Introduction to Machine Learning

Project 1: Regression

Sagar Vishwakarma

Per. No.: 50098079

Contents

Problem Statement

The project is to implement a linear regression on web search ranking data set. In the regression task, a dataset of vectors and target values is given, and we will need to apply regression methods to learn regression function which is used to rank vectors. We are asked to implement a linear regression method discussed in class, and compare the performance of our code with that of a downloadable regression package available on the web.

Theory

The simplest linear model for regression is given by:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

The key property of this model is that it is a linear function of the parameters $w_0, \ldots, w_D$.

A linear combinations of fixed nonlinear functions of the input variables can also be used as their prediction is much better than the simple model. It is given by:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

where $\phi_j(\mathbf{x})$ are known as *basis functions*.

And

where $\mathbf{w} = (w_0, \ldots, w_{M-1})^T$ and $\phi = (\phi_0, \ldots, \phi_{M-1})^T$

Now we calculate the parameters W in terms of ϕ and target vector t.

LINEAR REGRESSION

$$\hat{y} = w_0 + w_1\phi(x) + w_2\phi(x^2) + \cdots \quad w_{M-1}\phi(x^{M-1})$$

thus

$$\hat{y}_i = w_0 + \sum_{j=1}^{M-1} w_j \phi(x_i^j)$$

$\hat{y}_i$ is an approximation of t_i

Error in estimation $E_i = \hat{y}_i - t_i$

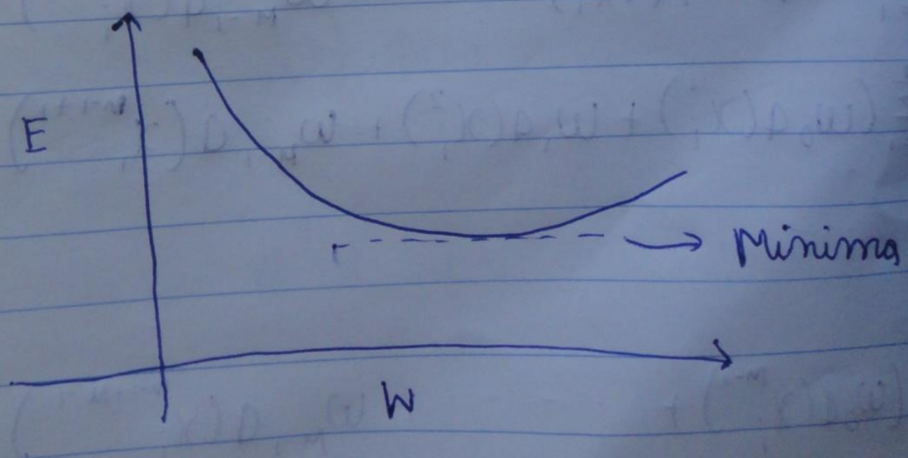
$$= w_0 + \sum_{j=1}^{M-1} w_j \phi(x_i^j) - t_i$$

Or

$$\boxed{E_i = \sum_{j=0}^{M-1} w_j \phi(x_i^j) - t_i}$$ - For 1 datapoint

$\therefore$ for N datapoints

$$E = \sum_{i=\phi}^{N} E_i = \sum_{i=1}^{N}\left(\sum_{j=0}^{M-1} w_j \phi(x_i^j) - t_i\right)$$



Now

$$\text{Square error} = \frac{1}{2} \sum_{i=1}^{N} \left[\left(\sum_{j=0}^{M-1} w_j \phi(x_i^j) - t_i \right) \right]^2$$

For finding minima, we differentiate wrt w_0, $w_1 \cdots w_{M-1}$

$$\therefore \frac{\partial E}{\partial w_0} = \frac{\cancel{2}}{\cancel{2}} \sum_{i=1}^{N} \left(\left(\sum_{j=0}^{M-1} w_j \phi(x_i^j) - t_i \right) \right) \cdot 1 = 0$$

$$\frac{\partial E}{\partial w_1} = \frac{\cancel{2}}{\cancel{2}} \sum_{i=1}^{N} \left(\left(\sum_{j=0}^{M-1} w_j \phi(x_i^j) - t_i \right) \right) \cdot \phi(x_i) = 0$$

$$\vdots$$

$$\frac{\partial E}{\partial w_{M-1}} = \frac{\cancel{2}}{\cancel{2}} \sum_{i=1}^{N} \left(\left(\sum_{j=0}^{M-1} w_j \phi(x_i^j) - t_i \right) \right) \phi(x_i^{M-1}) = 0$$

