CSE 574: Project 1: Linear Regression

Aniket Sadanand Mayakal, UBIT: amayakal , UBID : 50133703

***Abstract*—This report explains two models for linear regression: Maximum Likelihood Estimation (closed form solution) and Stochastic Gradient Descent. It explains the intuition to select the required parameters and relation between parameters and Error.**

***Keywords*—Maximum Likelihood Estimation, Stochastic Gradient Descent, Basis Functions, Design Matrix, Root mean square error.**

I. Introduction

Supervised learning is an important topic in machine learning. We are trying to predict an outcome y on some input x, which is a vector of D inputs based on previous known inputs(x_i is a vector of D i^{th} case inputs and output (y_i). Thus we are not explicitly programming the computer but making it to learn the pattern in the inputs.

Linear Regression is finding the best fitting linear function through the output points. We would cover two approaches to linear regression: Maximum Likelihood Estimation (closed form solution) and Stochastic Gradient Descent in this report.

Suppose we also don't expect the output y to be a linear function of input x. Hence, linear function won't work in this situation. So we allow non-linear functions of x by introducing the concept of basis functions.

II. Basis functions

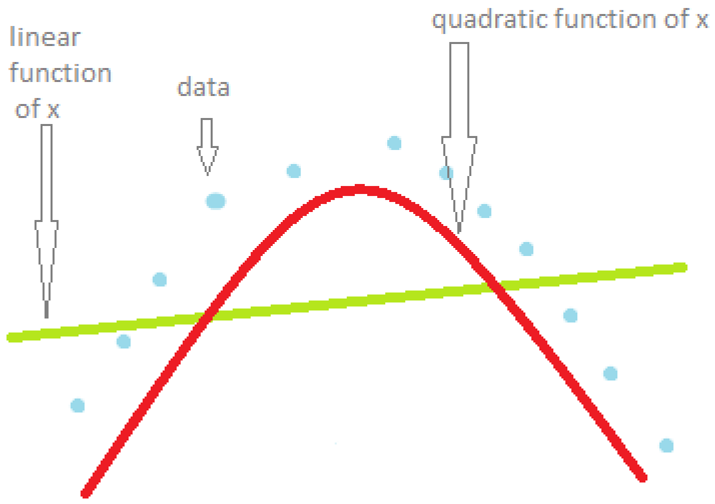

Fig. 1. Need for basis function

For a given set of data, sometimes polynomial functions of input x fits better than linear functions. As shown in fig 1, quadratic function is better suited for given data than linear function. Thus, in this case we use Basis functions.

Basis functions converts the non-linear function of x into linear function in terms of ϕ. That means, a non-linear function of x is linear in some feature space as function of ϕ. Thus , a function:

$$y(x) = ax^3 + bx^2 + cx + 1 \tag{1}$$

can be written as:

$$y(x, w) = w_0 + w_1\phi(x) \tag{2}$$

In matrix notations, this could be written as:

$$Y(x, w) = W^T\phi(x) \tag{3}$$

where, $\phi(x)$ is a vector of M basis functions with $\phi_0(x)$ = 1.

In this project, we use Gaussian basis function.

A. *Gaussian basis function*

Gaussian basis function is defined as:

$$\phi_j(x) = exp(-\frac{(x - \mu_j)^2}{2s^2}) \tag{4}$$

where μ_j is a vector in feature space and s is isotropic spatial scale.

B. *Design Matrix*

When we consider N input cases, we get Φ which is a N x M matrix with each row as $\phi(x_i)$ i = (1,2...N). This is know as design matrix.

$$\Phi = \begin{bmatrix} 1 & \phi_1(x_1) & ..\phi_{M-1}(x_1) \\ 1 & \phi_1(x_2) & ..\phi_{M-1}(x_2) \\ (\ 1 & \phi_1(x_3) & ..\phi_{M-1}(x_3) \\ . & . & . \\ 1 & \phi_1(x_N) & ..\phi_{M-1}(x_{M-1}) \end{bmatrix} \tag{5}$$

III. Maximum Likelyhood Estimation

Assume a gaussian function with mean zero and variance σ^2.
Thus the likelihood function for the parameters w and σ is the joint probability of all the outputs in the training set as a function of w and σ:

$$L(w, \sigma) = P(y_1, ..., y_N | x_1, ..., x_N, w, \sigma) \tag{6}$$

$$L(w, \sigma) = \prod_{i=1}^{N} N(y_i | W^T\phi(x_i), \sigma^2) \tag{7}$$

where N(y— μ, σ^2) is the density for y under a normal distribution with mean μ and variance σ^2.

