CSE/ECE-471: Statistical Methods in AI **Spring 2020**

Linear Regression

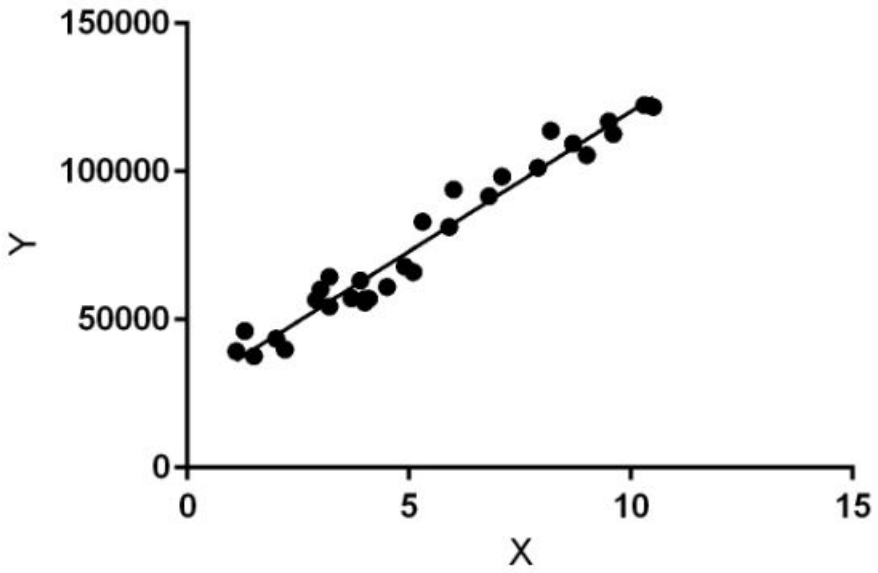
Prepared by: Nabeel Kabeer(2019900060), Sanidhay Arora (20171180), Lokesh Singh(2019201051)

In this note we discuss Linear Regression.

1 Introduction

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.



In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

2 Formulation

The goal of linear regression is to build a system that can take a vector $x \in \mathbb{R}^n$ and predict the value of scalar $y \in \mathbb{R}$ as it's output.The output of linear regression is a linear function of the input. We define the output to be :

$$y = w^T x + \epsilon \tag{1}$$

where $w \in \mathbb{R}^n$ is the vector of parameters or can also be thought of as weights which tells how each feature affects the prediction. ϵ refers to the residual or error incurred by the model.For most real data the noise usually follows a Gaussian distribution $\epsilon \sim \mathcal{N}(w^T x_i, \sigma^2)$

For a given observed of N input output pairs $(x_1, y_1), (x_2, y_2)...(x_N, y_N)$, each can be represented as :

$$y_i = w^T x_i + \epsilon_i \tag{2}$$

The best model will be the one which has the least total error for the entire set, defined as:

$$totalerror = \sum_{i=1}^{N} \epsilon_i \tag{3}$$

Each residual error can be defined as :

$$\epsilon_i = (w^T x_i - y_i)^2 \tag{4}$$

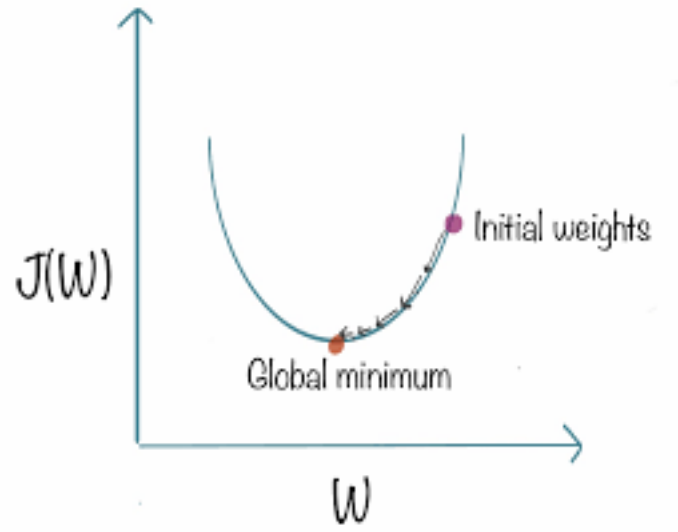
which follows from the assumption that they follows a Gaussian distribution, hence our total error is defined as :

$$totalerror = J(w) = \frac{1}{2} \sum_{i=1}^{N} (w^T x_i - y_i)^2 \tag{5}$$

The prime objective of the model is to reduce the total error on observed data.This total error can be called as a cost function of the model which depends on the chosen parameters.

