UBitName: abhave personNumber:50289049

CSE 574 Introduction to Machine Learning

Project 1.2 Report

Our linear regression function t has the form:

$$t = w^T\phi(X)$$

where w = (w_1, w_2..., w_m) is a weight vector to be learnt from training samples and ϕ = (ϕ_1,…, ϕ_m)>is a vector of M basis functions. We need basis functions to convert non-linear hypothesis to linear hypothesis. Our regression function can also be expressed as:

$$t= w_1 \phi_1(X) + w_2 \phi_2(X) + ... + w_m \phi_m(X)$$

LeToR dataset

LETOR is a package of benchmark data sets for research on Learning TO Rank, which contains standard features, relevance judgments, data partitioning, evaluation tools, and several baselines. LETOR4.0 contains 8 datasets. The 5-fold cross validation strategy is adopted, and the 5-fold partitions are included in the package. In each fold, there are three subsets for learning: training set, validation set and testing set.

The first few steps are common to both closed form and stochastic gradient descent solution. They are as follows:

1 - Data Processing

(a) We have dataset called 'Querylevelnorm_X.csv'. This dataset has 69623 rows and 46 columns. We read this dataset and divide the dataset into training set, testing set, and validation set. The division is 80%, 10% and 10% for training set, testing set, and validation set respectively.
Training set is used to train the model and then the regression model that we created is tested on the testing set.
Validation Set
We create a validation set so that the model can be tested on unseen data and validation error is generated

(b) The dataset is reduced to 69623 rows and 46 columns. We delete these 5 columns because the variance of these 5 columns comes to zero as most of the values in these 5 columns is 0.

2– Closed Form Solution

(a) k-Means Clustering

We have 46 columns in our dataset. It is difficult to do some may computations and may take longer time. Thus, to reduce time and make our model more efficient we use k-Means clustering and we get 10 different clusters with 10 means and thus we make 10 basis functions.

We get 10 clusters as k-Means clustering requires us to specify number of clusters we want.

(b) BigSigma (Σ)

BigSigma is the covariance matrix that will have the dimensions 41 X 41. It is represented byΣ. The matrix will be a diagonal matrix with all the diagonal elements as σ_{11}^2, σ_{22}^2 and so on up to σ_{4141}^2.All other elements will be zero.

We multiply each value of BigSigma by 200 as all values are in the range of 10^{-2} and 10^{-3} which is very. So we multiply all values by 200.

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

(c) Calculating Training Design Matrix $\phi(X)_{TR}$, Testing Design Matrix $\varphi(X)_{TE}$ and Validation Design Matrix $\phi(X)_{VA}$

From the formula in the above step we calculate Training Design Matrix $\phi(X)_{TR}$, Testing Design Matrix $\varphi(X)_{TE}$ and Validation Design Matrix $\phi(X)_{VA}$ as follows:

Training Design Matrix $\phi(X)_{TR}$ will be a matrix with dimensions 55698 X 10
Testing Design Matrix $\varphi(X)_{TE}$ will be a matrix with dimensions 6963 X 10
Validation Design Matrix $\phi(X)_{VA}$ will be a matrix with dimensions 6963 X 10

We calculate each element of the design matrix by the formula given below. Here x_1 is the datarow$_1$ and u is the mean and σ is the variance.

$$\phi_1(x_1) = e\,\frac{-(x_1-u)^2}{2\sigma^2}$$

The above equation can also be described as:

$$\phi_1(x_1) = e^{-\frac{1}{2}(x-u_1)^T \Sigma^{-1}(x-u_1)}$$

(d) Calculating the weights

We use the Morse Penrose Inverse method simply called the Pseudo Inverse method to calculate the weights. We use the formula given as follows:

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

(e) Calculating Error

When the model is trained using the training data and the model is functioning, we have no way of knowing whether the predicted value is correct. So, to test that out we use the error function.

In this project we are using root mean squared error or $E_{RMS.}$

We calculate the E_{RMS} for the testing, training and validation set.

$$E_{RMS}=\sqrt{\frac{1}{N}((t-\hat{t})^2)}$$

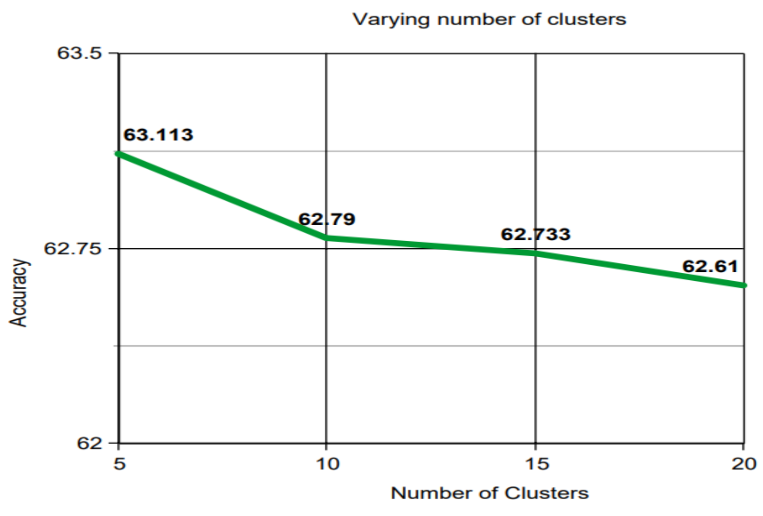
Results for Closed Form Solution with Radial Basis Function

M = 10 Lambda = 0.9

E_rms Training = 0.5494694067137841

E_rms Validation = 0.5384281741391039

E_rms Testing = 0.6279788453842932



Closed Form Solution

Why the need for Gradient Descent Solution?