Thus to maximize the likelihood function, we can also maximize the log of likelihood function and thus ignoring the parameters which are independent on w and σ, we need to maximize below function:

$$logL(w,\sigma) = -Nlog(\sigma) - \frac{1}{\sigma^2}\sum_{i=1}^{N}(y_i - W^T\phi(X_i))^2 \quad (8)$$

thus log likelihood function regardless of what σ might be, the maximum likelihood estimate of w is the value that minimizes the sum of squared errors over training cases. Considering design matrix $\Phi_{ij} = \phi_j(x_i)$. We can now write the sum of squared errors on training cases as

$$||Y - \Phi_w||^2 = (Y - \Phi_w)^T(Y - \Phi_w) \quad (9)$$

This is minimized for the value of w where its gradient is zero, which is where

$$-2\Phi^T(y - \Phi_w) = 0 \quad (10)$$

Solving this, the least squares estimate of w

$$W = (\Phi^T\Phi)^{-1}\Phi^T Y \quad (11)$$

A. Experimental results

Steps in experiment was as follows:

1) Extract data from Microsoft LETOR dataset(69623 inputs with each 46 features) and divide data as 80
2) Choose s (variance).
3) Choose M (feature space dimensions).
4) Choose lambda (for regularization).
5) Calculate μ_{cfs} M-1 $\times$ 46(features in input dataset).
6) Calculate Φ for training dataset.
7) Calculate W using eq.(11).
8) Calculate Φ for validation dataset.
9) Calculate root mean square error.
10) Go to step 1 if not acceptable and repeat till root mean square error is minimum.

1) Selecting parameters - s: Our goal is to minimize Error. Thus we need to choose s such that error is minimum. Let λ be zero for present time.
We find different combinations of m and s and plot a surface to find out minimum error. Thus from fig 2 we can conclude that as M and s increases error value is decreasing. So we extend our procedure by increasing the value of m even further. From fig 3, the error value is still decreasing, so we extend our procedure but this time keeping s as constant to save computation time. we will compare s with other values later. Thus by looking at validation data, we see that at M = 120, the change in error is not that significant. Thus we select M as 120. Now lets go back to select s. We check for error values in the range M = 120 - 125 and s = 1 - 1.1 Thus figure 5 shows that even though there is decrease in error value as u increase s, the decrease is very minimal, of order 10^{-3}. Thus we may ignore further values of s and select s as 1.
Now that we have selected s and M, lets consider λ. Fig 6 shows that as λ approaches 0.00001 from 0.0000000001, there is very little change in error. Fig 7 shows there is an increase in error as we increase λ. Thus we choose lowest possible λ.

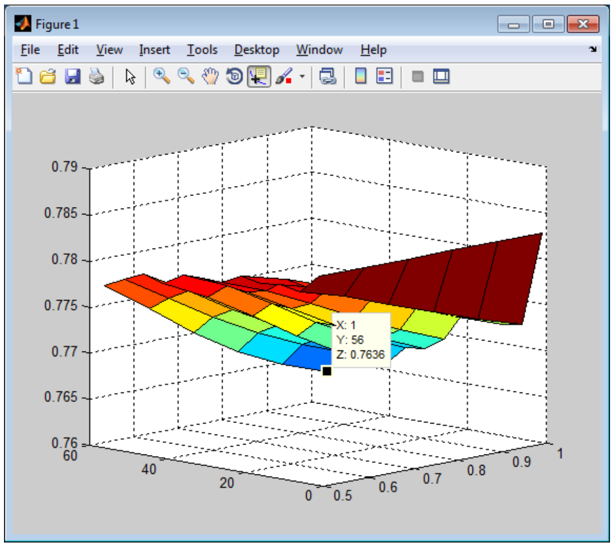

Fig. 2. on s ranges from 0.5 - 1 and M ranges from 0 - 60 and Z-axis shows the corresponding root mean square for corresponding values of s and m

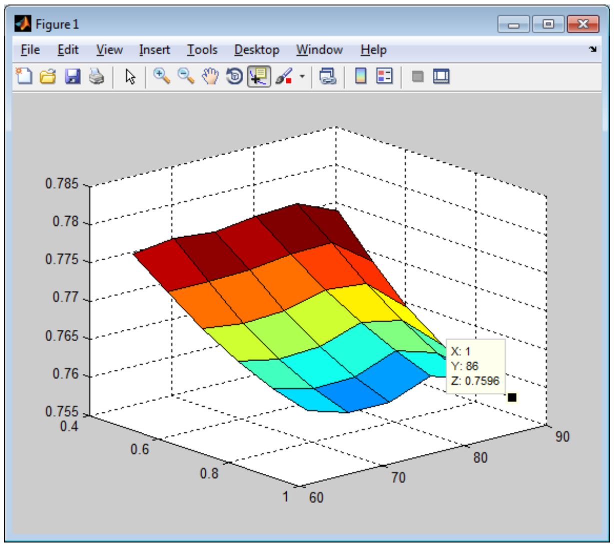

Fig. 3. s ranges from 0.5 - 1 and M ranges from 60 - 90 and Z-axis shows the corresponding root mean square for corresponding values of s and m

2) Test: Now that we have trained our model, we test it on Test data. Root mean square error on test data obtained was 0.88.

