AI & ML

1 row → sample

1 column → Criteria under that sample

Sample = ~~Attribute~~ Instance

Features = Attribute

In a row:

$n-1$ = Features

$[-1]$ = Output

Classification

Discrete Set of Output.

*** A classification Algo may predict a continuous value but the continuous value is in form of of a probability for a class label.

Regression

Continuous Set of Output.

*** A regression Algo may predict a discrete value, but the discrete value in the form of integer Quantity.

Just to know:

Human learning vs ~~HG~~ ML ~~AI~~ Algo

Human - brain stimulation.

ML - parameter.

parameter vs Hyperparameter (Not yet)

* Get idea of Training set, validation set & Test set.

By Example: SSC Exam:

Training set: Given that u can learn from.

i) class, coaching, private tution: Training set.

ii) Model Test, class Test, private tutor Exam: Valiation set.

iii) SSC Final Exam: Test set (Pass, Fail)

Dataset

- Training set (70%)
- Validation (15%)
- Test set (15%)

All are disjoint.

Training set: to fit the ~~para~~ parameters.

Validation set: to tune the parameteres. [Validation set is used to estimate prediction error for model ~~Alle~~ selection.]

Test set: to assess the performance. [the test set is used for assessment of the generalization error of the (Judgemental) final chosen model.

ML

- Supervised
  - Classification
    - predict(level)
      - binary class
      - Multi-class
  - Regression
    - predict(quantity)
- Unsupervised
- Re-inforcement

* What is Linear Regression?

⇒ Linear regression is a method to predict dependent variable (Y) based on values of independent variable (X). It can be used for the cases where we want to predict some continuous quantity.

* Training set of housing prices (Portland, OR)

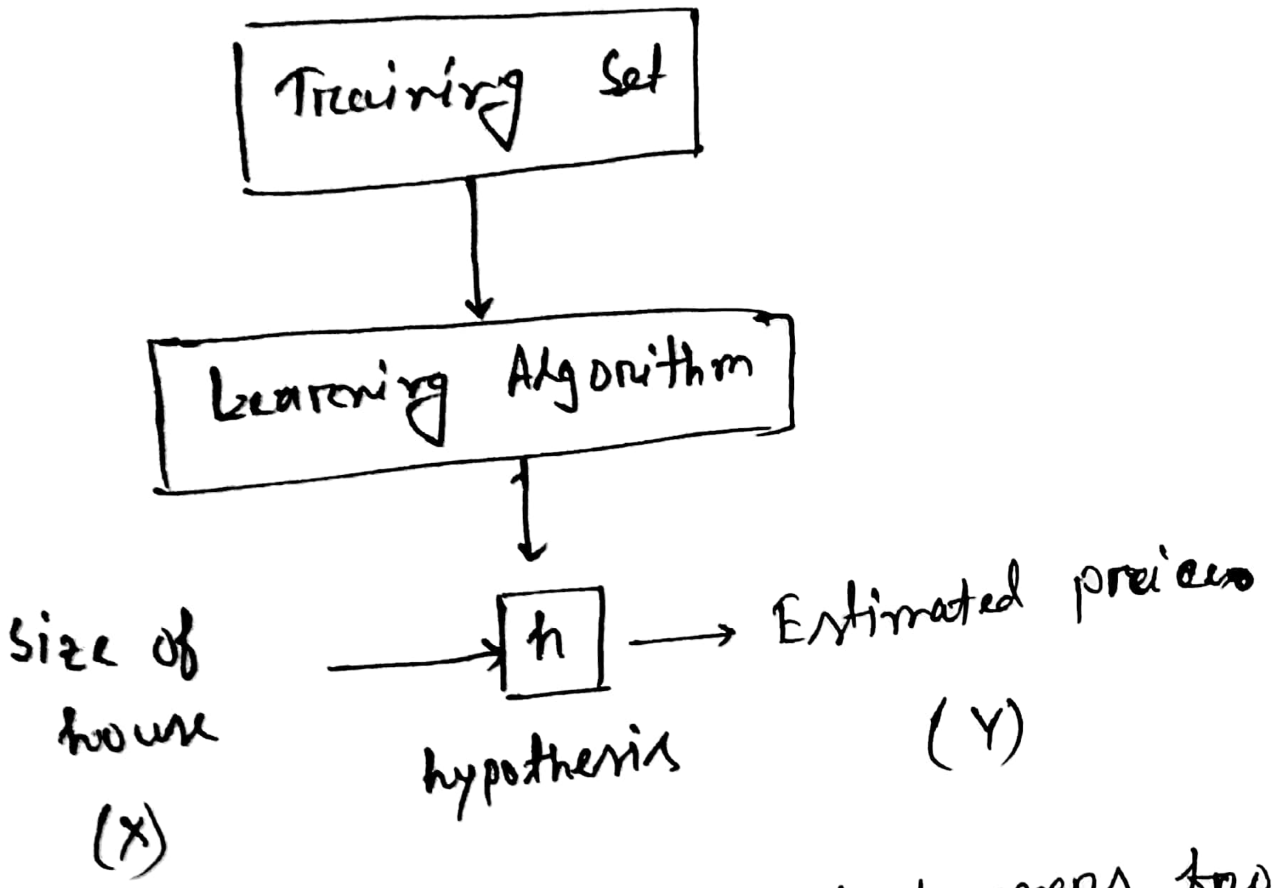
Size in feet^2 (X)	Price ($) in 1000's (Y)
2104	460
1416	232
1534	315
852	178
...	...

