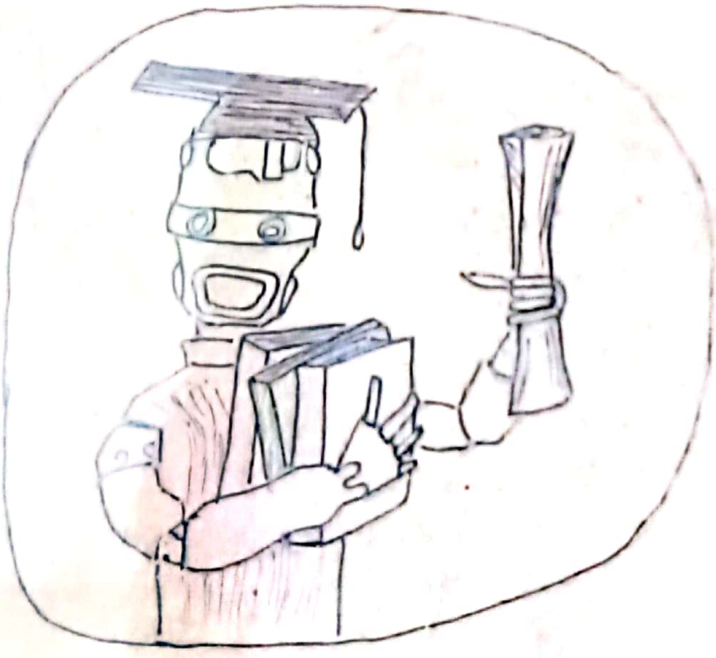
COURSERA

MACHINE LEARNING

BY

ANDREW NG,

STANFORD UNIVERSITY



NOTES BY RUTHUPARNA K (EC dept.)

MACHINE LEARNING

BY ANDREW NG

WEEK 1

INTRODUCTION

What is Machine Learning?

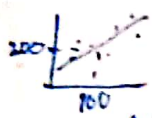
"A computer program is said to learn from a experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

— Tom Mitchell

Supervised Learning.

In supervised learning, we are given a labeled data set and already know what our correct output should look like, having the idea that there is a relationship between the input and output.

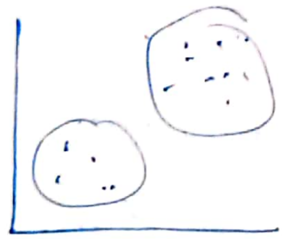
- Regression: Mapping input variables to some continuous function



- Classification: Mapping input variables to discrete catagories

3) Unsupervised Learning

When the data isn't labelled, the algorithm finds patterns and clusters the data.



4) Notation:

m - number of training examples
x - feature - input variable
y - target variable - output variable
$(x^{(i)}, y^{(i)})$ - ith training example where $i = 1, 2 \ldots m$

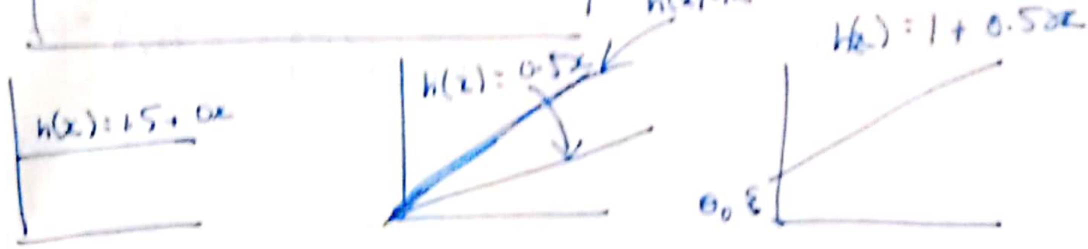
Univariate Linear Regression

Cost Function

hypothesis function

$$\boxed{h_\theta(x) = \theta_0 + \theta_1 x}$$

$\theta_0 = 0$
$\theta_1 = 1$
$h(x) = 1x$

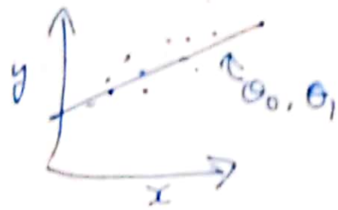


$\theta_0 = 1.5$ $\theta_1 = 0$

$\theta_0 = 0$ $\theta_1 = 0.5$

$\theta_0 = 1$ $\theta_1 = 0.5$

Aim



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

Cost Function: - also called squared error function

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

hypothesis function $\theta_0 + \theta_1 x^{(i)}$ (prediction)