Now, expanding the terms we get

$$\sum_{i=1}^{N} \left(w_0 + w_1 \phi(x_i^1) + \cdots w_{M-1} \phi(x_i^{M-1}) \right) = \sum_{i=1}^{N} t_i$$

$$\sum_{i=1}^{N} \left(w_0 \phi(x_i^1) + w_1 \phi(x_i^2) + \cdots w_{M-1} \phi(x_i^{M-1+1}) \right) = \sum_{i=1}^{N} t_i \phi(x_i)$$

$$\vdots$$

$$\sum_{i=1}^{N} \left(w_0 \phi(x_i^{M-1}) + \cdots + w_{M-1} \phi(x_i^{M-1+M-1}) \right) = \sum_{i=1}^{N} t_i \phi(x_i^{M-1})$$

Writing in matrix form

$$\underbrace{\begin{bmatrix} \Sigma 1 & \Sigma \phi(x_i) & \Sigma \phi(x_i^2) & \cdots & \Sigma \phi(x_i^{M-1}) \\ \Sigma \phi(x_i) & \Sigma \phi(x_i^2) & \cdots & \cdots & \Sigma \phi(x_i^{M-1+1}) \\ \vdots & & & & \\ \Sigma \phi(x_i^{M-1}) & \cdots & \cdots & \cdots & \Sigma \phi(x_i^{M-1+M-1}) \end{bmatrix}}_{[\phi(x)^T \cdot \phi(x)]} \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix}}_{W}$$

$$= \underbrace{\begin{bmatrix} \Sigma t_i \\ \Sigma t_i \phi(x_i^1) \\ \vdots \\ \Sigma t_i \phi(x_i^{M-1}) \end{bmatrix}}_{\phi(x)^T \cdot t}$$

Rewritting we get

$$[\phi(x)^T \cdot \phi(x)] \cdot W = \phi(x)^T \cdot t$$

multiplying by $[\phi(x)^T \cdot \phi(x)]^{-1}$ we get

$$W = [\phi(x)^T \phi(x)]^{-1} \phi(x)^T \cdot t$$

or

$$\boxed{W = (\phi^T \phi)^{-1} \phi^T \cdot t}$$

One technique that is often used to control the over-fitting phenomenon and the singular problem of $\Phi^T\Phi$ is that of *regularization*, which involves adding a penalty term to W.

$$\mathbf{w} = \left(\lambda \mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^{\mathrm{T}}\mathbf{t}.$$

Basis Functions

Gaussian Basis Function

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

Sigmoidal Basis Function

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$

Where:-

$$\sigma(a) = \frac{1}{1+\exp(-a)}.$$

Implementation

Importing the Dataset

The whole Microsoft Letor 4.0 dataset in imported in an excel file. The required details i.e. Relevance labels and all the 46 features are kept and the rest of the data is deleted from the file.

The dimension of the file obtained is thus 15211 by 47.

Now, the whole dataset is divided into three files:

a. Training data (40 %)
   1. Training features (2^{nd} to 47^{th} column)
   2. Training labels (1^{st} column)
b. Validation data (10%)
c. Testing data (50%)

Each of the files is converted into a .mat file to pass into the .m files.

Training

A train.m file is created which takes training features, training labels, M and Lambda as inputs**. For this project, the Gaussian Basis function is chosen.**

This m file calculates mean (mu) and sigma as follows:

a. The whole features data set is divided into M-1 parts
b. From each part, a mean and a hyperparamter calculated according to the variance is obtained by the below formula:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

$$s^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2$$

   Thus giving a total of M-1 means and hyperparamters
c. Now, for each part a ϕ is calculated by using the respective values of mean and hyperparamter accordingly as:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

   Thus giving a total of M-1 ϕ*'s*
d. Now Each ϕ is applied to each row of the training features thus giving the bigger ϕ as:

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

e. Now, Parameters W are calculated by:

$$\mathbf{w} = \left(\lambda \mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$$

Validation

A predict.m file is created which takes validation data, M and Lambda, mean, hyperparamter, W as inputs.