$m = 47$

Notation: m = Number of training Examples

x's = input var / features

y's = output var / target var

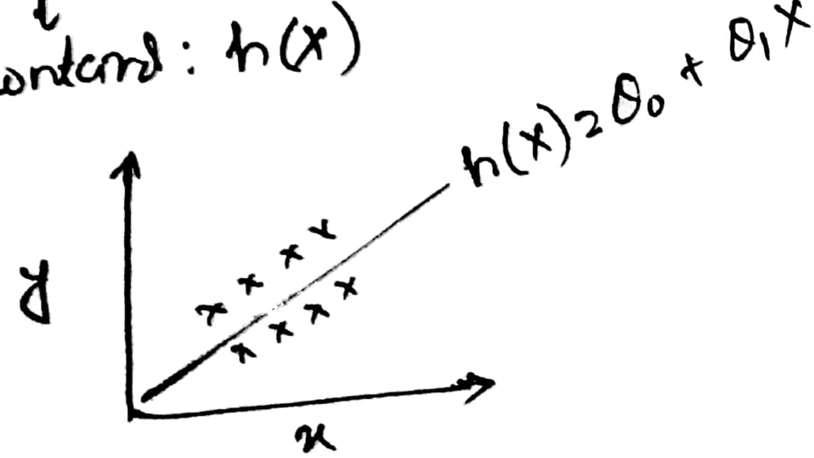


So, h is a function that maps from x's to y's.

How do we represent h?

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

Shortand: $h(x)$



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

θ_i's : Parameters

How do we choose parameters?

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y).

* Minimizing the problem:

$$\underset{\theta_0 \theta_1}{\text{Minimize}} \quad (h_\theta(x) - y)^2$$

For, (x_i, y_i):

\# training examples.

$$\boxed{\underset{\theta_0 \theta_1}{\text{minimize}}} \quad \frac{1}{2m} \sum_{i=1}^{m} \Big(\underbrace{h_\theta(x_i)} - y_i \Big)^2$$

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

It means you'll find me ~~the~~ the values of θ_0 & θ_1 that causes this expression to be minimized.

And this expr depends on θ_0 & θ_1.

By convention:

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

Minimize θ_0, θ_1 $\quad \underbrace{J(\theta_0,\theta_1)}$

cost function is also called squared Error function.

→ In this case, Not always.

Just summerize in this particular Area:

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

Parameters: θ_0, θ_1

Cost function:

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

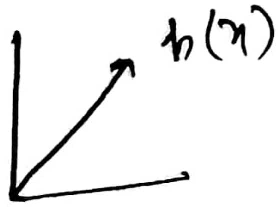
Goal: minimize θ_0, θ_1 $J(\theta_0,\theta_1)$ to get optimal.

