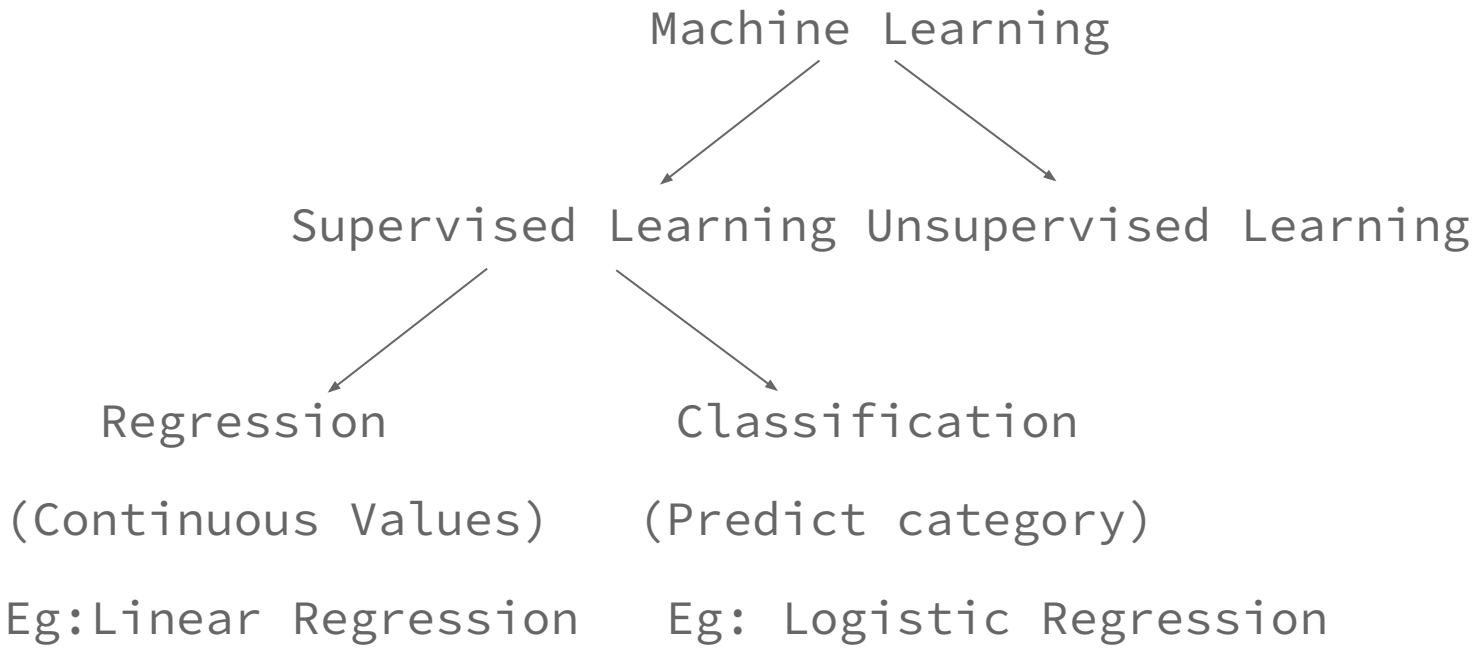
Machine Learning

[Class-2]

- Linear Regression
- Logistic Regression

- Aayush shandilya (IT)
- Abhinav Anand (IT)

ReCAp


Machine Learning
Supervised Learning
Unsupervised Learning
Regression
(Continuous Values)
Eg:Linear Regression
Classification
(Predict category)
Eg: Logistic Regression

Linear Regression

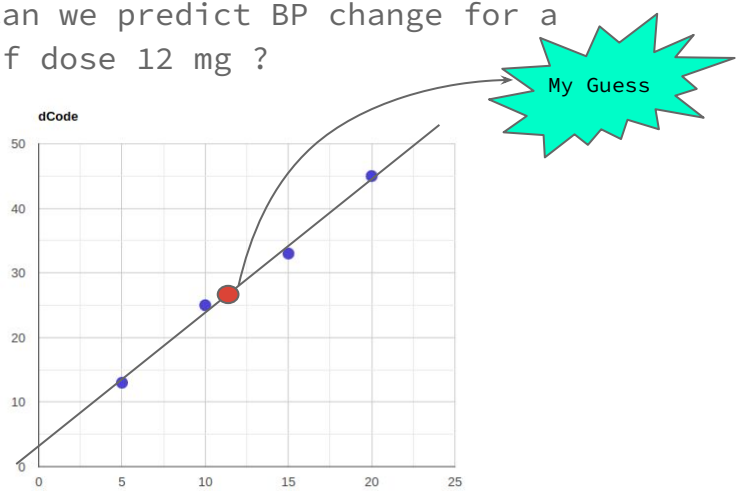
Let’s Start !

