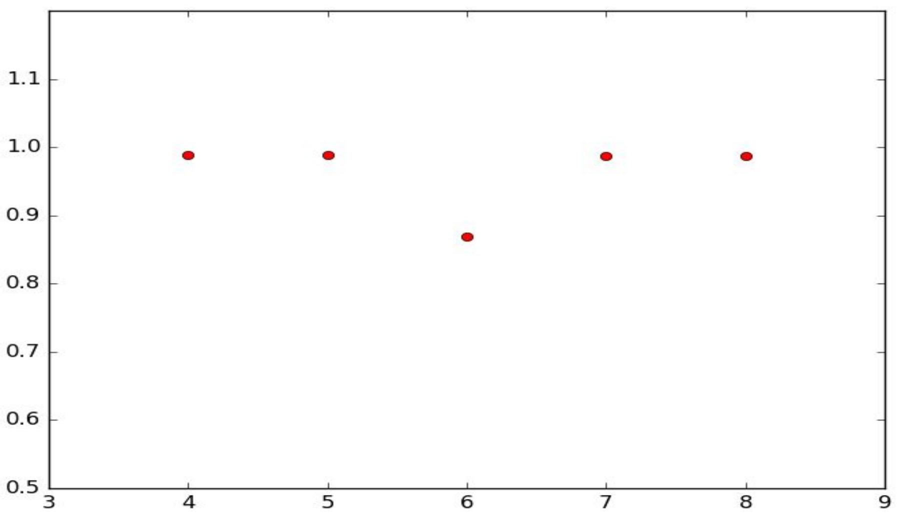
1. M vs Erms closed form validation set



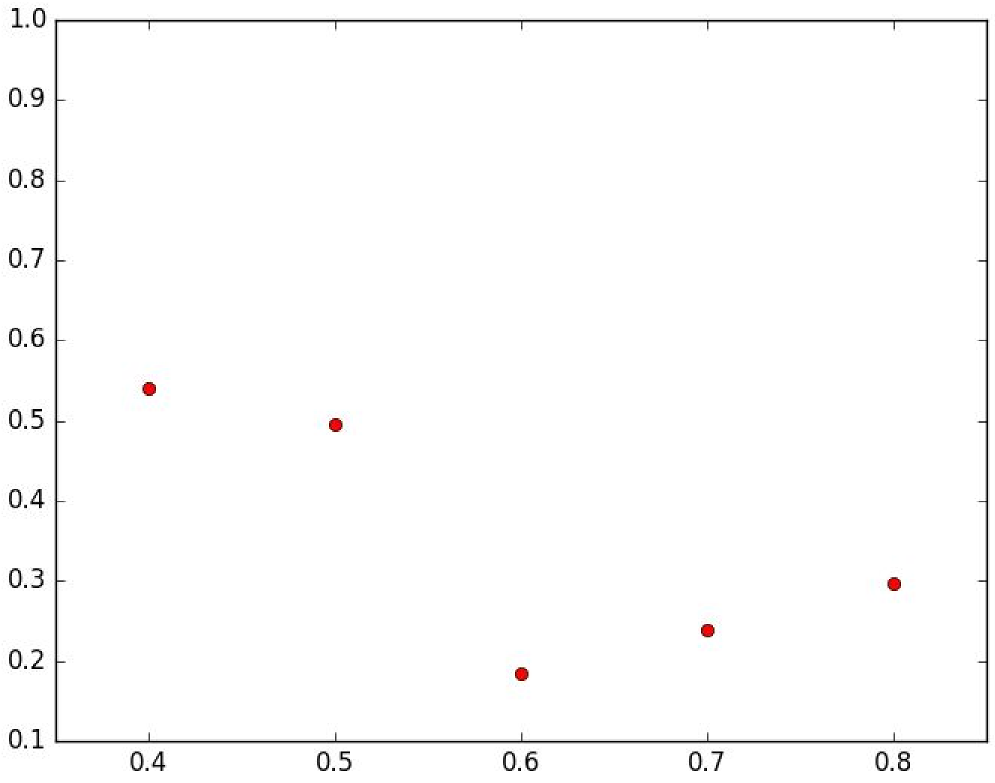
2. M vs Reg. Erms closed form validation set



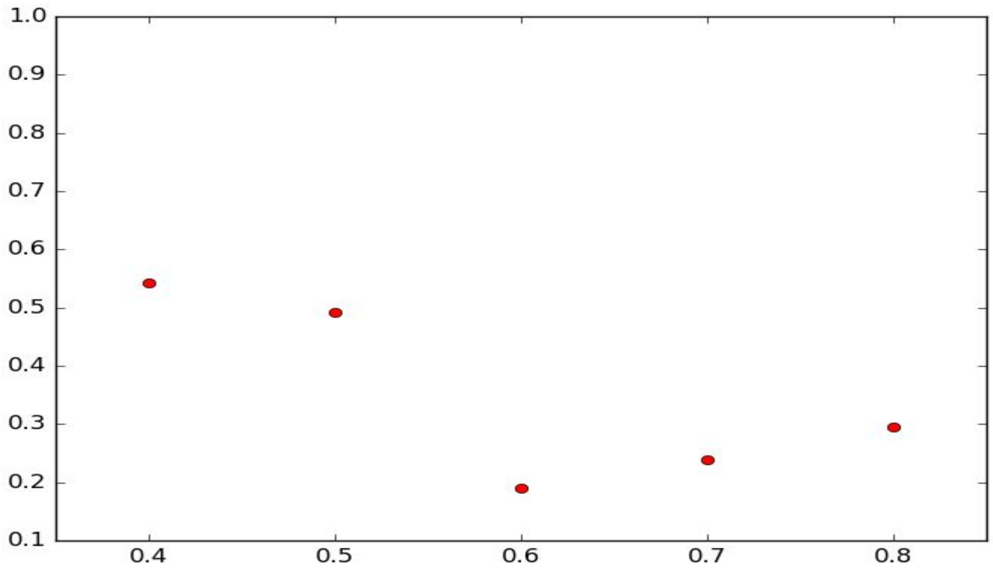
3 . M vs Erms Stochastic closed form validation set



