Learning to Rank using Linear Regression

Mihir Kulkarni
Computer Science and Engineering
University at Buffalo
Buffalo, NY-14214
UB Person number: 50168610
mihirdha@buffalo.edu

ABSTRACT

This project implements linear regression to solve the regression problem of a real world and a synthetic dataset. We train our model on a training data set and then evaluate the performance on a separate validation set. We are also using stochastic gradient descent to calculate the optimal solution and comparing it with the solution we got from the closed form.

***Index Terms*—Linear Regression, Stochastic Gradient Descent.**

I. Introduction

Linear regression is used extensively in Machine Learning to predict the future values of the output when the input data is given to the system. In this project we are implementing linear regression using closed form solution and trying to the optimal value of the weights by tuning the hyper parameters. We then use the optimal hyper parameters and try to get the optimal weights using stochastic gradient descent. We compare the weights generated from closed form solution and from stochastic gradient descent. For the closed from solution we first calculate basis function, variance matrix and randomly generated means.

We are implementing this project on MATLAB software by Mathworks.

II. Data set

The real world data set is Microsoft LETOR 4.0 Dataset. From this data set, we are using 'Querylevelnorm.txt' which consists of 69623 query-document pairs each consisting of 46 features. We are using 80% of these query-document pairs as training set to train our model. 10% is used as a validation set to check the performance of the model. Rest of the pairs are used as test set.

Synthetic data set is the data set given which is generated using some mathematical formula.

$$y = f(\mathrm{x}) + \varepsilon$$

Here f(x) is a function unknown to us and ε is noise. This unseen dataset will help us evaluate our model more effectively. We are dividing this data set and using 80% of the data set as training set to train our model, while rest 20% will be used as validation set.

III. Mean, variance and basis function

We are selecting random M means, where M is model complexity and each mean is 1xD vector. D is number of features in each data sample. Currently in the project, we are selecting random M data samples from the training data. A more sophisticated method for selecting mean would be K means clustering. It would generate more accurate basis function.

For the training data, we calculate variance. Representing this vector of variance in a diagonal matrix we get Sigma1. Dimension of Sigma if DxDxM.

$$\Sigma = \begin{pmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_D^2 \end{pmatrix}$$

Where,

$$\sigma_i^2 = \tfrac{1}{10}\mathrm{var}_i\,(\mathrm{x}).$$

To calculate a DxDxM Sigma matrix we are replicating DxD plane M times.

Basis function is a matrix of dimension NxM where N is number of training data samples. It is calculated as-

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathrm{x}_1) & \phi_1(\mathrm{x}_1) & \phi_2(\mathrm{x}_1) & \cdots & \phi_{M-1}(\mathrm{x}_1) \\ \phi_0(\mathrm{x}_2) & \phi_1(\mathrm{x}_2) & \phi_2(\mathrm{x}_2) & \cdots & \phi_{M-1}(\mathrm{x}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathrm{x}_N) & \phi_1(\mathrm{x}_N) & \phi_2(\mathrm{x}_N) & \cdots & \phi_{M-1}(\mathrm{x}_N) \end{bmatrix}$$

Where,

$$\phi_j(\mathrm{x}) = \exp\left(-\frac{1}{2}(\mathrm{x}-\boldsymbol{\mu}_j)^\top \Sigma_j^{-1}(\mathrm{x}-\boldsymbol{\mu}_j)\right)$$

IV. Closed form solution and RMS error

Closed form solution is calculated using basis function, Sigma, lambda and M.

$$\mathrm{w}_{\mathrm{ML}} = (\lambda\mathbf{I} + \mathbf{\Phi}^\top\mathbf{\Phi})^{-1}\mathbf{\Phi}^\top\mathbf{t}$$

For each solution, we find RMS error on training and validation set namely trainPer and validPer. We calculate the error as-

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^*)/N_{\mathrm{V}}}$$

V. Grid search

Changing values of lambda and model complexity changes solution and respective errors. To find the optimal solution, we must perform a grid search on lambda and model complexity. For each value of lambda and for each model complexity we find the closed form solution and RMS errors. We select the values of lambda and model complexity for which the RMS error on error on validation set is minimum.