Simplified

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

$$\theta_0 = 0$$

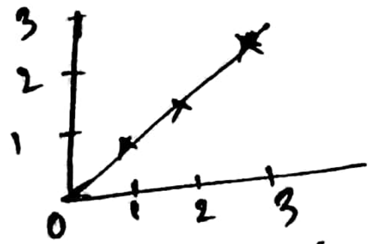
$$h_\theta(x) = \theta_1 x$$



$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \big(\underbrace{h_\theta(x_i)}_{\theta_1 x_i} - y_i\big)^2$$

hypothesis func: $h_\theta(x)$

For fixed θ_1, this is function of x



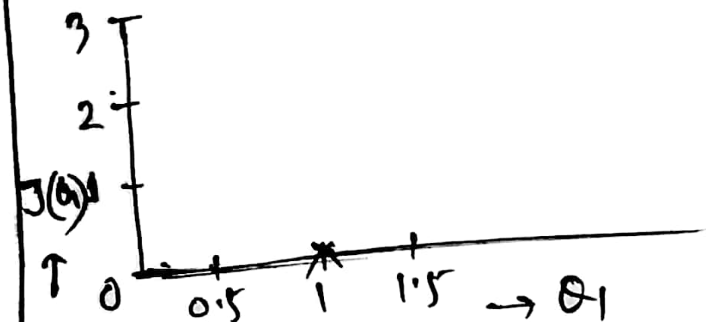
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\theta_1 x_i - y_i)^2$$

