Learning to Rank using Linear Regression

Ruturaj Molawade
University at Buffalo
UBIT - ruturajt
ruturajt@buffalo.edu

Abstract

Learning to Rank algorithm is used to solve problems that come to light in Information Retrieval.LeToR is used in document retrieval, definition search, key phrase extraction, sentiment analysis and anti-web spam.LeToR is not focused on providing exact score rather it gives relative ordering among searches. The goal of project is to solve LeToR problem using linear regression model which maps set of input features to target values. This problem is classified as supervised learning problem where we are provided with huge data set with corresponding output. We have used closed form solution and stochastic gradient approach to obtain linear regression solution.

1 Linear Regression Model

Linear regression is widely used in predictive analysis. Linear regression models are used in statistics for understanding relation between input and output variables e.g. models that gives relationship between input variables (X) and output variable (Y). Mathematically it can be represented as

$$Y = aX + b$$

where, Y is estimated variable score, X is input variable and b is regression coefficient. The linear regression can also be formulated in another way which takes input as weights and feature variables and gives target value.

$$y(x, w) = w^T \phi(x)$$

w is weight vector obtained from training the model on training set. After every epoch or iteration this weight gets updated.weights are updates in such way that it minimizes the difference between actual values and predicted values.

$$w = (w_0, w_1, w_2, w_3, w_4, ..., w_{M-1})$$

whereas ϕ is vector of basis function. We have used ϕ to transfer non linear relationship among input variables and target output into linear form.

$$\phi = (\phi_0, \phi_1, \phi_2, \phi_3, , ..., \phi_{M-1})$$

In this project we are using Gaussian radial basis functions as this function can model non linear relationship between input values and target. As well as this basis function will have more zero values as compare to non zero values which gives sparse matrix that is easy to compute.

$$\phi_j(X) = \exp(-1/2(x - \mu_j)^T (\Sigma_j)^{-1} (x - \mu_j))$$

μ_j is the center of basis function.Number of basis functions are calculated using K-means clustering which divides data randomly into N clusters and gives centroids of those clusters.

2 Likelihood Function

Maximum Likelihood Estimation helps to choose the parameters that helps to describe training data other than any other set of parameters.Objective function derived is based on principal of maximum likelihood. In likelihood function we consider joint probability of how response data Y is dependent on input values X vector and other parameters. This function compares multiple models to determine best fit of data. It maximizes log-likelihood function to estimate parameters. Likelihood function can be formulated as below:

$$p(t|X, w, \beta) = \Pi N(t_n|W^T\phi(x_n), \beta^{-1})$$

3 Regularization

While training the model we need to avoid problem of over fitting. Over fitted model gives low accuracy. Over fitting the model means that it covers unnecessary data points which don't really represent true properties.

$$E(w) = E_D(w) + \lambda E_W(w)$$

For regularization we need to add regularization term to error function.Regularization, reduces the variance of the model, without increase in its bias. It's also referred as unlearning in which we omit data points which don't provide value to the data.

There is concept called early stopping when we talks about regularization. In linear regression we calculate gradient descent which tries to learn more and more if number of iterations are increased. It forms very complex curve or equation which passes through almost every data point in given sample.

4 Closed-form Solution for W

Closed form solution is used to calculate weights. It's just the another method to achieve weights. Closed form solution is preferred when calculating matrix inverse is not a concern. It is formulated as

$$W_{ML} = (\phi^T\phi)^{-1}\phi^T t$$

t is the target vector i.e. $t = \{t_1, .., t_N\}$ and ϕ is the design matrix. We have to follow steps below to calculate weights.

1.We have to find design matrix ϕ. Size of design matrix would be (No. of data rows, No. of basis functions). Number of basis function itself is a hyper parameter. On given value of number of basis functions K- means clustering forms data clusters and gives centroids i.e. μ for every cluster. ϕ is given by

$$\phi_j(X) = \exp(-1/2(x-\mu_j)^T(\Sigma_j)^{-1}(x-\mu_j))$$

2. To calculate ϕ we need to find co variance matrix. Here we have 46 features out of this 5 features are omitted as their variance is zero. Co variance matrix is co relation between variance of one feature vector with other feature vectors and the vector itself.

3. In above equation x is data row from data matrix and μ is mean vector. Data vector and mean vector has same size i.e. (1,41) or (41,1).

$$\omega = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & ... & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \phi_2(x_2) & ... & \phi_{M-1}(x_2) \\ \phi_0(x_3) & \phi_1(x_3) & \phi_2(x_3) & ... & \phi_{M-1}(x_3) \\ \phi_0(x_4) & \phi_1(x_4) & \phi_2(x_4) & ... & \phi_{M-1}(x_4) \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \\ \phi_0(x_n) & \phi_1(x_n) & \phi_2(x_n) & ... & \phi_{M-1}(x_n) \end{pmatrix} \tag{1}$$

4. After calculating design matrix our next task is to find vector w which is given by

$$W_{ML} = (\phi^T \phi)^{-1} \phi^T t$$

here we have used Moore-Penrose pseudo-inverse because our design matrix is not square matrix here.

5 Stochastic Gradient Descent Solution for w

Now we will discuss another method to calculate weights i.e. Stochastic Gradient Descent Solution. Stochastic Gradient Descent is very similar to Gradient Descent the only difference is that in Gradient Descent we compute gradient based on complete training set. If we are dealing with large data set GD can be quite costly as it's taking single step for every pass on sample data. Eventually, convergence rate is less in this case. On other hand SGD doesn't compute Gradient Descent for every single data sample rather we choose some set of random points from whole data set.