VI. Stochastic gradient descent

Stochastic gradient descent works on single data sample at a time unlike Batch gradient descent. We start with an initial random value of solution. We modify this solution for each data sample.

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

Where,

$$\nabla E = \nabla E_D + \lambda \nabla E_W$$

In which,

$$\nabla E_D = -(t_n - \mathbf{w}^{(\tau)\top}\phi(\mathbf{x}_n))\phi(\mathbf{x}_n)$$

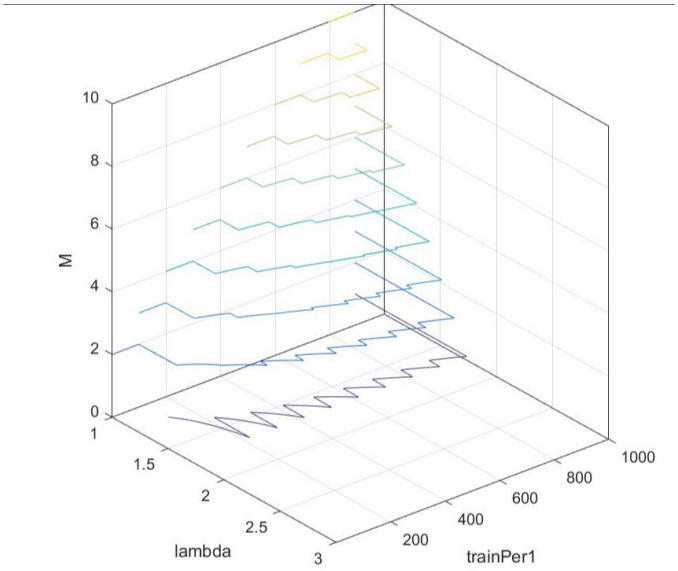
$$\nabla E_W = \mathbf{w}^{(\tau)}$$

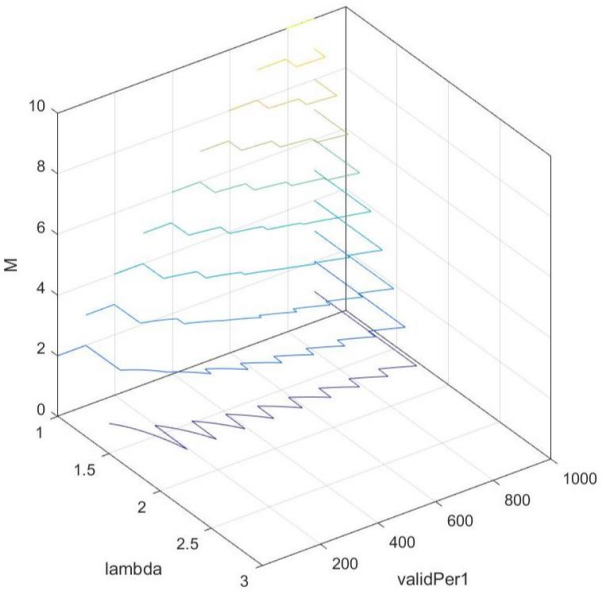
This process generates next solution which is used as previous solution for the next step. The process can be repeated E number of times where E is greater than N. That means we can iterate over the same data set multiple times to get accurate solution.

In our project, we need the norm of difference between final solution of stochastic gradient descent and closed form solution to be less than norm of difference between initial solution of stochastic gradient descent and closed form solution.

VII. Plotting variables

We are plotting all the pairs of variables against each other using commands of MATLAB





References

[1] https://www.willamette.edu/~gorr/classes/cs449/momrate.html

[2] Bishop - Pattern Recognition and Machine Learning

[3] Class material and TA's guidance.