$$= \frac{1}{2m} (0^2 + 0^2 + 0^2)$$

$$= 0$$

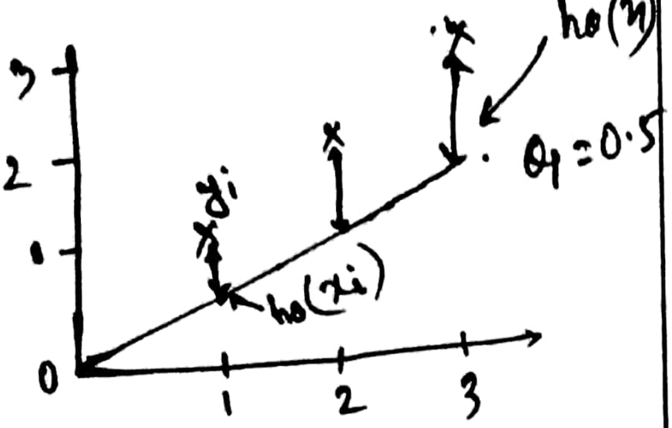
cost func: $J(\theta_1)$

function of the parameter θ_1

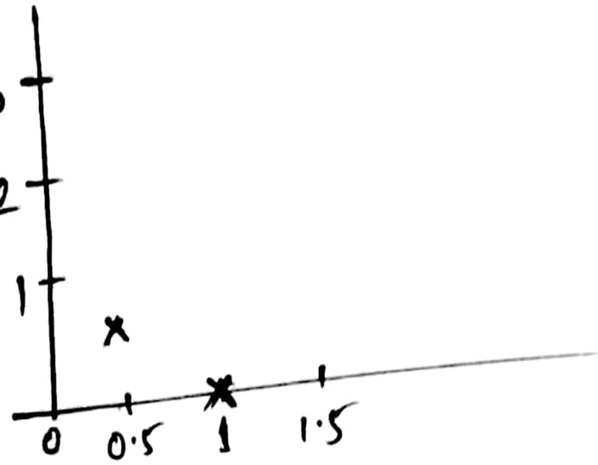


$$J(1) = 0$$

$h_\theta(x)$



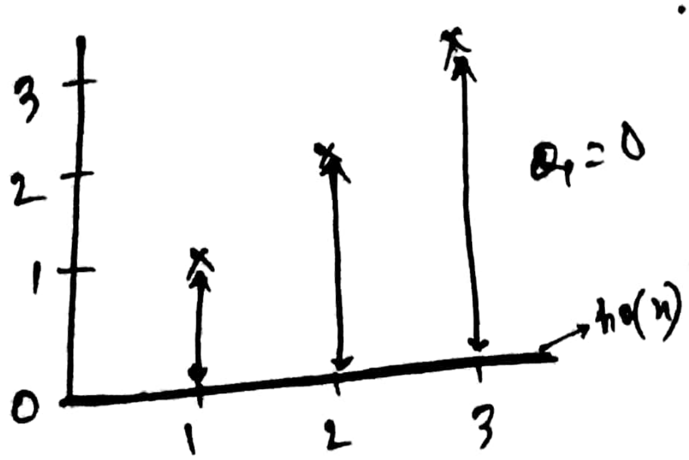
$J(\theta_1)$

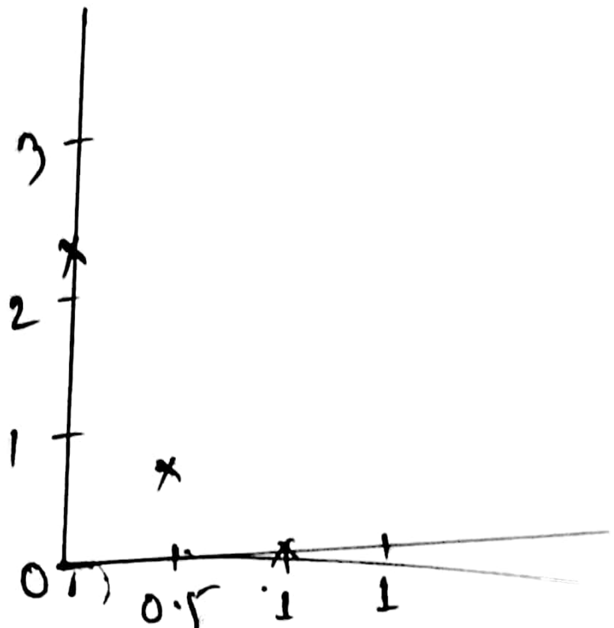


$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

$J(0.68)$

≈ 0.58

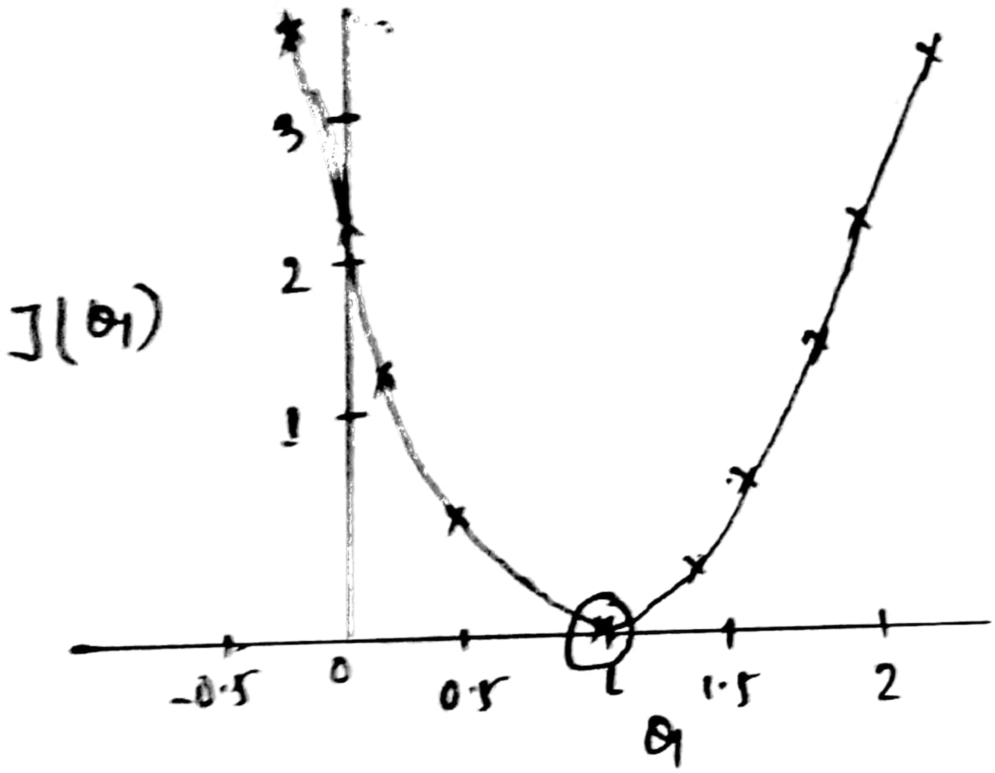




$$J(0) = \frac{1}{2m}\left(1^2 + 2^2 + 3^2\right)$$