IV. Stochastic Gradient Descent

Stochastic Gradient Descent is just an another way to find W. In this model, we start with random values of W and then adjust the values based on every input. Thus we tune W on every input. This is used when we have large set of data.
General form of Stochastic gradient descent is given as

$$W^{(\tau+1)} = W^{(\tau)} + \eta(y_n - W^{(\tau)T}\phi_n) \quad (12)$$

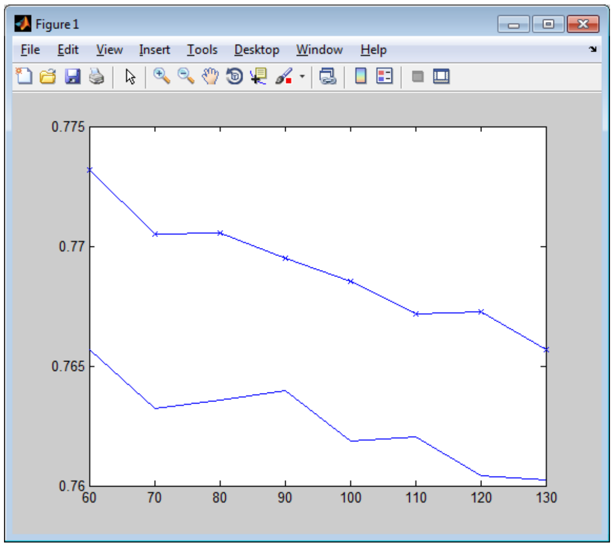

Fig. 4. s =1 M ranges from 60 - 130 and Z-axis shows the corresponding root mean square for corresponding values of s and m. Line '—-x' is training data and solid line is validation data curves.

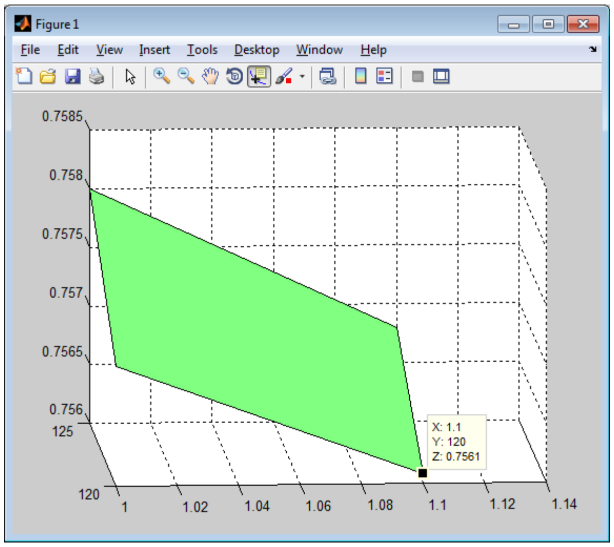

Fig. 5. s ranges from 1 to 1.1 and M ranges from 120 - 125 and Z-axis shows the corresponding root mean square for corresponding values of s and m.

where η is the learning rate and ϕn is the basis function for n^{th} input - vector of dimension M.

A. *Experimental results*

We follow the same steps in Stochastic gradient descent as in Closed form solution, only difference is how we calculate our W. We ran a set of tests for value of η and found that for η = 1, 0.5, 0.05,0.005 the error was increasing and thus the value of eta needed to be decreased. At 0.0005, the error started reducing and thus we select η as 0.0005. Figure 8 shows that around M = 20, the error for validation dataset is minimum and thus we select M as 20.
Having s as 1 and M as 20, we train the model for different models of λ and check for minimum error. As we go from 0.001 to 1, we see that error starts increasing. Thus, we select λ with least error. Thus λ = 0.01.

1) Test: Now that we have trained our model, we test it on Test data. Root mean square error on test data obtained was 0.89.

