Data Science

&

Machine Learning

Soumyansh

Simple Linear Regression

Supervised Machine Learning

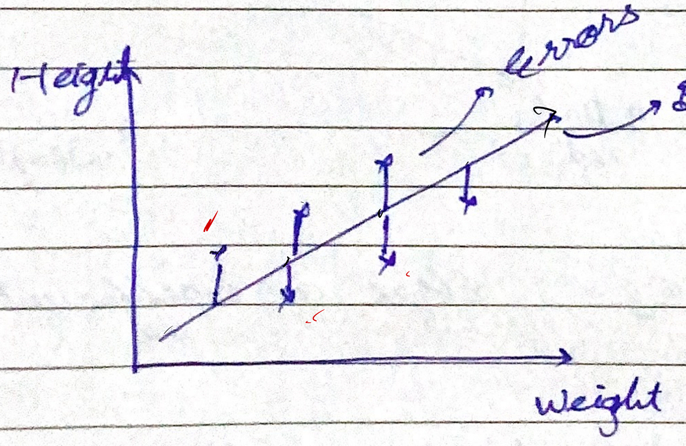
Dataset

I/P feature → **Weight** | **Height** → Dependent feature

Weight	Height
74 kg	170 cm
80 kg	180 cm
75 kg	175.5
-	-

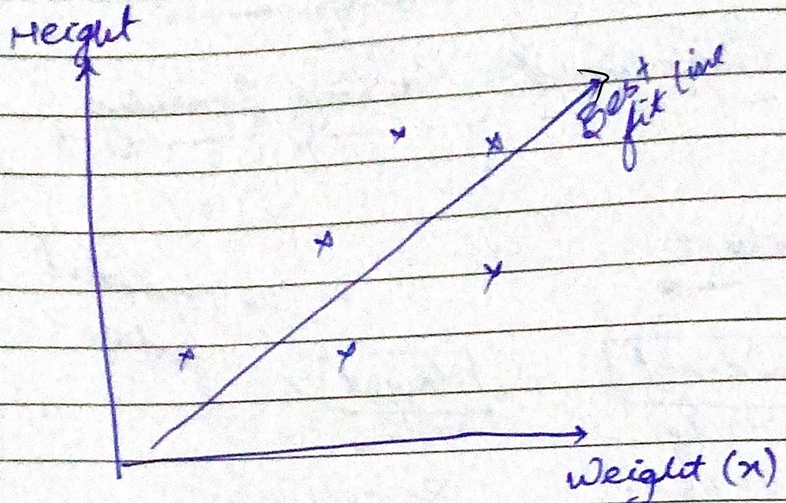
} Past Data

New weight → $\boxed{\text{Model}}$ → Height
(Training)



We try to minimise the distance.