$$= \frac{1}{6} \cdot 14 \approx 2.3$$

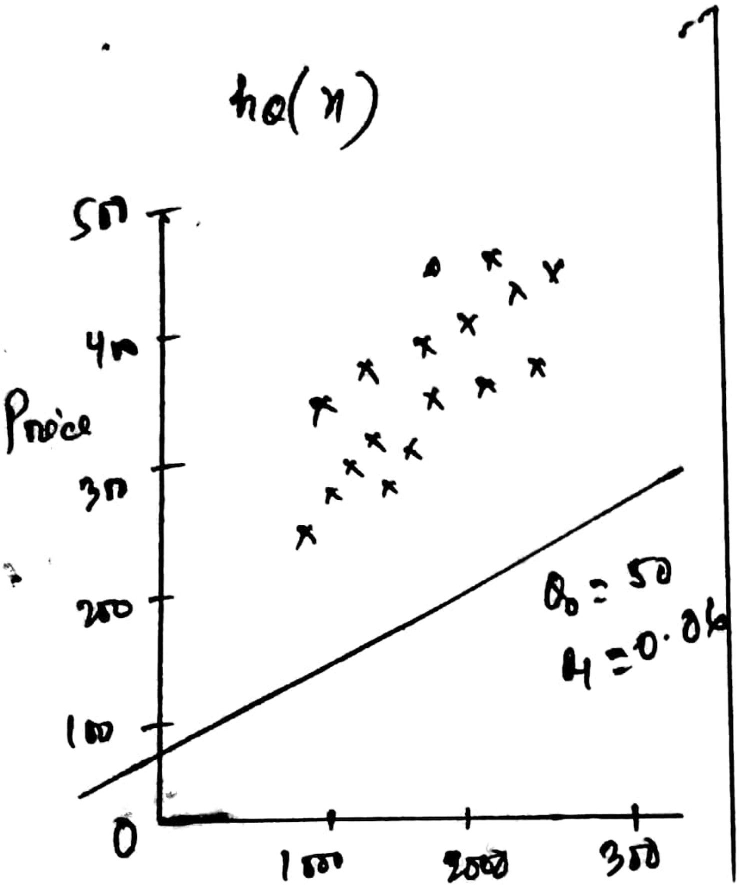
$J(\theta_1)$

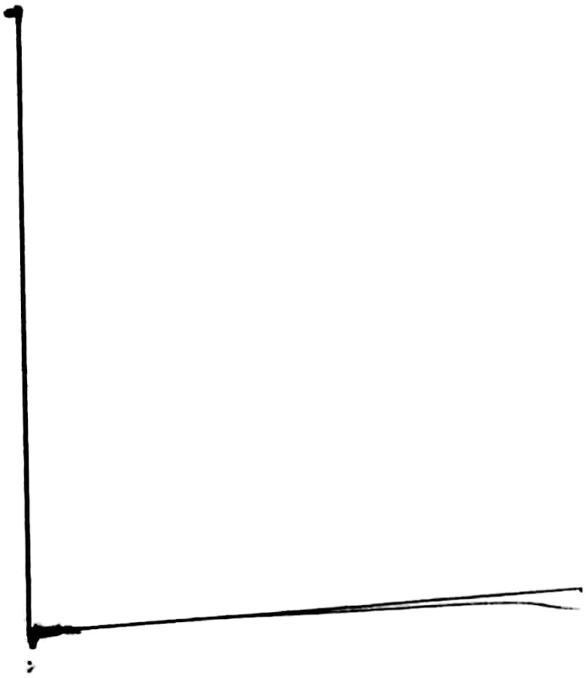


: $J(\theta_1)$

Minimize $J(\theta_1)$: $\theta_1 = 1$

$h_\theta(x)$





$h_\theta(x) = 50 + 0.06x$

Learn to know!

contour plots

or contour figures.

Gradient descent

is an Algorithm to minimize the cost function $J(\theta_0, \theta_1)$. It's not only used in Linear Regression, its generally used in almost of ML ~~pr~~ places.