a. **The value of M = 2 to 20 and Lamda =1 to 4 with step size of 0.5**
b. The validation data is divided into features and labels.
c. Using W, means and hyperparamters, A predicted labels of the validation dataset is calculated for different M and Lambda.
d. From Each predicted label, a root mean square error is calculated as:

$$E_{rms} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - t_i)^2}{N}}$$

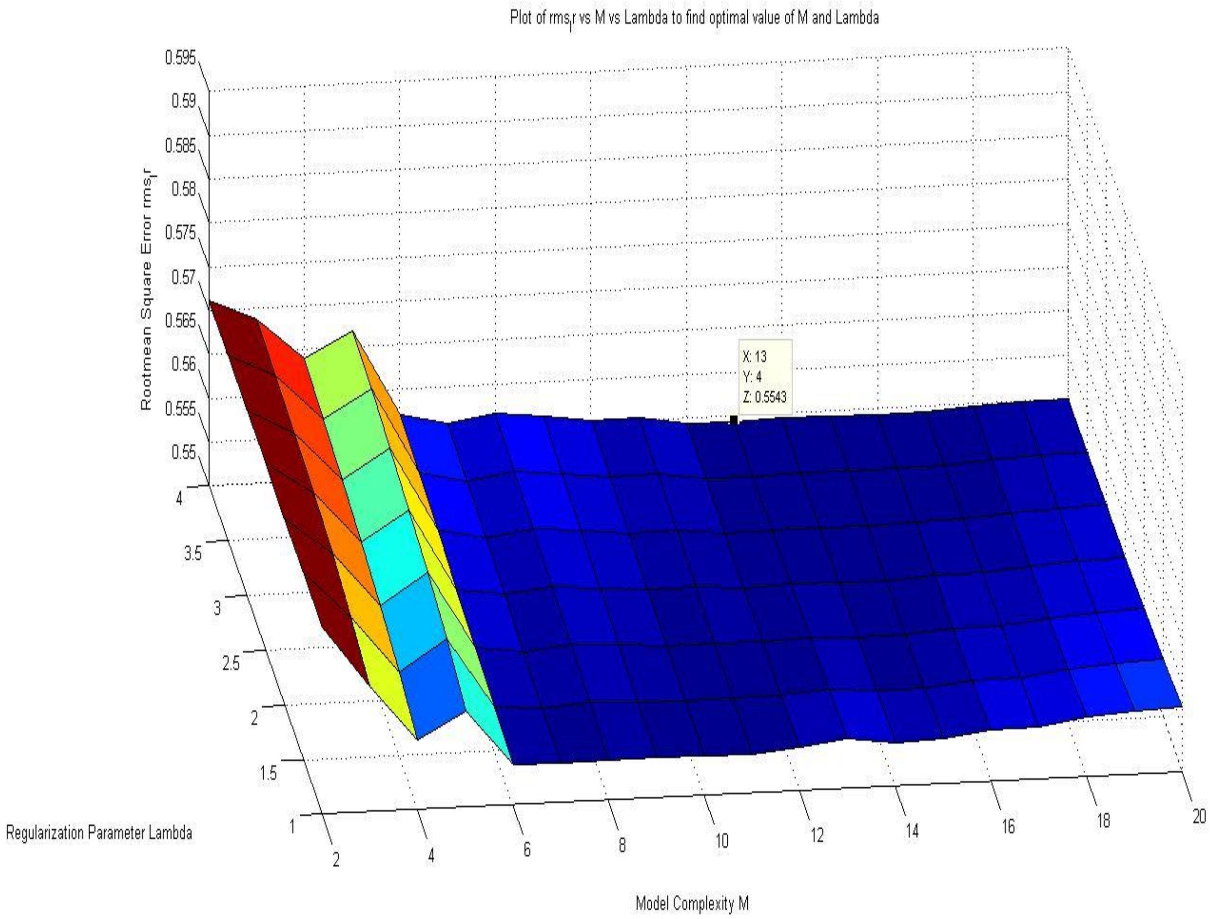
e. A plot of rms_lr vs M vs Lambda is generated and the minimum values of rms_lr is noted.
f. Thus we get the optimal values of M and Lambda.

Testing

a. With the optimal values of M and L, The testing features is passed into predict.m that specific model and predicted labels for testing data are calculated and subsequently the rms_lr is obtained.
b. The same testing data is passed into the neural networks basis function set and the rms_nn is obtained.

Observations

The following graph is obtained for rms_lr vs M vs Lambda:



Here, the optimum values of M and Lambda are obtained are:

M = 13

Lambda = 4

Using these values,

The rms_lr obtained for Validation data set = 0.5643

The rms_lr obtained for testing data set = 0.601

For Neural networks, the rms_nn obtained for testing data set = 0.2563

Conclusion

For the given values of μ and s, the best result is obtained by using a regression model which implements the Gaussian basis function with an order (M) of **13** and a regularization factor (Lambda) of **4** from the point of view of the root mean square error.

The Neural network proves to be a much better model than linear regression comparing the root mean square error of each of them respectively.