ML Algorithms →

1) Simple Linear Regression –

∴ (i) It is defined as type of Regression algorithms that models, relationship between a dependent variable and a single independent variable

$$y = \beta_0 + \beta_1 X + \varepsilon$$

→ y → predicted value of the dependent variable (y) for any given value of independent variable X.

→ $\beta_0$ is intercept, predicted value of y when X = 0

→ $\beta_1$ is Regression coefficient –

↳ i.e How much we expect y to change as x increases.

→ X → Independent variable

↓

(variable we expect is influencing y)

→ ε → Error –

Eg → ①

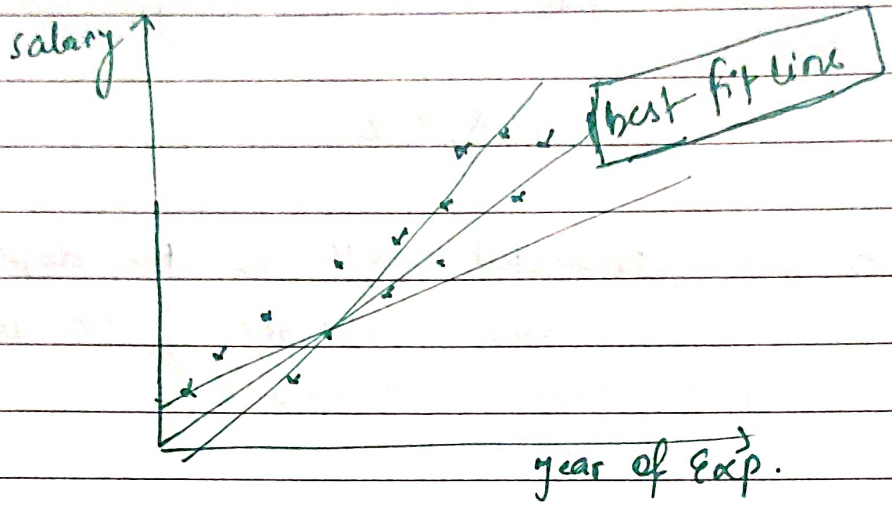
| Number of Room (← independent variable) | price (→ dependent variable) |
|---|---|
| 2 | ₹400 |
| 3 | ₹600 |
| 1 | ₹200 |
| 4 | ₹800 |
| 5 | ₹1000 |

* → Dependent variable must be continuous / Real

Consider

ML model → trained based on years & salary

predict → salary based on input year




⇒ Simple Linear Regression :- Based on training dataset a linear line (predicted) will be drawn w.r.to axes such that the error i.e dist b/w point (co-ordinate on line - predicted) is minimal with respective of actual data at the same point -

→ distance b/w actual point and predicted point is nothing it's Error, or MSE - in case of Linear Regression

$$y = mx + c$$

when coming to predicted line (co-ordinate) with respect to actual distribution of data

So, the best line always represent the distance between predicted and actual point at co-ordinate is minimum and to minimize the error we use cost function or MSE (Mean Square Error) in case of simple linear Regression.

iii) Cost function :- In case of comparing different lines and their predicted x and y relationship we generally use cost function i.e Mean square Error.

i.e

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

or,

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$y_i \rightarrow$ observed values

$\hat{y}_i \rightarrow$ predicted values.

Let us consider case study on simple linear Regression :- $\theta_0 = 0$

$h_\theta(x) = \theta_1 x$ (-

$\downarrow$ $\downarrow$

$y$ on (- slope)

$h_0(n) = Q_1(n)$ → equation.