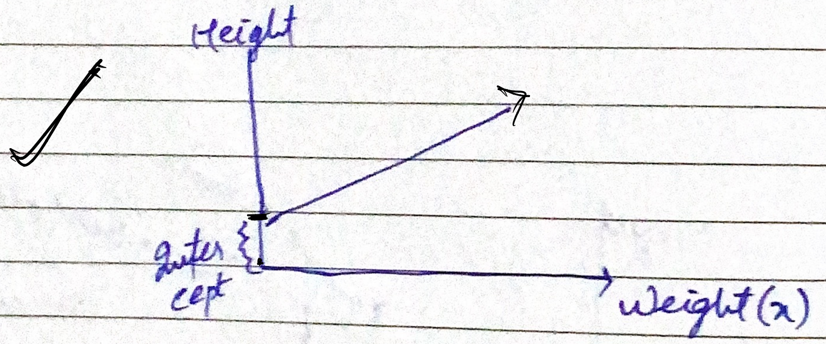
$\boxed{Y = mx + c}$



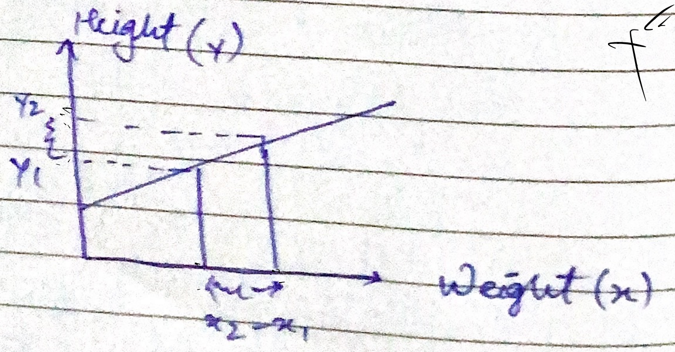
$h_\theta(x) = \theta_0 + \theta_1 x$

Imp Now θ_0 = Intercept

If $x = 0$ then $h_\theta(x) = \theta_0$



Imp Now θ_1 = Slope or coefficient



$Y = mx + c$

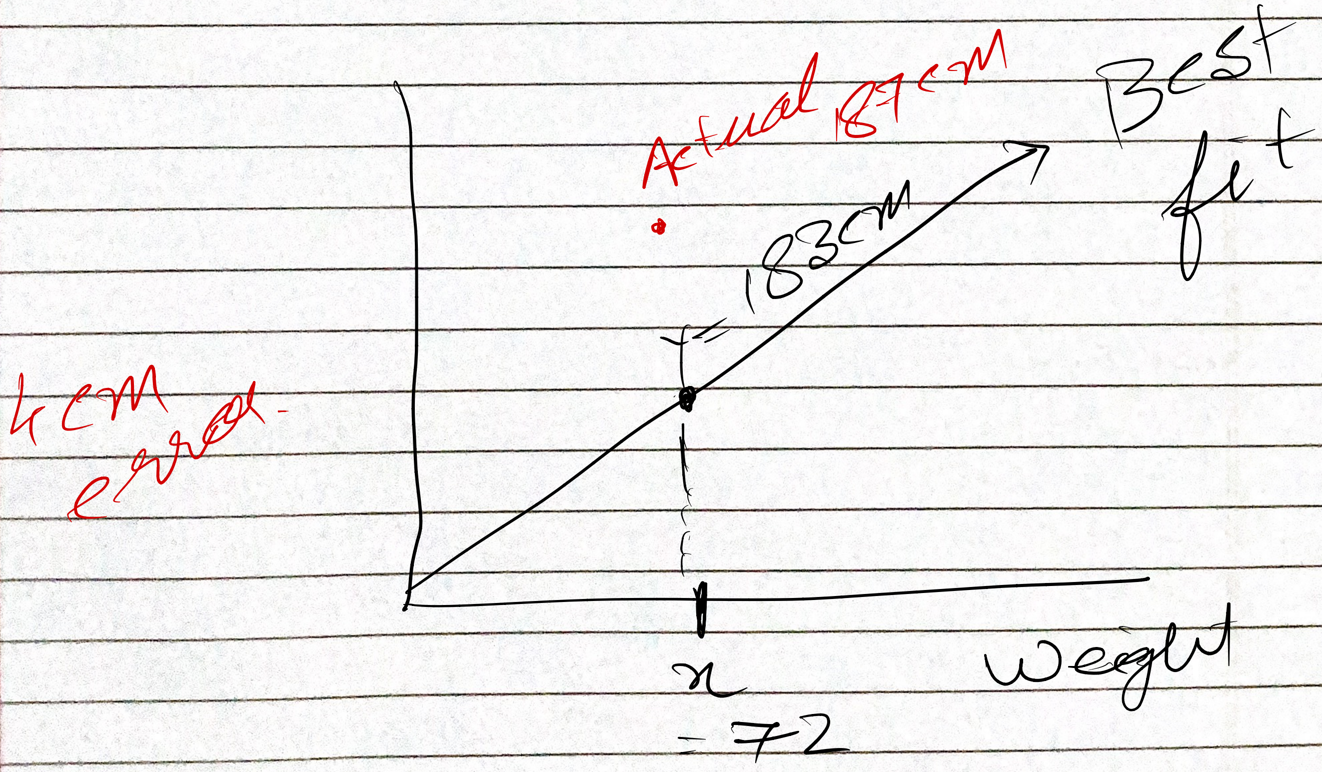
$\frac{Y_2 - Y_1}{x_2 - x_1}$

Error = $y - \hat{y}$ [where $\hat{y}$ = predicted value of y by model]

Now our aim is to minimise the sum of all errors

$$\boxed{\sum_{min} \{ y - \hat{y} \}}$$

$\hat{y} = y$ predicted



Machine Learning → Notes Part 2

$\boxed{\text{Cost function}}$

$\boxed{\text{Gradient Descent}}$ → An algorithm to minimize a function by optimizing its parameters

Let's say there is a Science Test Max marks = 50

Friend asks you to guess his marks?

