Project 2: Learning to Rank using Linear Regression

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

The goal of this project is to use machine learning to solve a problem that arises in Information Retrieval, one known as the Learning to Rank (LeToR) problem. We formulate this as a problem of linear regression where the input is a vector value which is mapped to a function y(x,w) to derive the target value t. Here the target value t is a scalar quantity and w represents the weight.

We have two objectives in this project:

1. Train a linear regression model on LeToR dataset using a closed-form solution.

2. Train a linear regression model on the LeToR dataset using stochastic gradient descent (SGD).

The LeToR training data consists of pairs of input values x and target values t. The input values are real-valued vectors (features derived from a query-document pair). The target values are scalars (relevance labels) that take one of three values 0, 1, 2: the larger the relevance label, the better is the match between query and document.

We can apply this model to learn from the training data and then apply that to the testing data to derive output vector from it. the output vector can then be compared to actual target value to calculate error.

1 General Concepts

1.1 What is Linear regression?

Linear regression attempts to model the relationship between two variables by fitting a linear equation to the previously observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.

for example: one might want to determine the price of an apartment based on how big the apartment is. Here the price is the explanatory variable and the size of the apartment is the dependent variable.

A linear regression line has an equation of the form $Y = bX + c$, where X is the explanatory variable and Y is the dependent variable.

When we consider linear regression in machine learning, the variable b and c can be replaced by the corresponding weights. So the equation can look like the following: Y = $wX_1 + wX_2 +$ ... where w represents the weight.

For this project the linear regression equation can be represented as: $y(w, x) = w^T \phi x$

Where, $w = (w_1; w_1, ..., w_M - 1)$ is a weight vector to be learned from training samples and $\phi = (\phi_0, ..., \phi_M - 1)^T$ a vector of M basis functions.

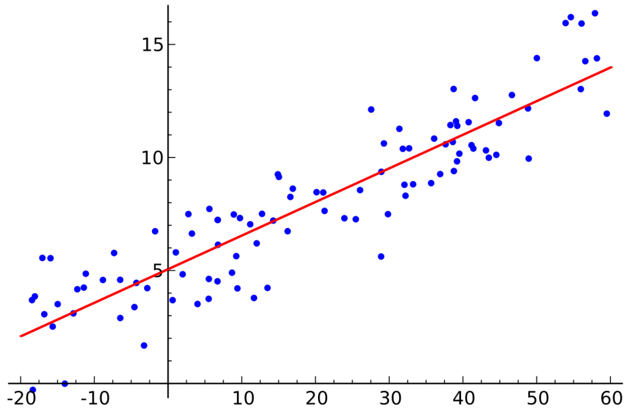


Figure 1: Neural Network

1.2 What is Basis Function?

In this project we have 46 features in the input vector, in simpler terms, for each output y the corresponding input vector x has 46 elements. Now if we look into the definition of linear regression, there was only one dependent variable x for each of the explanatory variable y. So it is impossible to plot multiple x values against one singular y value.

As a solution to this problem we can represent y as the combination of M basis functions: $y = \phi(x) + \phi_1(x_1) + \phi_1(x_2) + ... + \phi_M(x_M - 1)$

Basis function helps to represent non linear scalar values of x in a linear form.

In this project, we are using Gaussian curve to represent the basis functions, i.e. there are M Gaussian curve corresponding to M basis functions.

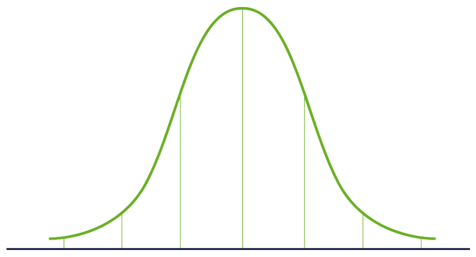


Figure 2: Gaussian Curve

2 Closed-form Solution for w

In a linear regression model there are M numbers of basis functions ($\phi(x)$) representing M number of Gaussian curves. If we put all the basis functions together into a single matrix, that matrix is called the design matrixc(Φ), which looks like the following:

$[\phi_1(x) + \phi_1(x_1) + \phi_1(x_1) + ... + \phi_M - 1(x_1)]$

$[\phi_2(x) + \phi_2(x_2) + \phi_2(x_2) + ... + \phi_M - 1(x_1)]$

....