We have used closed form solution or pseudo inverse method, so what is the need for gradient descent. In pseudo inverse method, we take the inverse of the covariance matrix.

This may not be possible, if the matrix in non-invertible. If that is the case, we will be unable to perform pseudo inverse method. Also calculating inverse of large matrix is very difficult and takes a lot of computations. For that reason, we use Gradient descent solution. Gradient descent is an efficient optimization algorithm that attempts to find a local or global minimum of a function.

Step 3 – Stochastic Gradient Descent Solution

The data processing step remains the same for the stochastic gradient descent solution.

The k-means Clustering, BigSigma and Calculating Training Design Matrix $\phi(X)_{TR}$, Testing Design Matrix $\varphi(X)_{TE}$ and Validation Design Matrix $\phi(X)_{VA}$ also remain the same for gradient descent solution.

The next step that is the weights calculation step completely changes for gradient descent solution.

Here are the steps for calculating the weights:

(a) Initialize weights
We randomly initialize the weights, by taking the dot product of W and 220.
Here W is the weights produced by closed form solution. We could use randomly initialize W by any value.

(b) Update weights iteratively
We update the weights iteratively i.e. for each datapoint we update all the weights.
It is as follows:

$w_1 = w_1 + \eta_{\Delta w_1}$

.

.

$w_{10} = w_{10} + \eta_{\Delta w_{10}}$

Here we are updating all the 10 weights iteratively by adding the product of the learning rate and the partial derivative of the error function. The partial derivative is as follows:

$$\Delta w_1 = \frac{\partial E}{\partial \omega_1} = \frac{1}{2}(t - \hat{t})^2$$

$$\Delta w_1 = \frac{\partial}{\partial \omega_1}\frac{1}{2}\left(t - \omega^T \phi(x)\right)^2$$

$$\Delta w_1 = \frac{\partial}{\partial \omega_1}\frac{1}{2}\Big(t - \big(w_1\phi_1(x) + \cdots + \omega_{10}\phi_{10}(x)\big)\Big)^2$$

$$\Delta w_1 = -\phi_1(x)(t - \omega^T \phi_1(x))$$

$$\Delta w_{10} = -\phi_{10}(x)(t - \omega^T \phi_{10}(x))$$

So, the updated weights $w_1 \ldots w_{10}$ will be as follows:

$w_1 = w_1 + \eta(-\phi_1(x)(t - \omega^T \phi_1(x)))$

.

.

$w_{10} = w_{10} + \eta(-\phi_{10}(x)(t - \omega^T \phi_{10}(x)))$

(c) Calculating error

$$E_{RMS} = \sqrt{\frac{1}{N}((t - \hat{t})^2)}$$

The calculation of the root mean squared error is same as the closed form solution.

Results for Stochastic Gradient Descent Solution:

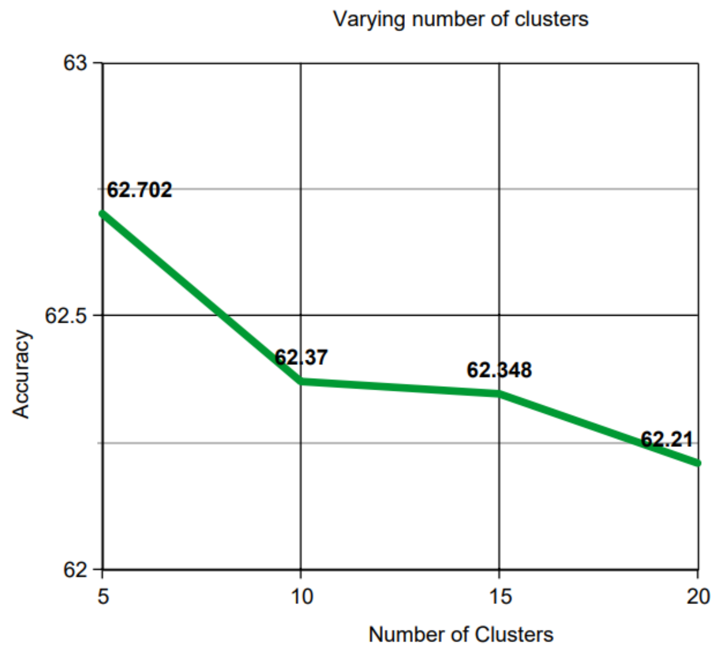
M = 10 Lambda = 0.0001 eta=0.01

E_rms Training = 0.54964

E_rms Validation = 0.53846

E_rms Testing = 0.62372

The accuracy of both the closed form solution and stochastic gradient descent solution for varying number of clusters is similar as shown in the graph below.



Gradient Descent Solution

References:

1- https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220

2- https://en.wikipedia.org/wiki/Gradient_descent

3- https://machinelearningmastery.com/linear-regression-for-machine-learning/