Let's Guess ..

Let's say we have Blood pressure change of patients against various dosage of a drug.

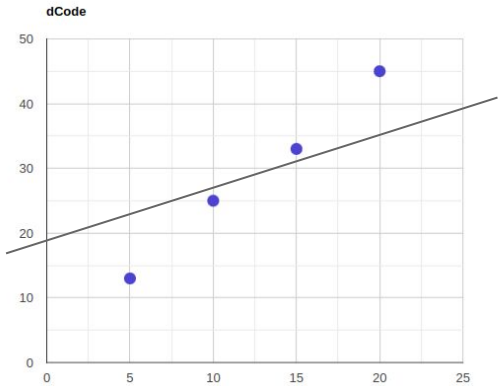
Change in Blood Pressure	Dosage (mg)
13	5
25	10
33	15
45	20
??	12

Can we predict BP change for a of dose 12 mg ?

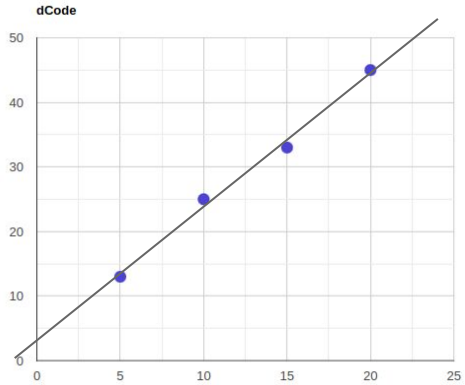


So, this process of finding the best-fit line to fit the data in-order to predict is **Linear Regression**

Which line is better ?



(A)



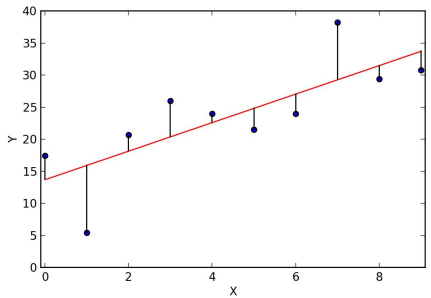
(B)

So a line with least difference from each points would be a better one or **better fit.**

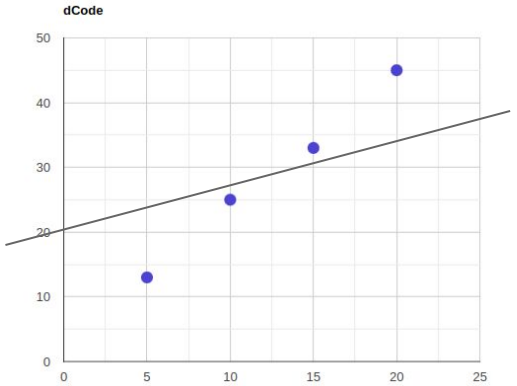
Let's define our aim

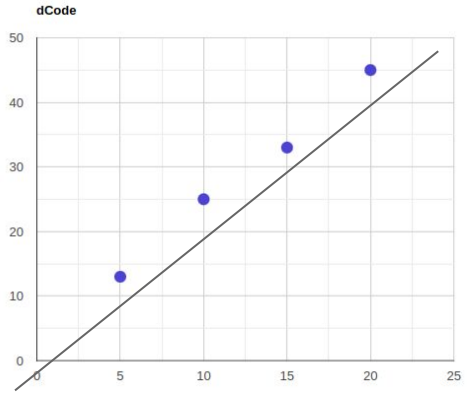
Let's define a cost function whose minimum value corresponds to best fit line.

$$\text{Cost}_{\text{points}} = \sum_{n=1}^{M} \left(Y_{\text{Predicted}} - Y_{\text{Actual}} \right)$$,assuming there are M data



Again , Which line is better ?



