ML 574 Project 1.2

Daniel Amirtharaj

`damirtha@buffalo.edu`
UB Person Number 50291137

1 Projrect overview

This project uses Linear Regression to solve the *LeToR* (Learning to Rank) Problem. Two approaches have been used to compare their respective advantages/disadvantages, i.e the closed form solution and the gradient descent algorithm.

Objective

The objective here is to implement both closed form solution and the gradient descent algorithm for the *LeTor* program using Python. The following are the tasks involved.

1. Pre-Process the LeToR dataset, and extract data labels and relevant features from the input data set.
2. Split the raw data obtained into 3 sets, the training, validation and test sets.
3. Generate cluster centers in the training data, and calculate values of mean (Centroid) and variance.
4. Using these values, generate basis functions for the training input samples.
5. Using the basis functions generated, use the closed form solution to find weights to fit the linear regression model.
6. Use the gradient descent algorithm, as an alternative to the closed form solution to find weights to fit the linear regression model.
7. Evaluate ERMS and accuracy of this model using the predicted labels and the given labels for each data sample.
8. Change parameters such as learning rate, number of basis functions and regularization coefficient and observe changes in performance, and finally pick model setting with best performance.
9. Using the learned parameters, generate an output file giving the correct classification for any range of inputs.

2 Analysis

Since most of the key concepts used in the project have already been described in detail in the project description, these are gone through briefly here.

2.1 The *LeToR* problem

LeToR is an information retrieval problem that heavily relies on machine learning to effectively rank search results based on their order of relevance. Here it is assumed that the order of relevance of the sample data is already available.

2.1.1 Radial Basis functions

Radial basis functions, add non-linearity to the dataset by transforming the input features into another set of transformed features that are linearly related to the target vector. This is extremely useful while modeling non-linear functions using linear regression. They are computed, by taking the gaussian radial basis function of each data sample centered around a point (centroid of cluster) in the f dimensional feature space of the input dataset.

2.1.2 k-means clustering

k-means is an advanced clustering algorithm that employs unsupervised learning. Here points in the f dimensional feature space are randomly chosen and are grouped into a fixed number of clusters M, each with a centroid such that a data sample in a cluster is closest to the centroid of that cluster. This is useful when similar data has to be grouped together based on certain parameters.

2.1.3 Regularization

Regularization is a tool used to avoid overfitting in machine learning algorithms. Since the training data can be tuned and adjusted to give the maximum performance on a dataset, its performance on the dataset is not reliable, and does not represent how the algorithm will behave with unseen data. In order to prevent the model from giving great training performances but poor ability to generalize, regularization is used. It ensures that the weights learned do not assume disproportionate or abnormally large values, preventing overfitting.

2.1.4 Closed form solution

The closed form solution is used to solve machine learning problems using mathematical equations, and in particular linear algebra. It works by working out unknown parameters using known values and parameters, similar to simultaneous equations.

2.1.5 Gradient descent algorithm

The gradient descent algorithm is used as an alternative solution to solve machine learning problems, and has several advantages over the closed form solution. Since the closed form solution involves taking the inverse of a matrix, it may not always be feasible to use it when solving for large datasets. Gradient descent also works well when modeling non-linear datasets whose cost functions may not be convex and have a single point of convergence.

2.1.6 Training/Validation/Test sets

The dataset is split into 3 sets, the training, validation and the test sets, usually in the ratio 8:1:1. The training dataset is used solely to train the model, while the validation dataset is used to tune hyper-parameters on the model, and finally the test set is used to measure the model's performance, and this gives a good idea on how well the model has learned, and its ability to generalize over unseen data samples.

3 Methodology

3.1 Data Preprocessing

The LeToR dataset was presented in a .txt file and required a bit of pre-processing before it could be used to train any model. Columns from indices 2 to 48 represent the features of the input dataset and were read into the program as the input data. Column 1 in the file represents the data labels and these was read into the program as the data labels/ target labels. The dataset was then split into 3 sets, the training, validation and test sets in the ratio of 8:1:1.

3.2 Generating Basis functions

Using the input features from the training set, M basis functions were generated in the following steps,

1. M cluster centroids were generated either by picking M random samples from the training dataset, or by applying k-means clustering on the f-dimensional feature space of the training set.
2. The variances corresponding to each feature dimension for sample data within a cluster were computed, this could be taken to be unique to each cluster or the uniform spread of the entire input data.
3. The basis for each data sample was computed for each cluster centroid and variance using the gaussian radial function, resulting in a design matrix 'phi' which was then used as the input to the linear regression algorithm.

3.3 Regression with closed form solution

Linear regression to solve this problem was modeled using the closed form solution first. The mathematical equations used to work out the variables to be found (the weights in the regression model in this case) were written down and computed using the 'numpy' library in python. The code submitted, gives a detailed description on the steps used here.

3.4 Regression with Gradient descent algorithm

The equations that establish gradient descent were laid out in python with the help of the 'numpy' library. Gradient descent is implemented using fixed steps that move the weights learned along the gradient of the cost function, and this is laid out as a program in python.

3.5 Validation of the model

The model was then validated using different values of M (number of basis functions), lambda (regularization parameter) and learning rate, to see which hyper-parameter gave the best solution and was able to obtain a better model with better performance.

3.6 Testing

Now that the model has been trained with the input dataset, its performance on a dataset it has not encountered yet will be able to give a good idea of how good the model is at generalizing and predicting outputs of new unseen samples.

4 Results

4.1 Closed form Solution

The model was trained using the closed form solution implemented with linear algebra. Different configurations were chosen to be tested and their results were evaluated. The following sections below highlight the findings.