Guess 1 → 45 ?
friend → Too far
Guess 2 → 40 ?
friend → Still far
Guess 3 → 37
friend → Very close

$\boxed{y = mx + c}$

Imp → In GD we start with random guess & then slowly move to the right answer.
How fast/slow we'll converge to the answer is determined by learning rate

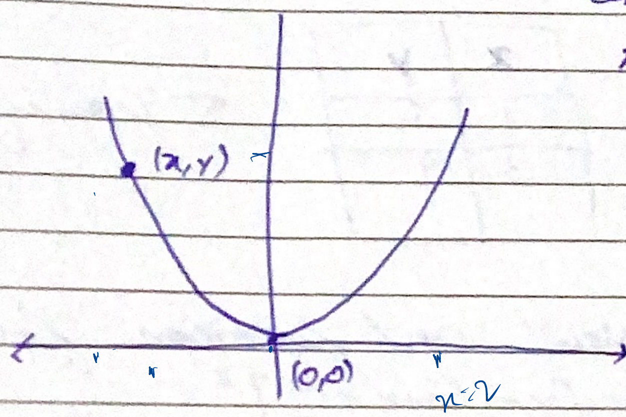
$\boxed{\text{New value} = \text{Old value} - \text{Stepsize}}$

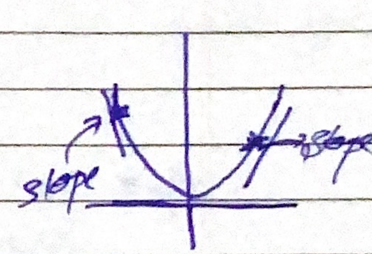
where Stepsize = Learning rate × slope

Ex → A Simple Square function

$$f(x) = x^2$$

Now as per GD this function should be minimised.





Now Step 1) Random guess (−2, 3) (x & y)

Step 2 → Here $x = -2$ then $f(x) = x^2$ &
Slope of function = $\frac{d}{dx} f(x) = 2x = 2(-2) = -4$

Step 3 → If $x = -1$ then $2x$ ie slope = -2
Hence we are getting close to zero ie moving in correct direction

Step 3 → If $x = +2$ then $2x$ ie slope = 4 which is beyond zero so we're in wrong direction now

Step 4 → Now $\boxed{\text{new value} = \text{old value} - \text{Slope} \times LR}$

& if a function has multiple parameters unlike $f(x) = x^2$, ie function having more than one parameter.

Ex Cost func^n for regression =

$$J(m,c) = \sum_{i=1}^{n} (y_i - (mx_i + c))^2$$

Step5 → Now there are two params m & c

Step6 → Again random guess let's say
{ $c = 0$, $m = 1$ }

x	y
1	2
3	4

← Training Data

Step7 → Now cost function for linear regression
$= [y - (mx + c)]^2$

Plugging the values for training data (1,2) & (3,4)

$J(m,c) = [2 - (c + m \times 1)]^2 + [4 - (c + 3m)]^2$

partial derivative

$\frac{dJ}{dc} = -2[2 - (c+m)] - 2[4 - (c+3m)]$

for $c = 0$ & $m = 1$

differentiation wrt c

$= -2[2-1] - 2[4-3]$

$= -4$

Step8 Now because i differentiated wrt c
so my
new c = old c − LR × (−4) ↗ slope
$= 0 - (0.001) \times (-4)$
$= 0.004$

Step9 Similar to above now we'll differentiate wrt m & new value of m will be found

Step10 → Now new values will be taken for consideration & entire process will get repeated

Step11 → When algo will stop?
Ans → When there won't be much improvement on cost for values of c & m

Imp → Choosing learning Rate: Try to be moderate