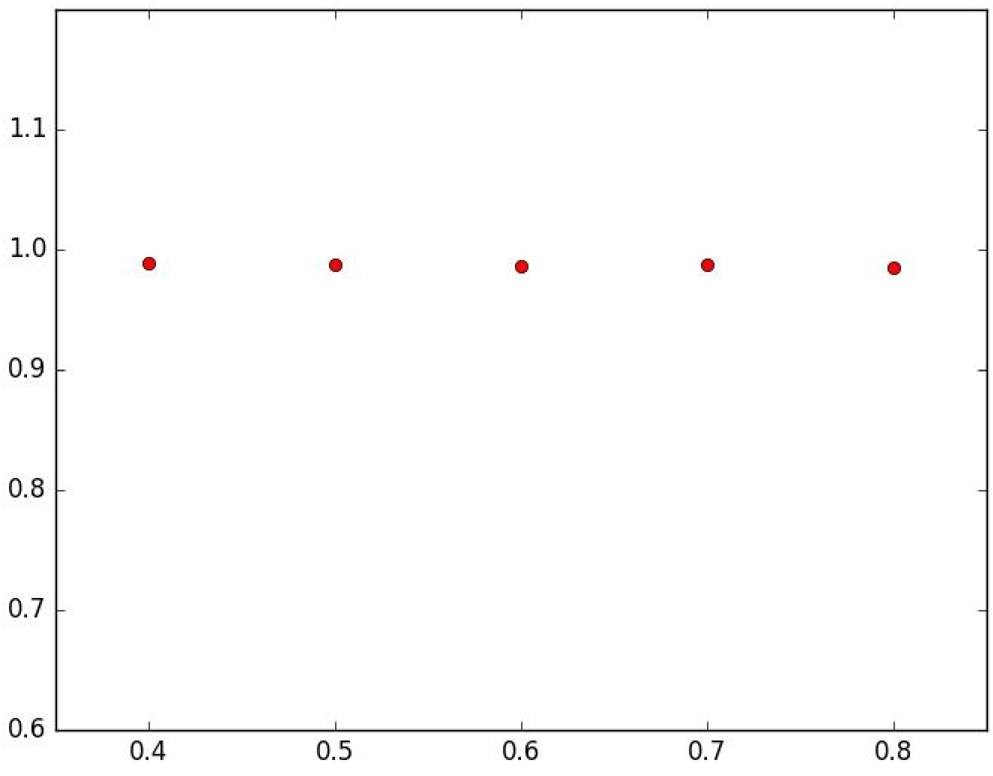
4. M vs Reg. Erms stochastic closed form validation set



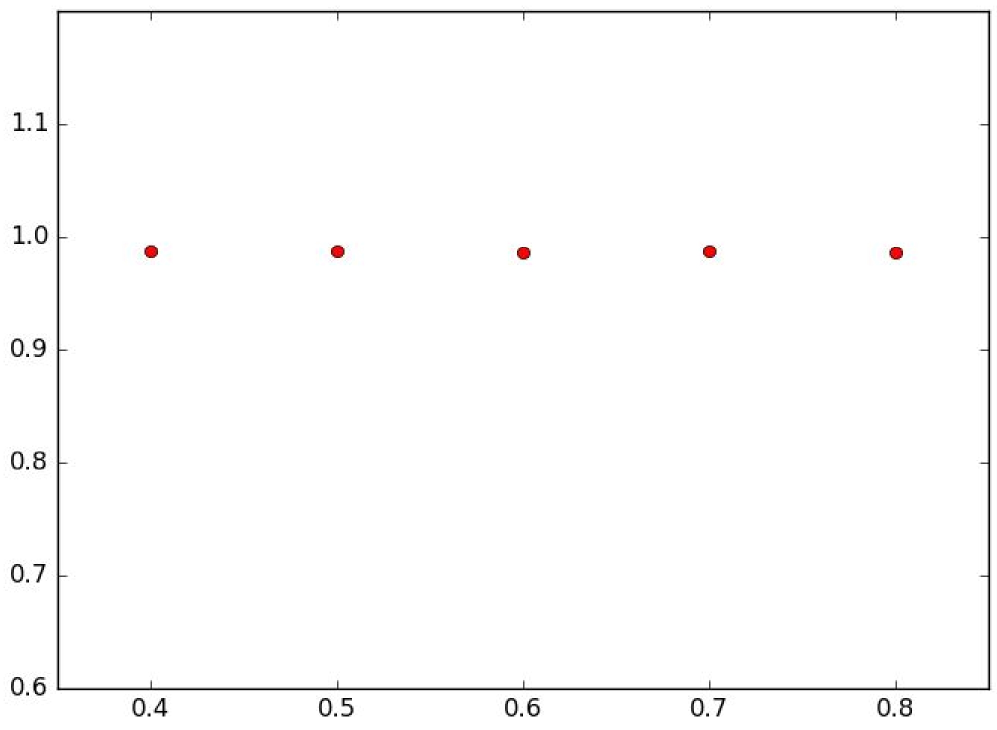
5. ETA vs Erms closed form validation set



6. ETA vs Reg. Erms using closed form validation set



7. ETA vs Erms Stochastic closed form validation set



8. ETA vs Reg. Erms stochastic closed form validation set

Hence the hyper tune parameters were observed to be the best of the available inorder to provide the best machine learning capacity for the algorithm.