4.1.1 Random cluster centroids for Mu

When using a random cluster centroid for Mu, along with a uniform spread for all basis functions, the following results were obtained for varying values of M (number of basis functions), and lambda (regularization parameter). Here performance measures are given in RMS (Root mean squared) error.

It can be observed here that as M increases, the value of ERMS decreases for a constant value of lambda, until M=200, and it starts to increase after that. This gives an idea of a good spot where linear regression is able to reach its maximum performance. As the number of features is very small or too large for a constant sized dataset, the ability of the model to maintain linearity will decrease, and this result is a consequence of this. It is also evident that the ERMS values change little for M between 50 and 200, and a final M can be chosen using a trade off between accuracy and performance (as M increases the program become slower).

As the value of lambda increases, ERMS increases as well. It can be seen that regularization hardly makes any difference, and that the model is probably better off without it. This might be due to both

M	ERMS Training	ERMS Validation
1	0.565	0.555
5	0.555	0.544
10	0.548	0.541
50	0.539	0.536
100	0.537	0.534
200	0.535	0.532
400	0.536	0.534

Table 1. Performance measures (ERMS) on the linear regression model using closed form solution, for lambda =0.01 and varying values of M.

lambda	ERMS Training	ERMS Validation
0	0.531	0.530
0.01	0.535	0.532
0.03	0.536	0.533
0.1	0.538	0.534
0.3	0.539	0.534
1	0.540	0.535
10	0.543	0.537

Table 2. Performance measures (ERMS) on the linear regression model using closed form solution, for M=200 and varying values of lambda.

the distribution of the data and the large number of data samples used. When large data samples are used, the algorithm naturally avoids overfitting due to the amount of information available.

4.1.2 Cluster centroids for Mu, using k-means clustering

When using the k-means clustering algorithm to find the cluster centroids for Mu for a fixed number of centroids M (number of basis functions), along with a uniform spread for all basis functions, the following results were obtained for varying values of M and lambda (regularization parameter).

M	ERMS Training	ERMS Validation
1	0.565	0.554
5	0.553	0.541
10	0.550	0.539
50	0.540	0.537
100	0.537	0.535
200	0.536	0.534
400	0.535	0.533

Table 3. Performance measures (ERMS) on the linear regression model using closed form solution with k-means, for lambda=0.01 and varying values of M.

The performance of the model keeps decreasing with the value of M, with higher values of M, lowering the ERMS of the validation set. The improvement in performance is minimal when compared to the computational cost overhead required to maintain and work with matrices with large number of features (basis functions in this case), and hence a value of M between 50 and 200 can be chosen and set for this model. Similar to the previous result for closed form without k-means, it can be observed that regularization hardly makes a difference. Again, this can be attributed to the large number of data samples available (n»M).

It can also be observed that there is not much difference between random clustering and the usage of k-means. This is can be attributed to the fact that uniform spread is assumed across the dataset in both cases, and hence there is no notable difference.
The configuration that gives the best performance here is, M=200 and lambda=0. This gives an ERMS of 0.530 on the validation set and 0.614 on the test set.

lambda	ERMS Training	ERMS Validation
0	0.532	0.531
0.01	0.536	0.533
0.03	0.538	0.535
0.1	0.539	0.535
0.3	0.540	0.536
1	0.541	0.536
10	0.544	0.537

Table 4. Performance measures (ERMS) on the linear regression model using closed form solution with k-means, for M=200 and varying values of lambda.

4.2 Gradient descent algorithm

When the model was re-trained using the gradient descent algorithm, different configurations were chosen and were tested and their results were evaluated. The following sections below highlight the findings. The radial basis here were generated with centroids chosen using k-means.

M	ERMS Training	ERMS Validation
1	0.599	0.586
5	0.653	0.643
10	0.655	0.642
50	0.651	0.649
100	0.661	0.664
200	0.770	0.768
400	2.292	2.293

Table 5. Performance measures (ERMS) on the linear regression model using gradient descent with k-means, for lambda=1, learning rate=0.01 and varying values of M.

For a constant value of lambda and learning rate, it can be seen that increasing the value of M has no change or effect on the performance of the model. This can be explained by the fact that only 400 samples were used to train the model using gradient descent. When the number of clusters increase, cluster centroids far away from the first 400 samples may be chosen which may result in the information spread across data samples, and hence more samples would be required for the model to perform better.

lambda	ERMS Training	ERMS Validation
0	2.39	2.40
0.01	2.29	2.31
0.03	2.12	2.12
0.1	1.59	1.6
0.3	0.789	0.789
1	0.599	0.586
10	0.643	0.628

Table 6. Performance measures (ERMS) on the linear regression model using gradient descent with k-means, for M=1, learning rate=0.01 and varying values of lambda.

It can be observed clearly from the above table that when lambda increases, the model converges to a minima better, and then increases again for large values of lambda.

The configuration that gives the best performance here is, M=1 and lambda=1. This gives an ERMS of 0.586 on the validation set and 0.692 on the test set. The accuracy of this model is 70.23%.

5 Conclusion

Training the model using the closed form solution and the gradient descent algorithm has highlighted how they perform differently and why gradient descent is preferred over the closed form solution. The gradient descent algorithm was able to perform nearly as good as the closed form solution with just 400 data samples, reducing the computational expense greatly, while having accuracy comparable to the closed form solution.

References

[1] Numpy documentation
`https://docs.scipy.org/doc/numpy/reference/`

[2] Code posted in UB learns.

[3] Gradient Descent Algorithms, by Sebastian Ruder.
`http://ruder.io/optimizing-gradient-descent/`