3 Gradient Descent

We want to chose w such that it minimizes $J(w)$.To achieve this let's use a search algorithm which starts with an initial guess and eventually reaches the minima. One such algorithm that we can use is the **Gradient Descent**. Gradient Descent exploits the fact that applying a gradient operator (∇) on a scalar function results in a vector valued function which points in the direction of greatest increase.Hence by updating the parameters w, such that $J(w)$ moves in the opposite direction of maximum increase will eventually take us to the global minima (since $J(w)$ is a convex function)



Let's compute the Gradient of our cost function $J(w)$:

$$J(w) = \frac{1}{2}\sum_{i=1}^{N}(w^T x_i - y_i)^2$$
$$\implies J'(w) = \frac{1}{2})\sum_{i=1}^{N} 2x_i(w^T x_i - y_i)$$
$$\implies J'(w) = \sum_{i=1}^{N} x_i(w^T x_i - y_i)$$

Now we can devise an algorithm which can be used to find w which reaches the minimum value for $J(w)$

3.1 Algorithm for Gradient Descent

Algorithm 1: Gradient Descent Algorithm

1 Choose a learning rate $\alpha \in \mathbb{R}^+$
2 Choose a random point $w \in \mathbb{R}^n$
3 Repeat step 4 till convergence
4 Set $w := w - \alpha \sum_{i=1}^{N} x_i(w^T x_i - y_i)$

This algorithm is also called **batch gradient descent**.One thing that we can notice is for a single updation of the parameters our batch gradient descent algorithm has to scan through the entire training set, this operation can be costly for large values of N.One solution to this problem is to given by a method called **stochastic gradient descent**.

3.2 Convergence Proof

If $J(w)$ is the cost function ,then according to Taylor series , for a small value h, $J(w+h)$ can be approximated as :

$$J(w+h) \approx J(w) + hJ'(w) \tag{6}$$

according to the weight updation ,w_t at time t updates to w_{t+1} which is given as :

$$w_{t+1} = w_t - \alpha J'(w) \tag{7}$$

Therefore the cost function can be represented as:

$$J(w_{t+1}) \approx J(w_t - \alpha J'(w_t))$$
$$\implies J(w_{t+1}) \approx J(w_t) - \alpha J'(w_t)^2$$
$$\implies J(w_{t+1}) < J(w_t)$$

From this $J(w_{t+1})$ will be always less than the cost from the previous update $J(w_t)$ given that $\alpha > 0$ and $\alpha J'(w)$ is a small quantity

3.3 Stochastic gradient descent

In this algorithm, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single training example only.

Algorithm 2: Stochastic Gradient Descent Algorithm

```
Loop
    Randomly select m subset of training examples
    for i ← 0 to m do
        w_j := w_j − α(w_j x_i − y_i)x_i   ∀j
    end
```

Stochastic gradient descent can start making progress right away, and continues to make progress with each example it looks at. Often, stochastic gradient descent gets w "close" to the minimum much faster than batch gradient descent. (Note however that it may never "converge" to the minimum, and the parameters w will keep oscillating around the minimum of $J(w)$; but in practice most of the values near the minimum will be reasonably good approximations to the true minimum.For these reasons, particularly when the training set is large, stochastic gradient descent is often preferred over batch gradient descent.

4 Direct/Normal Form

We will write $J(w)$ in a matrix notation, given by :

$$J(W) = \frac{1}{2}(AW - Y)^T(AW - Y)$$

$$A = \begin{bmatrix} \dots & {X_1}^T & \dots \\ \dots & {X_2}^T & \dots \\ & \vdots & \\ \dots & {X_N}^T & \dots \end{bmatrix}_{N \times d}$$

$$W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{d-1} \end{bmatrix}_{d \times 1}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N \times 1}$$

We can find an exact solution to this problem if we can compute $\frac{\partial J}{\partial W}$ and equate it to 0 to find the value of parameters w at which we can reach the global minima:

4.1 Computing Gradient of $J(w)$

$$\begin{aligned} J(W) &= \frac{1}{2}(AW - Y)^T(AW - Y) \\ \implies \frac{\partial J}{\partial W} &= \frac{1}{2}\frac{\partial}{\partial W}[(AW^T - Y)^T(AW - Y)] \\ \implies \frac{\partial J}{\partial W} &= \frac{1}{2}\frac{\partial}{\partial W}[(AW)^T AW - (AW)^T Y - Y^T AW - Y^T Y] \end{aligned}$$

