Project 1.2 Implementation

Amit Menon
Department of Computer Science
State University of New York at Buffalo, NY
amitanil@buffalo.edu

Abstract

The aim of this project is to provide a solution to the Learning to Rank problem (LETOR) using 2 approaches.

1. Using closed form solution for linear regression on the LETOR dataset
2. Using stochastic gradient descent (SGD)

1 Data Analysis and preprocessing.

We read input and target output from two separate files Querylevelnorm_x.csv and Querylevelnorm_t.csv respectively. The first step in the analysis is to check that the data obtained is in what format and range. All the 46 input features in the LETOR dataset are normalized, so we need not perform any normalization from our end. Function "data_inp.cov()" gives a covariance matrix of all 46 input features with each other. This showed us that there were some columns for which the covariance could not be calculated. We remove these columns from our input data set.

We need to form our training, testing and validation data. The data is partitioned as per the percentages specified i.e. 80% 10% 10% respectively. I first split the raw data into 2 one for training and the rest for testing and validation which I split manually into two parts again.

1.2 Big Sigma Inverse and Clusters

We use Gaussian radial basis function to as an input to the model instead of full input vector.

As stated above the covariance matrix is obtained using the .cov function, we need the variance of features with itself as we calculate Gaussian basis function in a vectored form. The following lines of code achieves that.

```
BigSigma = data_inp.cov()
BigSigma = np.diag(np.diag(BigSigma))
BigSigma_inv = np.linalg.inv(BigSigma)
```

Gaussian Basis functions works on the distance of points from the center point (centroid) of the data set. To get a set number of centroids on which we need to apply the Gaussian basis function, K means clustering algorithm is used. The parameter "n_clusters" thus becomes a hyper-parameter for the algorithm.

1.3 Need for Gaussian Basis Function and Addressing overfitting

A. Gaussian Basis Function:

We have 41 input vectors after data preprocessing. When number of features increase, the

probability of it fitting a linear model decreases greatly and thus makes it more prone to under-fit. Conversely if we use a high degree hypothesis, chances of over fitting increases. Since we want to use linear regression, a Gaussian Basis function which provides us a better fit to the data to be able to use linear regression.

Using Gaussian basis function gives us following advantages:

1. We control the complexity (linearity) of the model by letting us choose number of clusters (or basis functions) which decides the dimensionality of the model.
2. It helps us convey all the information of the 41 input vectors into a reduced matrix

B. Regularization.

In order to address overfitting, we use a regularization parameter "reg_lamda" for closed form one and "lamda" in the gradient descent one. These parameters act as a trade of between fitting the data well and keeping weights small, because smaller weights results in simpler hypothesis which are less prone to over fit.

Increasing lamda values decreases the values of weights.

3 Tuning Parameters for Closed form

Closed form solution is an easier solution to compute as it doesn't involve iterating over the whole data. We construct a design matrix using a Gaussian basis function. This matrix forms our input. Certain caveats with using the closed form solution are.

1. It is dependent on making sure that the input matrix is invertible
2. We need to find center points for the data and decide on that number of basis functions which becomes additional overhead.

3.1 Modification of reg_lamda:

Reg lamda	ERMS Training	Erms Valdiation	Erms Testing
0.03	0.4017	0.38424	0.53476
0.001	0.4017	0.38424	0.53476
15	0.4017	0.38424	0.53476

3.1 Modification of cluster numbers:

Number of clusters	ERMS Training	Erms Valdiation	Erms Testing
10	0.4017	0.38424	0.53476
15	0.39832	0.383956	0.533602
25	0.39562	0.381518	0.530905
50	7.24E+195	0.37892	0.528206

4 Tuning Parameters for Gradient descent

4.1 Modification of lamda:

For a constant value of learning rate (alpha) initially for extremely small values of lamda, the error was high as seen below. This meant that the value of lamda was to small to correct the overfitting problem in the set number of iteration. On increasing the iterations for same lamda value, the ERMS reduced further and accuracy improved.

As I kept increasing lamda the ERMS was minimum for lamda = 2 and then increased as I increased the value of lamda,albeit extremely slowly. This is because at one point increasing the values of lamda, decreases the values of weights to the extent that the model under-fits

the data.

Gradient descent	ERMS Training	Erms Valdiation	Erms Testing
Lamda			
0.01	1.70342	3.6559	3.73799
2	0.4017	0.38424	0.53476
3	0.40209	0.385015	0.53605
5	0.40258	0.386314	0.53802
10	0.403292	0.38851	0.54107

4.2 Modification of learning rate (alpha):

As seen below small values of alpha, converges slowly to the local minimum and fails to converge to local minimum if the iterations are less. If we keep on increasing alpha, the ERMS continues to reduce, until 0.04 after which it increases sharply. This is the point where the data has over fit and has nullified the effect of regularization.

Gradient descent	ERMS Training	Erms Valdiation	Erms Testing
Learning rate			
0.01 (400 iters)	17.4875	143.447	142.725
0.001 (1200 iter)	0.68139	2.1622	2.27009
0.02	0.4017	0.38424	0.53476
0.04	0.4017	0.384249	0.534769

4.3 Modification of cluster numbers:

Increasing the number of clusters, increases the computation cost of design matrix. Starting from small values of n_clusters and gradually increasing, ERMS improves and maxes out for n_clusters = 15.

Number of clusters	ERMS Training	Erms Valdiation	Erms Testing
1	0.4124	0.39454	0.54862
5	0.40496	0.386801	0.54095
10	0.4017	0.38424	0.53476
15	0.39832	0.39832	0.5336

4.4 Modification of iterations:

For a fixed set of other parameters, increasing the iterations will improve the performance until at one point it crosses the global minima. Printing out the values of erms, after a set of intervals for training verifies this observation as seen below.

```
410.01888484648407
1.41768657880601 02
0.397561531655023 8
0.39739711230668645
0.398224839633107 7
0.398318331327974 5
0.39832829079668924
0.39832936656761 15
done.
```

I printed these values after every 50 iterations, this gives me the idea that optimal number of iterations is between 200 -250.

Conclusion:

We have thus analyzed the two solutions of linear regressions and observed the effect on hyper parameters on both these solutions.