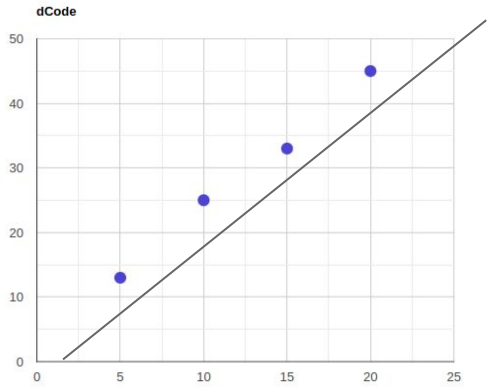


(A) (B)

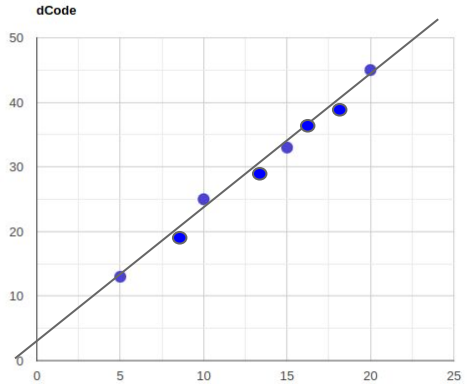
Acc. to our error function (A) is better as positive negative difference cancels out. Also it does not punishes for larger errors.

$$\text{Cost}(J) = \sum_{n=1}^{M} (Y_{\text{Predicted}} - Y_{\text{Actual}})^2$$

Again , Which line is better ?



(A)



(B)

No. of data point should also be considered hence let's take mean error

$$J = \frac{\sum_{n=1}^{M} (Y_{Predicted} - Y_{Actual})^2}{M}$$

Why Squared ?

The cost function of a linear regression is root mean squared error or mean squared error. They are both the same; just we square it so that we don’t get negative values.

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

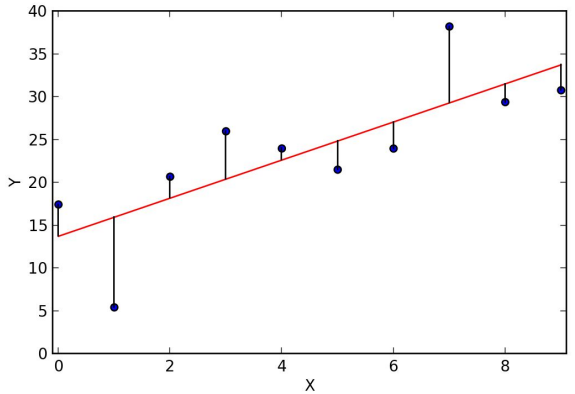
Why Divide by 2*M ?

It is because when you take the derivative of the cost function, that is used in updating the parameters during gradient descent, that 2 in the power get cancelled with the ½ multiplier, thus the derivation is cleaner.

OUR AIM

Aim : Get BEST FIT line or one with minimum error or cost.

$$\text{Cost} = \frac{\sum_{n=1}^{M} (Y_{Predicted} - Y_{Actual})^2}{2.M}$$



We need to find a line with minimum cost. But how ?

Let's find the one

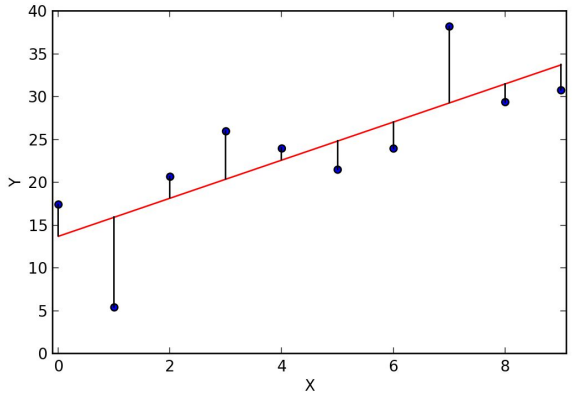
Let's say, $Y_{predict} = \theta_0 + \theta_1.X$

Let's say, $\theta_0 = 0$ for simplicity

Hence, $Y_{predict} = \theta_1.X$

$$J = \frac{\sum_{i=1}^{M} (Y_{Predicted} - Y_{Actual})^2}{2.M}$$

$$J(\theta_1) = \frac{\sum_{i=1}^{M} (Y_i - \theta_1.X_i)^2}{2.M}$$



Aim: Find such θ_1 such that J is minimum.

