Project 2: Learning to Rank using Linear Regression

Aniket Thigale
UB# : 50168090

A) Objectives:

This project is to implement linear regression to solve regression problem and evaluate its performance. The objective is to learn how to map an input vector x into a scalar target t. In this project, we finish four tasks:

1. Train a linear regression model on real dataset using batch method.
2. Train a linear regression model on real dataset using stochastic gradient descent.
3. Train a linear regression model on synthetic dataset using batch method and evaluate its performance.
4. Train a linear regression model on synthetic dataset using stochastic gradient descent and evaluate its performance.

B) DataSets Used:

1. Real World Set : We use the Microsoft LETOR 4.0 Dataset. LETOR is a package of benchmark data sets for research on Learning To Rank released by Microsoft Research Asia.

2. The only way to truly determine the generalization power of our model would be to test it on an separate unseen dataset. Therefore we will generate synthesized data using some sort of mathematical formula

$$y = f(x) + \varepsilon$$

where ε is noise. After we train our model based on given training set data, the model's performance is tested on another unseen group of data.

C) Building the Models :

1. Data Extraction :

Extract feature values and labels from the data: Process the original text data file into a MATLAB matrix that contains the feature vectors and a MATLAB vector that contains the labels.

2. Data Partition:

First create a random permutation of the data :

```
X_rand = randperm(length(X_new));
```

Then take 80% of data as Training set

```
noTrainDocs = floor(0.8 * length(X_new));
trainingIndexes = X_rand(1:noTrainDocs);
X_training = X_new(trainingIndexes,:);
Y_training = Y_new(trainingIndexes,:);
```

Similarly,
Take 10% of data as Validation set and 10% of data as Test set.

3. Train Model parameters

For a given group of hyper-parameters such as M, μj (mu) , Σj (sigma), λ (lambda), η (learning parameter) train the model parameter w on the training set.
Select a random no of data points (M and mu) from the training set and calculate sigma,ie, the covariance matrix for each data point.
Calculate the design matrix for training and validation points.
The closed-form maximum likelihood solution to linear regression with quadratic regularization, assuming Gaussian noise, has the form

$$w_{ML} = (\lambda I + \Phi^T \Phi)^{-1} \Phi^T t$$

where $t = \{ t_1 \dots t_N \}$ is the outputs in the training data and Φ is the design matrix.

We use the Gaussian radial basis functions :

$$\Phi_j(x) = exp(-\tfrac{1}{2} (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j))$$

For Stochastic Gradient Descent Method we use:

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

where $\Delta w^T = -\eta^T deltaE$ is called weight updates.

where

$$deltaE = -1 * (Y\_training(i,:) - (w^{(T)T} * phi(i,:).')) * phi(i,:) + lambda * w^{T} ;$$

Then calculate training and validation errors for the real and synthetic data sets.
eg) Training Error for real data set:

```
trainingError = 0.5 * (Y_training - (phi * w1)).' * (Y_training - (phi * w1));
trainPer1 = sqrt(((2 * trainingError) / noTrainDocs));
```

eg) Validation Error for real data set:

```
validationError = 0.5 * (Y_validation - (phi2 * w1)).' * (Y_validation - (phi2 * w1));
validPer1 = sqrt(((2 * validationError) / noValidationDocs));
```

4. Tune hyperparameters

Validate the regression performance of your model on the validation set. Change your hyper-parameters and repeat step 3. Try to find what values those hyper-parameters should take so as to give better performance on the validation set.
Find parameters with minimum validation error while not overfitting the date.
Find values for a range of M and λ values. The parameters which gives the minimum validation errors give the best performance. This is also known as Grid Search.
For adaptive learning parameter we can also use the Bold Driver Method.

I tested M1 and M2 for the range 4 to 9 and lambda1 and lambda2 in the range 0.1 to 0.5.
On tuning the hyperparameters I found out that my efficient parameters were as follows:
M1 = 9, M2 = 8
lambda1 = 0.6000 , lambda2 = 0.2000
trainPer1 = 0.5707, validPer1 = 0.5782
trainPer2 = 0.1491, validPer2 = 0.1247

Conclusion

We have thus trained a linear regression model on real and synthetic learning to rank datasets using batch method and stochastic gradient descent method.