$[\phi(x_N) + \phi_1(x_N) + \phi_1(x_N) + ... + \phi_M - 1(x_N)]$

As mentioned earlier, in this project we can represent the output y as:

$y(w, x) = w^T \phi x$ ——-> (1)

Where, $w = (w_1; w_1, ..., w_M - 1)$ is a weight vector to be learned from training samples and $\phi = (\phi_0, ..., \phi_M - 1)^T$ a vector of M basis functions.

Therefore in order to achieve our objective, i.e. in order to be able to plot all the x we need to determine the value of the the weight matrix w.

from (1) we can simply calculate the value of w by multiplying the target vector y with the inverse of the Design Matrix (Φ^T).

But here, the problem is that the Design Matrix (Φ) is not a square matrix, as we all know in order to be inversed a matrix has be a square matrix.

To, solve this problem, we will apply moore-penrose generalized inverse method to derive the weight matrix:

$w_M = (\Phi^T \Phi^- 1)^- 1.\Phi^T.t$

where $t = [t_1, t_2, ..., t_n]$ is the vector of outputs in the training data and Φ is the design matrix.

Another form for writing the equation of moore-penrose generalized inverse is the following: $w_* = (I + \Phi^T \Phi^- 1)^- 1.\Phi^T.t$

where is called the **"Regularizer"**: Regularizer helps the model to avoid over-fitting.

Tn the closed form solution method, there is no iteration, so there is no scope of reducing the error by adjusting the weights. Therefore, the error is normally higher in this method than the gradient descent approach.

2.1 Stochastic Gradient Descent Solution for w

In this method also we start with the following linear regression equation for this project:

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

Where, $w = (w_1; w_1, ..., w_M - 1)$ is a weight vector to be learned from training samples and $\phi = (\phi_0, ..., \phi_M - 1)^T$ a vector of M basis functions.

We start by putting two random values as the initial weights. Then we calculate the error function or cost function which represents the error in the linear regression model.

Our objective in the gradient descent approach is always to reduce the cost function with each iteration. With each iteration, we adjust the weights in such a way that the error function or the cost function reduces down to the minimum.

The following equation represents the gradient descent mechanism:

$\Delta E = -\alpha(t_n - (w^T \phi(x_n))) - \phi(x_n) + \Delta w$

Where, ΔE ———-> Change in the error

α ————> Learning rate

t_n ————> Target value

$(w^T \phi(x_n))$ —> Predicted value

Δw ——> Regularizer

Regardless of the slope's sign, ΔE will always converge to the minimum value, eventually.

The size of each steps taken towards the minimum value is decided by the learning rate α. Smaller α will result in smaller steps and bigger α will result in bigger step size.

taking too small or too large value for α can result in unreasonably long time to converge and completely failing to converge to the minimum, respectively.

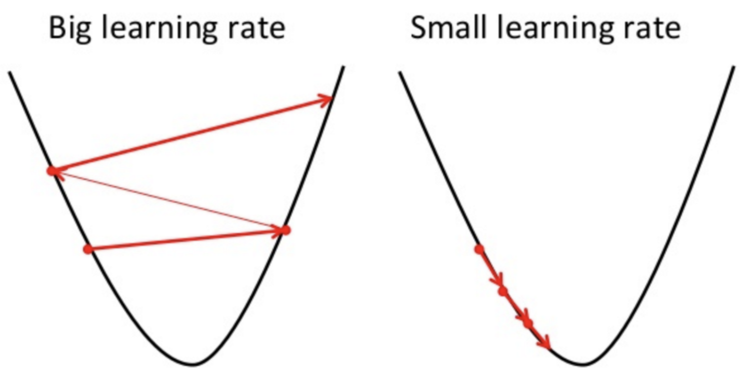


Figure 3: Gradient Descent

2.2 Difference between gradient descent and stochastic gradient descent:

In Gradient Descent approach, we calculate the cost function based on the complete training set; hence, sometimes it is also call it batch Gradient Descent.

In case of very large data sets, Gradient Descent can be quite costly since we are only taking a single step for one pass over the entire training set – thus, larger the training set, slower the algorithm updates the weights and the longer it might take to converges to the minimum value.

On the other hand, in case of Stochastic Gradient Descent, we use only a subset of the entire training set to do the update for a parameter in a particular iteration. As we are using subsets, it is also called mini batch Gradient Descent, For example:

The stochastic gradient descent algorithm first takes a random initial value $w^{(}0)$. Then it updates the value of w using:

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

where $\Delta w^{(}\tau) = \eta^{(}\tau)\nabla E$ is called the weight updates. It goes along the opposite direction of the gradient of the error. $\eta^{(}\tau)$ is the learning rate, deciding how big each update step would be. Because of the linearity of differentiation, we have $\nabla E = \nabla E_D + \delta \nabla E_W$ in which $\nabla E_D = -(t_n - w^{(}tau)^T \phi x(n))(n)$

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

2.2.1 Pre-processing of data:

In this project we have divided the entire LeToR data set into M clusters. Then for each cluster, we have calculated a mean value (μM). The mean value will then be subtracted from each of the term of the cluster to calculate the variance.