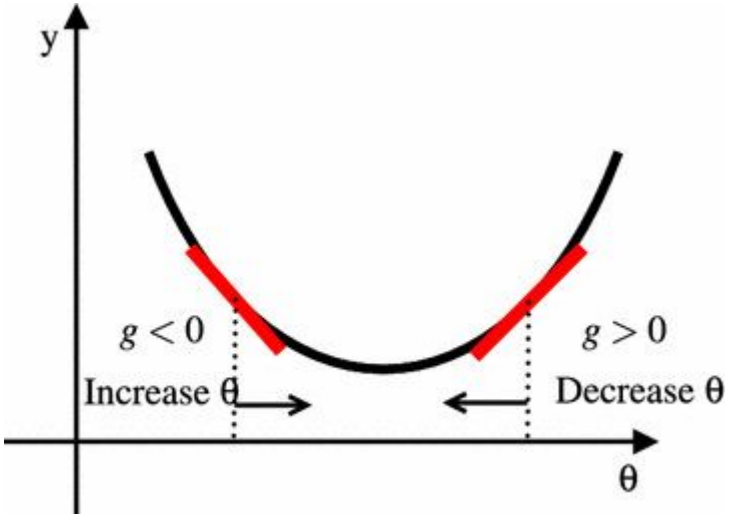
Gradient Descent Algorithm

1. Select a random θ_i.
2. Update the value of θ by:

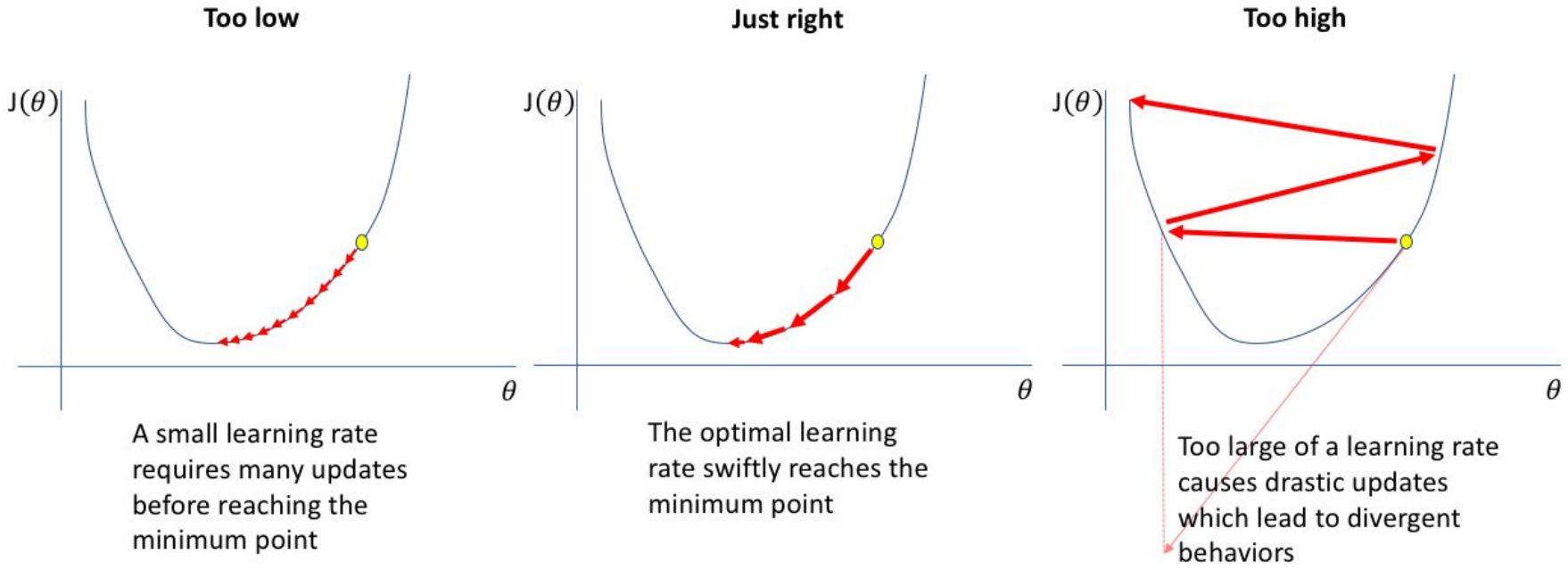
$$\theta_{i+1} = \theta_i - \alpha . \frac{\partial J(\theta)}{\partial \theta}\Big|_{\theta_i}$$

3. Repeat Step 2 until minimum is achieved.

Here α is called learning rate.