y  m

$h_0(n) = 1$ $n = 1$

$h_0(n) = 2$ $n = 2$

$h_0(n) = 3$ at $n = 3$

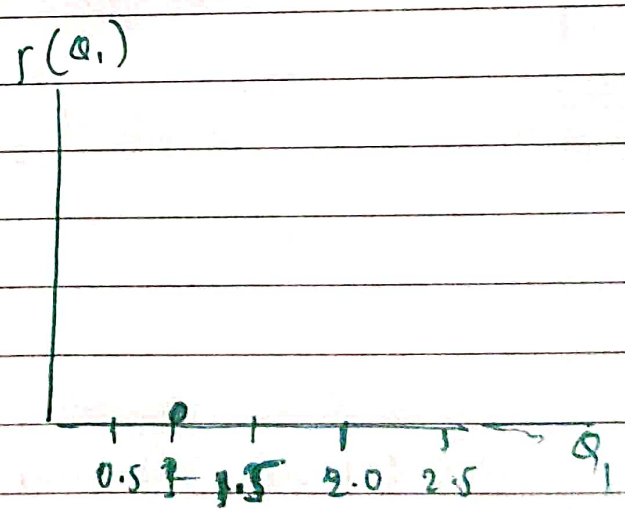
$h_0(n) = 4$ at $n = 4$

Here 45° the line passing through origin $\theta = \tan^{-1}(45) = 1$

$$J(Q_1) = \frac{1}{n} \sum_{i=1}^{n} \left( h_0(n)^i - y^{(i)} \right)^2$$

$$= \frac{1}{3} \left[ (1-1)^2 + (2-2)^2 + (3-3)^2 \right]$$

$J(Q_1) \Rightarrow 0$




Consider case II

$Q_1 = 0.5$

$h_1(n) = Q_1 \times n$

$h_1(n) = 0.5 \quad n = 1$

$h_1(n) = 0.5 \times 2$ at $n = 2$

$h_1(n) = 0.5 \times 3$ at $n = 3$

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

for case II

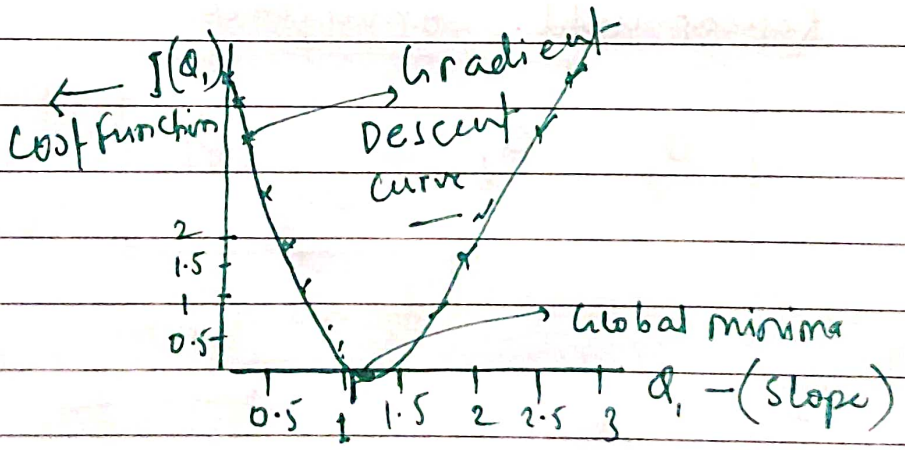
$$\Rightarrow \frac{1}{3}\left[ (0.5-1)^2 + (1-2)^2 + (1.5-3)^2 \right]$$

$$\Rightarrow \frac{1}{3}\left[ 0.25 + 1 + 2.25 \right]$$

$$\Rightarrow \frac{3.50}{3} = \boxed{1.66}$$

Observation:- while changing $\theta_1$ (slope) we see the different $J(\theta_1)$ → cost function

So, Let's visualize:-




Here cost function will be upward parabola - called as "Gradient Descent"

## Convergence algorithm

Question? Can we take random slope So, not all.

So, what should be the way to choosing or optimizing $\alpha$ (slope) so that gd (descent gradient always conserve towards global minima) So, Here is the concept comes in existence

i.e Convergence Algorithm states:
return to change or optimize the value of $\alpha_1$ → slope.

Repeat until Convergence

$$\alpha_j = \alpha_j - \alpha \frac{\partial}{\partial \alpha_j} J(\alpha_j)$$

→ partial differentiation of gradient

→ learning rate

**Case Study I**

For $\frac{\partial}{\partial \alpha_j} J(\alpha_j) \rightarrow$ (-ve)

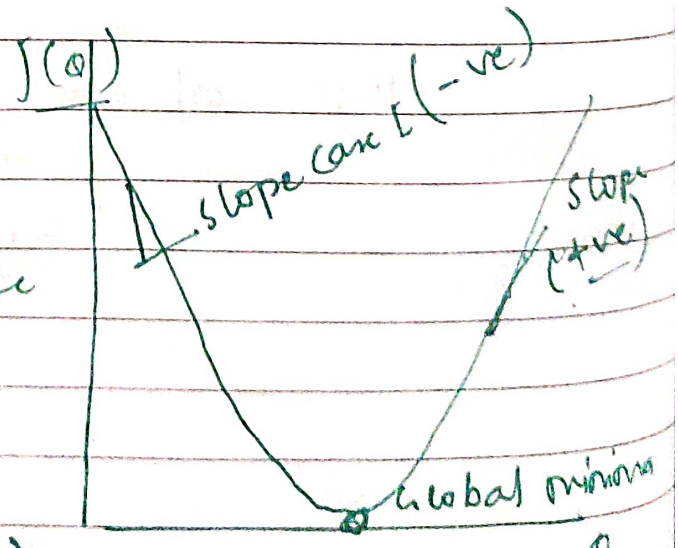
↓

negative slope

So,

$$\alpha_j = \alpha_j - \alpha (-ve)$$

$$\alpha_j = \alpha_j + \alpha \left(\frac{\partial}{\partial \alpha_j} \alpha_j\right)$$



$\theta_j = \theta_j + \alpha \cdot 0$

$\theta_j$ increase in case I

Case II

$$\theta_j = \theta_j - \alpha (+ve)$$

↳ slope of gradient (+ve)

$\theta_j \rightarrow$ decreases

Fact :- (i) If gradient slope (-ve)

$\theta_j = \theta_j - (+ve\ value) \rightarrow \theta_j$ decrease

(ii) If gradient slope (+ve)

$\theta_j = \theta_j - (-ve\ value) \rightarrow \theta_j$ increase

(iii) $\alpha$ - Learning rate is large.

↳ Gradient descent can overshoot the minimum.

(iv) $\alpha \rightarrow$ is very small :- If easy to reach local minima.