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ | $J(\theta_0, \theta_1, \theta_2, \ldots \theta_n)$

$\min_{\theta_0 \ldots \theta_n} J(\theta_0 \ldots \theta_n)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum.

:= Assignment

repeat untill convergence {

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

}

Learning rate

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

Simultaneous update θ_0 and θ_1

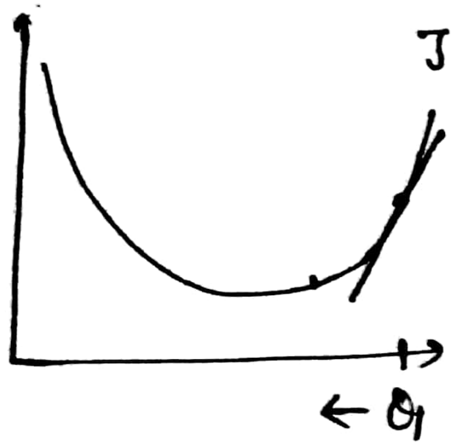
Correct: Simultaneous Update

$\Rightarrow temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\Rightarrow temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\Rightarrow \theta_0 := temp0$

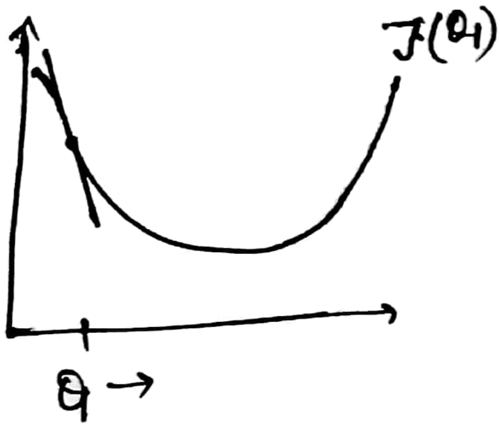
$\Rightarrow \theta_1 := temp1$



$(\theta_1 \in \mathbb{R})$

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

$$\theta_1 := \theta_1 - \alpha \cdot (\text{positive no})$$



$$\frac{\partial}{\theta_1} J(\theta_1) \leq 0$$

$$\theta_1 := \theta_1 - \alpha\,(\text{negative no})$$

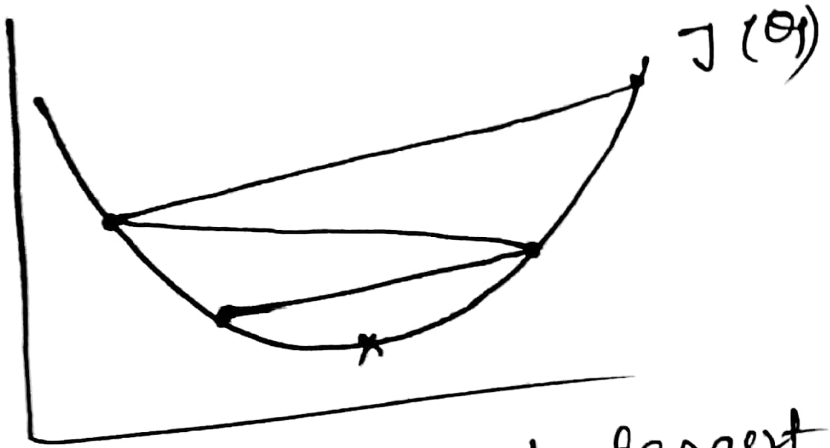
so, If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