Importance of learning rate



What if there are multiple attributes

Now equation of line

$$h(x) = \theta_0 + \theta_1.X_1 + \theta_2.X_2 + \theta_3.X_3 + \theta_4.X_4 + \ldots\ldots\ldots + \theta_n.X_n$$

Or $h(x) = \theta^T.X$ where $\theta=[\theta_0,\theta_1,\theta_2,...,\theta_n]$ and $X=[1,X_1,X_2,...,X_n]$

Therefore,

$$J(\theta) = \sum_{i=1}^{M} \frac{(Y_i - h(x^i))^2}{2.M}$$

Derivative wrt to a vector is called gradient and represented by ∇.

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_i} \end{pmatrix}$$

Gradient descent for multiple attributes

hence , updating θ would be

$$\theta_{i+1} = \theta_i - \alpha . \nabla_\theta J(\theta)\Big|_{\theta_i}$$

$$\begin{pmatrix} \theta_0^{i+1} \\ \theta_1^{i+1} \\ \vdots \\ \theta_n^{i+1} \end{pmatrix} = \begin{pmatrix} \theta_0^{i} \\ \theta_1^{i} \\ \vdots \\ \theta_n^{i} \end{pmatrix} - \alpha . \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_i} \end{pmatrix}$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

...

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} [h_\Theta(x_i) - y_i]^2$$

$h_\Theta(x_i)$: Predicted Value; y_i: True Value

Gradient Descent

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1)$$

α: Learning Rate

Now,

$$\frac{\partial}{\partial \Theta} J_\Theta = \frac{\partial}{\partial \Theta} \frac{1}{2m} \sum_{i=1}^{m} [h_\Theta(x_i) - y]^2$$

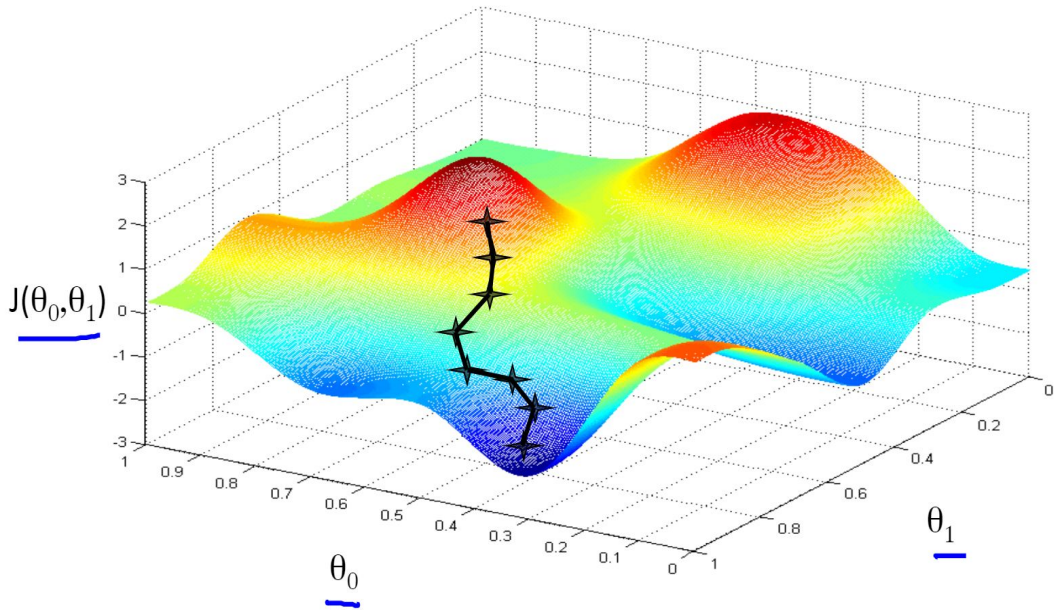
$$= \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x_i) - y) \frac{\partial}{\partial \Theta_j} (\Theta x_i - y)$$

$$= \frac{1}{m} (h_\Theta(x_i) - y) x_i$$

Therefore,

$$\Theta_j := \Theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} [(h_\Theta(x_i) - y) x_i]$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$