Now, in the LeToR data set, there are few such columns which have 0 in all the elements. Therefore, those columns will have 0 variance and will make no effect on the model.

Therefore, hose columns have been removed before using the data set.

```
def GetTargetVector(filePath):
    t = []
    with open(filePath, 'rU') as f:
        reader = csv.reader(f)
        for row in reader:
            t.append(int(row[0]))
    #print("Raw Training Generated..")
    return t
```

```

def GenerateRawData(filePath, IsSynthetic):
    dataMatrix = []
    with open(filePath, 'rU') as fi:
        reader = csv.reader(fi)
        for row in reader:
            dataRow = []
            for column in row:
                dataRow.append(float(column))
            dataMatrix.append(dataRow)

    if IsSynthetic == False :
        dataMatrix = np.delete(dataMatrix, [5,6,7,8,9], axis=1)
    dataMatrix = np.transpose(dataMatrix)
    #print ("Data Matrix Generated..")
    return dataMatrix
```

2.2.2 Evaluation of Error:

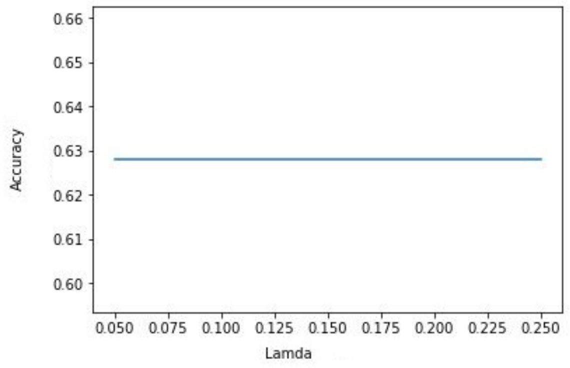
In this project we have used root mean square method to evaluate the error function.

The root mean square function can be written as following: $E_RMS = \sqrt{(t_v - p_v)^2/N_V}$

where t_v is the target value, p_v is the predicted value and N_V is the number of observations.

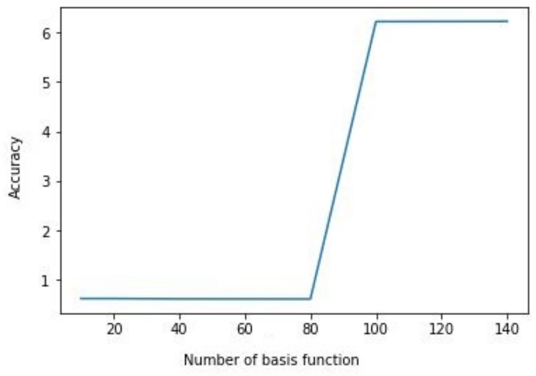
3 Hyper-parameters:

a Changing the lamda value doesnt affect the accuracy significantly, the ERMS value remains almost constant:

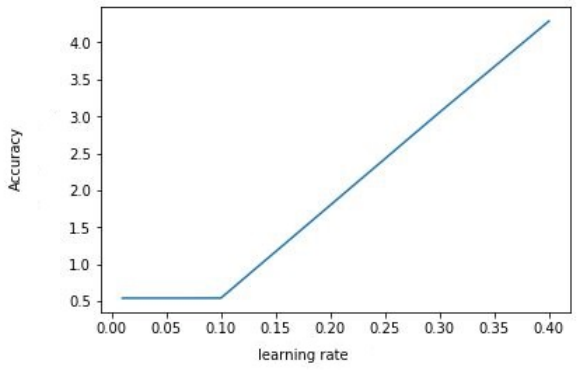


b By increasing learning rate, the accuracy doesnt change at first, but if it is increased after 0.10, the accuracy increases significantly :

c By increasing learning rate, the accuracy doesnt change at first, but if it is increased after 0.10, the accuracy increases significantly :



function.JPG



rate.JPG

4 references

https://www.coursera.org/learn/machine-learning

Bishop - Pattern Recognition And Machine Learning - Springer 2006

https://en.wikipedia.org/wiki/Machine$_learning$