($P = AW$ and $Q = Y$ are both vectors.Using $P^TQ = Q^TP$)

$$\begin{aligned}
\implies \frac{\partial J}{\partial W} &= \frac{1}{2}\frac{\partial}{\partial W}[(AW)^TAW - 2(AW)^TY + Y^TY] \\
\implies \frac{\partial J}{\partial W} &= \frac{1}{2}[2A^TAW - 2A^TY] \\
\implies \frac{\partial J}{\partial W} &= A^TAW - A^TY
\end{aligned}$$

4.2 Finding W with least cost

The least value of $J(W)$ is achieved were:

$$\begin{aligned}
\frac{\partial J}{\partial W} &= 0 \\
\implies A^TAW - A^TY &= 0 \\
\implies W &= (A^TA)^{-1}A^TY
\end{aligned}$$

In cases were A is a non-square matrix one can use **pseudoinverse** to calculate W, given by :

$$W = (A^TA)^+A^TY$$

5 Comparision Normal vs SGD

In both gradient descent (GD) and stochastic gradient descent (SGD), you update a set of parameters in an iterative manner to minimize an error function.

While in GD, you have to run through ALL the samples in your training set to do a single update for a parameter in a particular iteration, in SGD, on the other hand, you use ONLY ONE or SUBSET of training sample from your training set to do the update for a parameter in a particular iteration. If you use SUBSET, it is called Minibatch Stochastic gradient Descent.

Thus, if the number of training samples are large, in fact very large, then using gradient descent may take too long because in every iteration when you are updating the values of the parameters, you are running through the complete training set. On the other hand, using SGD will be faster because you use only one training sample and it starts improving itself right away from the first sample.

SGD often converges much faster compared to GD but the error function is not as well minimized as in the case of GD. Often in most cases, the close approximation that you get in SGD for the parameter values are enough because they reach the optimal values and keep oscillating there.

6 Regularization

Regularization Term = $\lambda \sum_{i=1}^{N} || w_i ||^2$

$$J(w) = \frac{1}{2}\sum_{i=1}^{N}(w^Tx_i - y_i)^2 + \lambda \sum_{i=1}^{N} || w_i ||^2$$

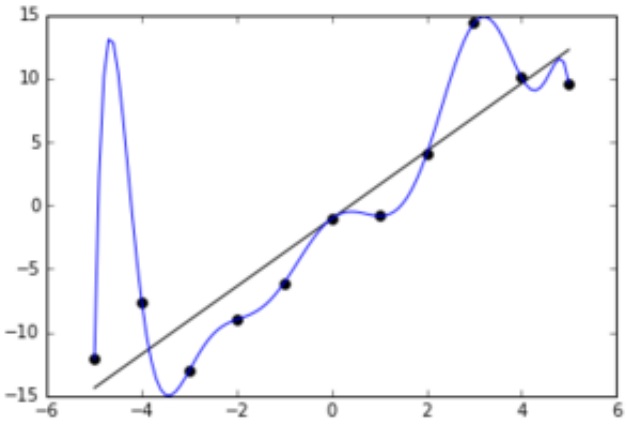
λ is a parameter which controls the importance of the regularization term. $\sum_{i=1}^{N} || w_i ||^2$ is typically chosen to impose a penalty on the complexity of $(w^T x_i - y_i)^2$. As w_i increases it penalises the result of J(w) so we try to reduce the Regularization term to make maximum $w_i = 0$.

A theoretical justification for regularization is that it attempts to impose Occam's razor on the solution. From a Bayesian point of view, many regularization techniques correspond to imposing certain prior distributions on model parameters. Regularization can serve multiple purposes, including learning simpler models, inducing models to be sparse and introducing group structure into the learning problem.

7 Overfitting

Overfitting is a modeling error that occurs when a function is too closely fit to a limited set of data points. Overfitting the model generally takes the form of making an overly complex model to explain idiosyncrasies in the data under study.

In reality, the data often studied has some degree of error or random noise within it. Thus, attempting to make the model conform too closely to slightly inaccurate data can infect the model with substantial errors and reduce its predictive power.



References

[1] http://cs229.stanford.edu/notes/cs229-notes1.pdf

[2] Wikipedia

[3] GeeksforGeeks