Gradient descent for linear regression

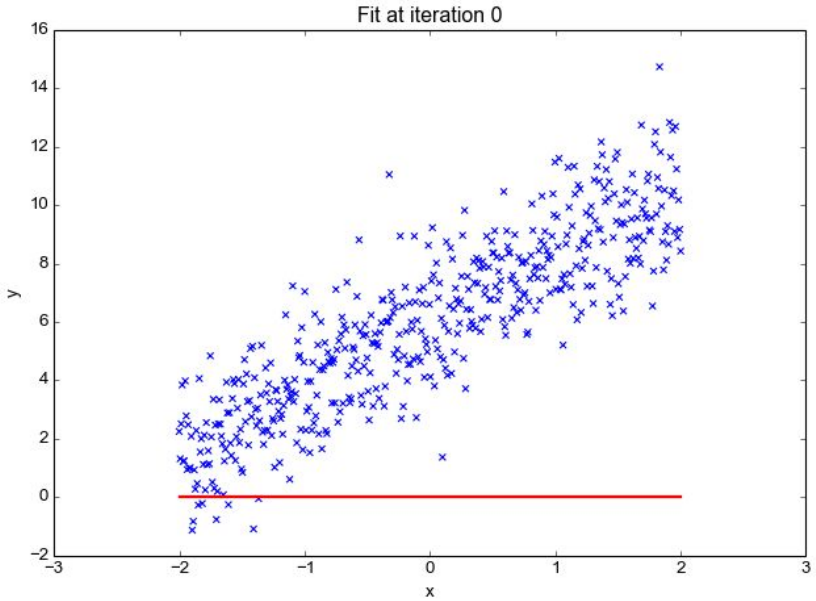
Repeat until convergence{

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x^{(i)}$$

}

Update θ_0 and θ_1 simultaneously



Linear Regression

Advantage :

- Useful if need to predict in very low latency

Issue :

- Too simple, might not fit on data cleanly or perform poorly.

LoGISTIC REGRESSION

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

TYPES OF LOGISTIC REGRESSION

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types –

Binary or Binomial

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

Multinomial

In such a kind of classification, dependent variable can have 3 or more possible *unordered* types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

Intuition

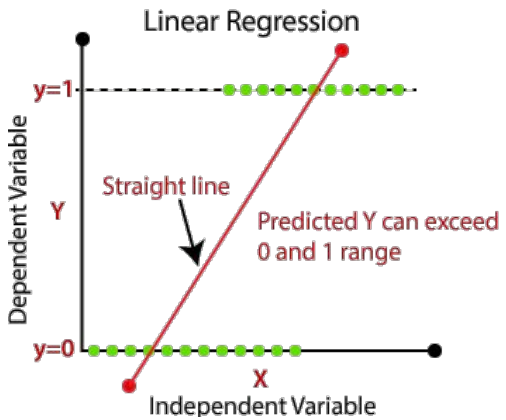
Linear :

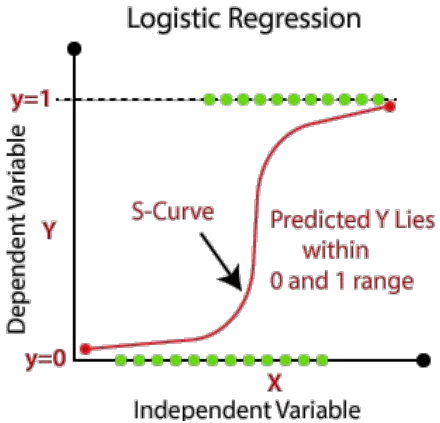
$Y = mx + c$

Logistic :