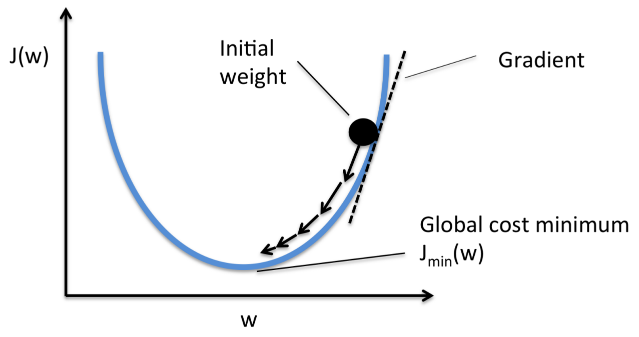
SGD converges faster as compare to GD but the error function is not always as minimized as we obtained one in GD. We need to consider trade-off between correctness and speed when it comes to selecting one of the above approach. Generally approximation we find in SGD for parameters is good enough because many times once it reaches to minimum it just oscillates here and there. To calculate stochastic gradient descent we follow steps below:

1. First we initialize with some random weights.

2. Then we update weights by using below equation.

$$w^{\tau+1} = w^{\tau} + \Delta w^{\tau}$$

where Δw^{τ} is weight update which is formulated as $\Delta w^{\tau} = -\eta^{\tau} \nabla E$, η^{τ} is **learning rate**, decides how much weight is need to be adjusted with respect to loss gradient.



where ∇E is given by

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

$$\nabla E_D = -(t_n - w^{(T)T} \phi(X_n)) \phi(x_n) \tag{2}$$

6 Evaluation

Model is being evaluated using root mean square method.

$$E_{RMS} = \sqrt{2E(w^*)/N_v}$$

7 Training, Testing, Validation Dataset

- **Training Dataset**: Training data set contains data sample which are used for training models i.e. sample of data used to fit model. The model observes the data and lerans from it.
- **Testing Dataset**: Training data set is used to examine performance of trained model. Model is tested only once it has completed training. It gives generalized notion about model fitting. If model is over trained on training samples it won't gives better accuracy on testing data generally such model is described as example of over fitting.
- **Validation Dataset**: Validation data set is used to minimize over fitting. Validation data set helps to tune parameters. In neural networks we don't update weights in validation set but we tend to change hyper parameters.

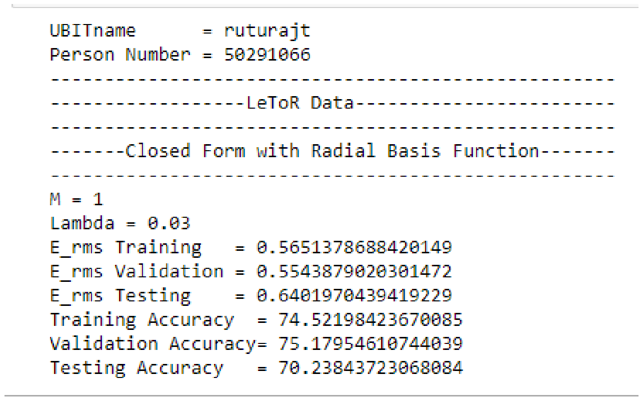
8 Results for Closed Form Solution :

Hyper-parameters parameters

Accuracy Obtained for Closed Form

			Training		Validation		Testing	
	M	λ	E RMS	Accuracy	E RMS	Accuracy	E RMS	Accuracy
1.	1	0.03	0.5651	74.52	0.5543	75.17	0.6401	70.23
2.	2	0.03	0.5627	74.52	0.5509	75.17	0.6384	70.23
3.	8	0.03	0.5506	74.13	0.5408	74.86	0.6292	69.90
4.	15	0.03	0.5406	74.08	0.5307	74.60	0.6192	68.50
5.	25	0.03	0.5489	74.45	0.5500	73.10	0.6208	69.07
6.	55	0.03	0.5945	74.12	0.5373	74.12	0.6245	69.51
7.	100	0.9	0.5216	74.18	0.5663	74.11	0.6470	67.32
8.	5	0.04	0.5428	74.34	0.5510	73.13	0.6451	71.11
9.	10	0.04	0.5894	74.92	0.5374	74.67	0.6779	69.87
10.	15	0.04	0.5459	73.50	0.5264	74.47	0.6173	69.04
11.	20	0.04	0.5561	74.32	0.5390	73.70	0.5407	68.55
12.	30	0.04	0.5790	73.23	0.5490	73.75	0.6141	68.60

Best Accuracy obtained for closed form solution :



```
UBITname      = ruturajt
Person Number = 50291066
----------------------------------------------------
------------------LeToR Data------------------------
----------------------------------------------------
-------Closed Form with Radial Basis Function-------
----------------------------------------------------
M = 1
Lambda = 0.03
E_rms Training   = 0.5651378688420149
E_rms Validation = 0.5543879020301472
E_rms Testing    = 0.6401970439419229
Training Accuracy  = 74.52198423670085
Validation Accuracy= 75.17954610744039
Testing Accuracy   = 70.23843723068084
```

9 Results of Stochastic Gradient Descent:

Results Obtained from SGD

			Training	Validation	Testing
	Learning Rate	Epochs	E RMS	E RMS	E RMS
1.	0.03	400	0.5496	0.5484	0.6137
2.	0.03	800	0.5596	0.5384	0.61304
3.	0.08	200	0.6730	0.6573	0.7426
4.	0.09	400	0.5751	0.6740	0.6409

References

- (https://thenewstack.io/letor-machine-learning-web-search-technique-thats-turned-key-info
- (https://chemicalstatistician.wordpress.com/tag/gaussian-basis-function-models/)
- (https://qph.fs.quoracdn.net/main-qimg-b7a3a254830ac374818cdce3fa5a7f17.webp)