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

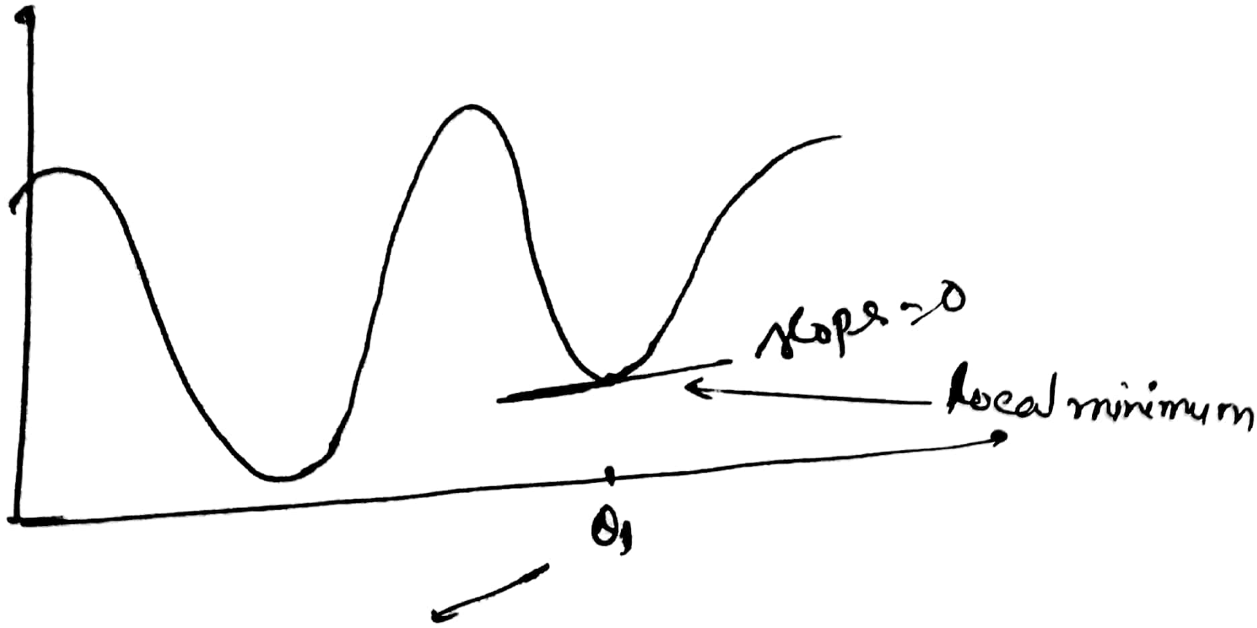
~~$\theta_1 : \theta_1$~~

lets see example:

α is too large means learning steps are very big.



If α is too small, gradient descent can be slow



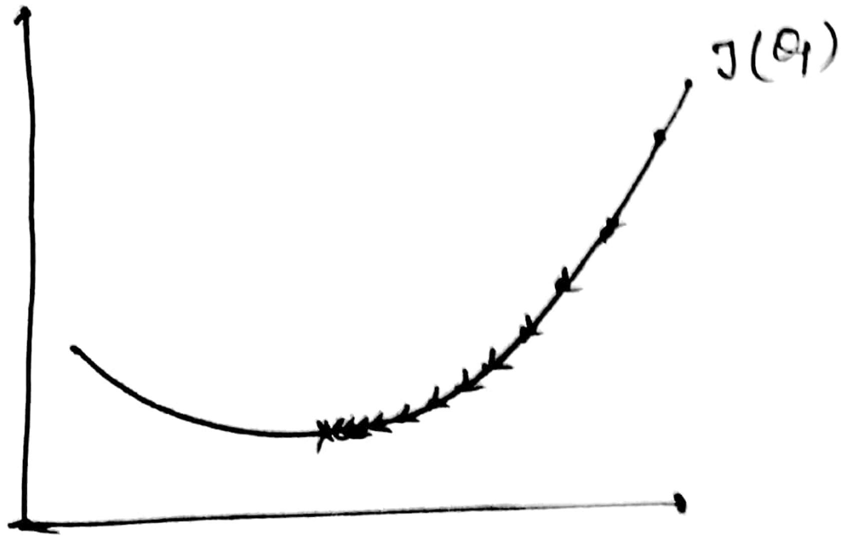
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

$$\theta_1 := \theta_1 - \alpha \cdot 0$$

$\theta_1 := \theta_1$ remain unchanged.

* Gradient descent can converge to a local minimum, even with the learning rate α fixed.

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



As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.

Gradient descent Algo:

repeat until convergence

$\{$

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

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

$\}$

Linear Regression Model

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

$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_i - y_i)^2$$

Now,

θ_0 | $j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$

θ_1 | $j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) \cdot x_i$

That's final Gradient descent Algo for Linear Reg.

A 3D bow shape function represent that linear Regression cost function.

Another called: "convex function".

"Batch" Grading Descent.

- "Batch": Each step of gradient descent ~~usage~~ uses all the training example.

$$\sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$