$\log(y/(1-y)) = mx + c$

$Y = 1/(1+e^{-z})$ where $z = mx + c$

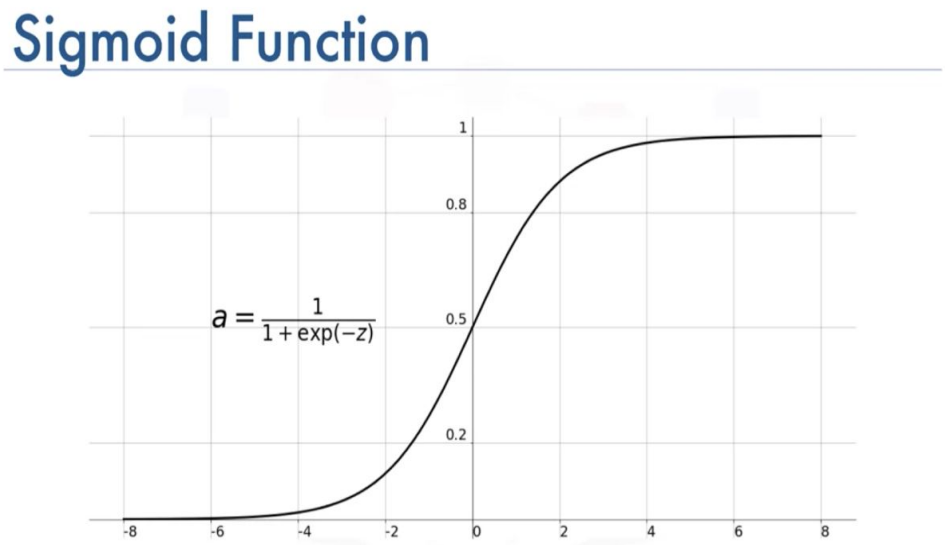




Sigmoid Function

hø(x)=g(ØTx)where0≤hø≤1

Here, g is the logistic or sigmoid function which can be given as follows –



Logistic Regression Model

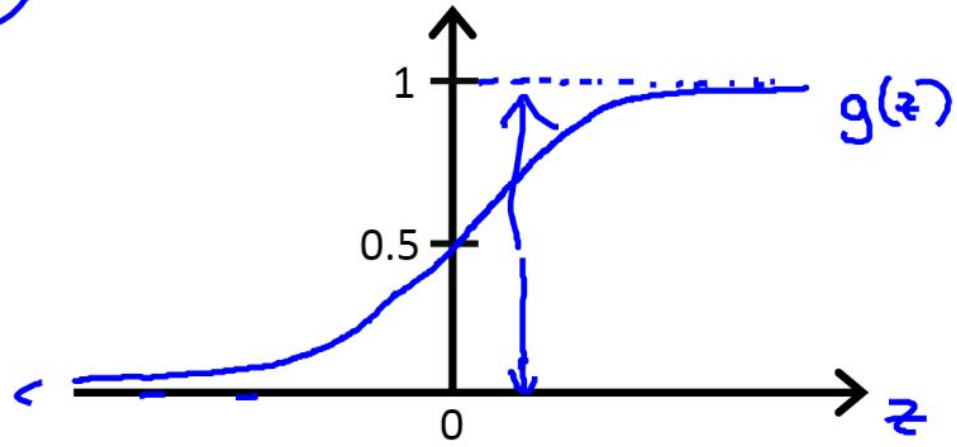
Want $0 \le h_\theta(x) \le 1$

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$z = \theta^T x$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

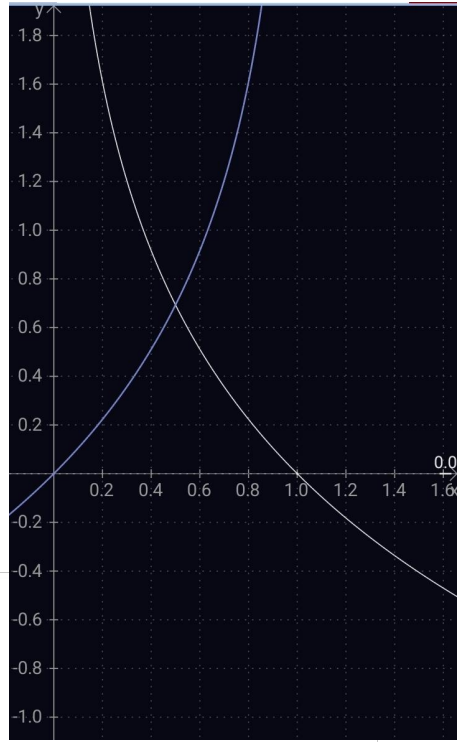


- Sigmoid function
- Logistic function

Parameters θ.