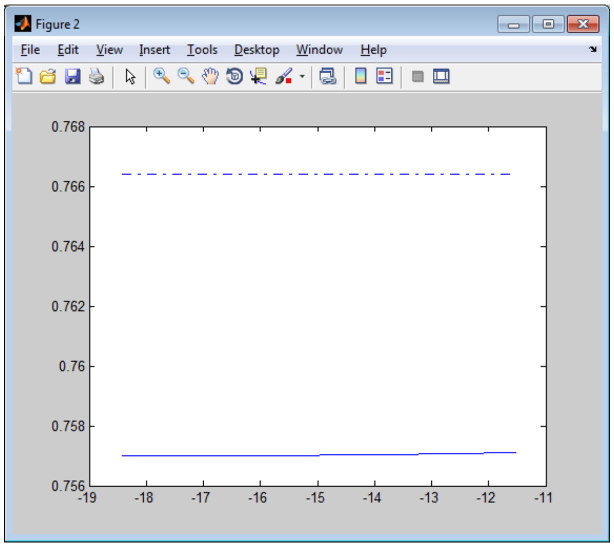

Fig. 6. Plot of ln λ vs root mean square error.

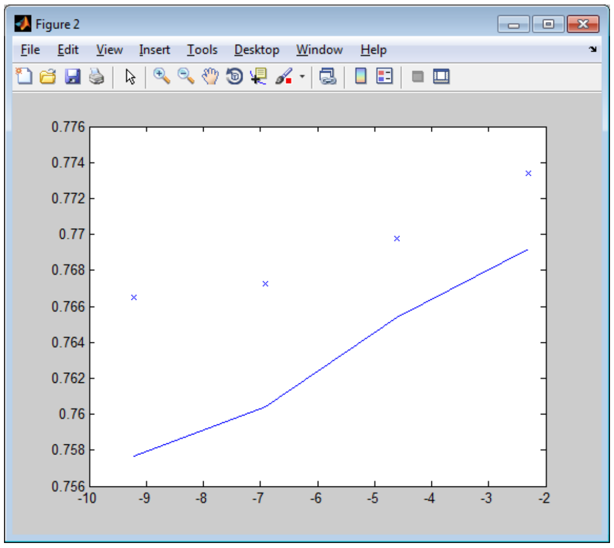

Fig. 7. Plot of ln λ vs root mean square error.

V. Conclusion

Complete project took 325 seconds to execute. Closed form solution took 156 seconds to train and 19 seconds to test, on the other hand, Stochastic gradient descent took 142 seconds to train and 2.309 seconds to test. Both models gave very

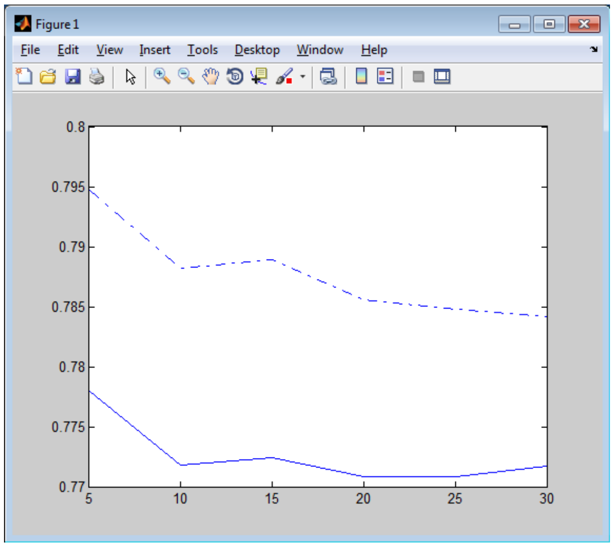

Fig. 8. Plot of M vs root mean square error. at s = 1. '–' represents training dataset and solid line represents validation dataset

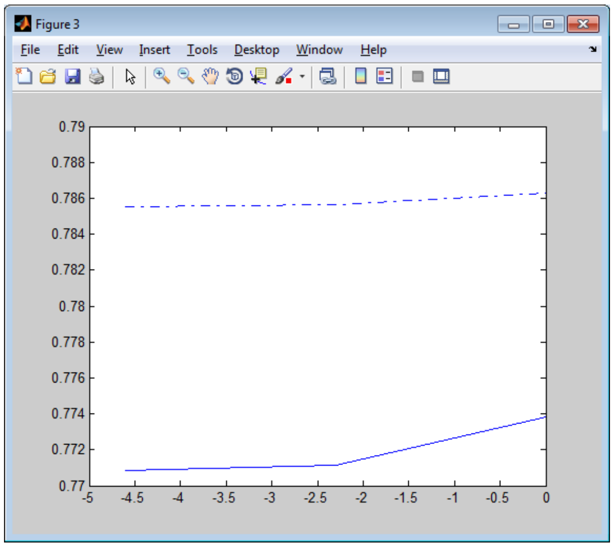

Fig. 9. Plot of ln λ vs root mean square error.

close root mean square error (E = 0.88 for MLE and 0.89 for SGD) for and complexity of Stochastic gradient descent is much lower than Maximum Likelyihood Estimation model.

REFERENCES

[1] http://www.utstat.utoronto.ca/ radford/sta414.S11/week1b.pdf

[2] http://www.youtube.com/user/mathematicalmonk