Cost Function

```
Cost(hθ(x), y) = { -log(hθ(x)) if y = 1;
                 { -log(1 - hθ(x)) if y = 0;
```

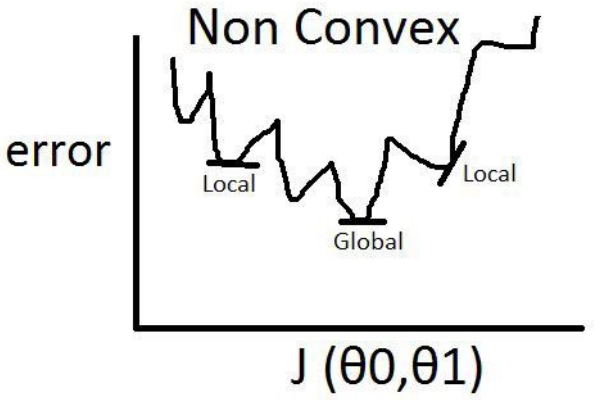


Logistic regression cost function

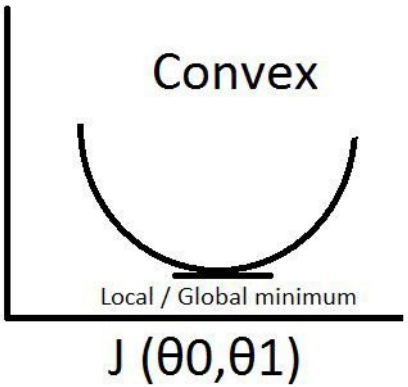
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

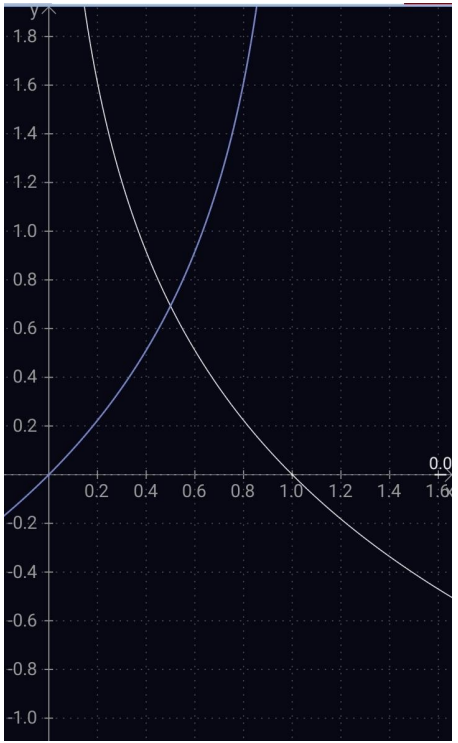
Why Log ?



$$h(X) = \frac{1}{1 + e^{-(\theta 0 + \theta 1 * X)}}$$



$$h(X) = \theta 0 + \theta 1 * X$$



Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$\}$ (simultaneously update all θ_j)

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

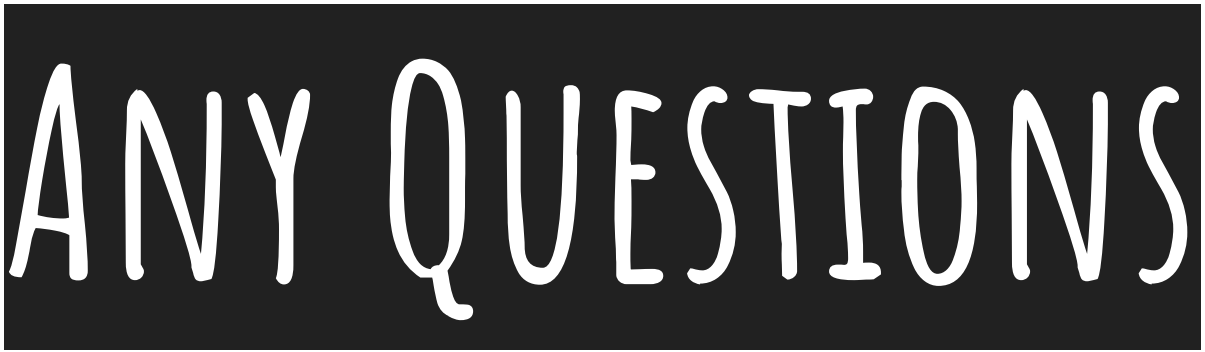
Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$\}$

For more Information : Visit


Any Questions