Value of y (actual)

$\therefore$ We minimise $J(\theta_0, \theta_1)$
θ_0, θ_1

Understanding cost function (Intuition I)

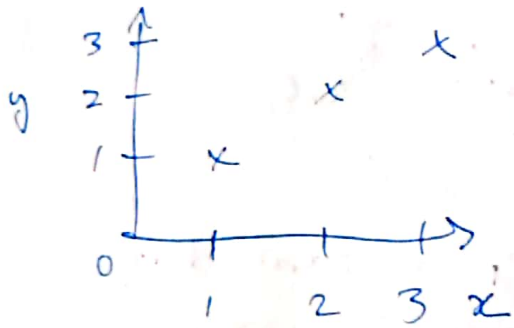
Let $\theta_0 = 0$

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

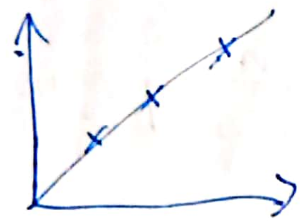
$$\therefore \quad J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \Big(\underbrace{h_\theta(x^{(i)})}_{\theta_1 x^{(i)}} - y^{(i)}\Big)^2$$

$\therefore$ We minimise $\underset{\theta_1}{} J(\theta_1)$

consider given data set

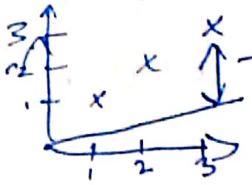


① Let $\theta_1 = 1$:



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

② Let $\theta_1 = 0.5$:

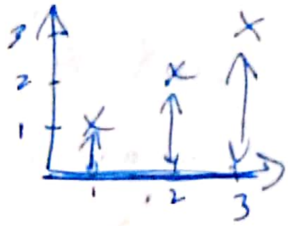


— Difference btwn predicted & actual value $h_\theta(x^{(3)}) - y^{(3)}$

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

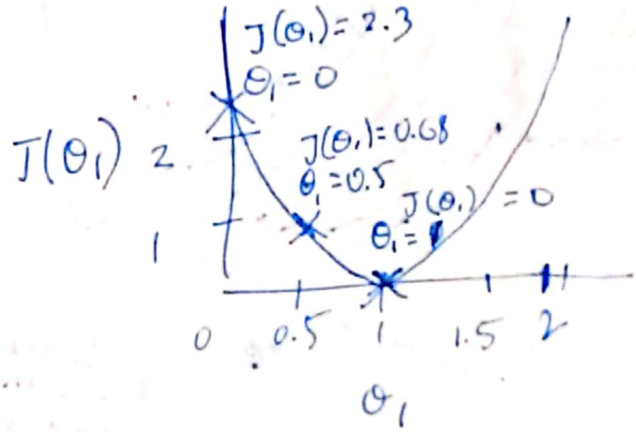
$$= 0.68$$

③ Let $\theta_1 = 0$:

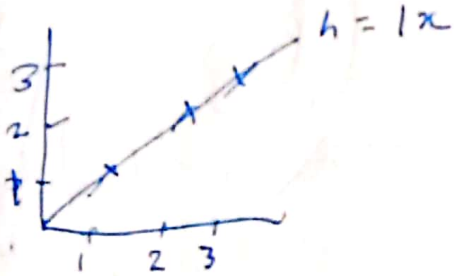


$$J(0) = \frac{1}{2 \times 3}\left(1^2 + 2^2 + 3^2\right) = \frac{14}{6} = 2.3$$

By plotting different values of θ_1 we get:

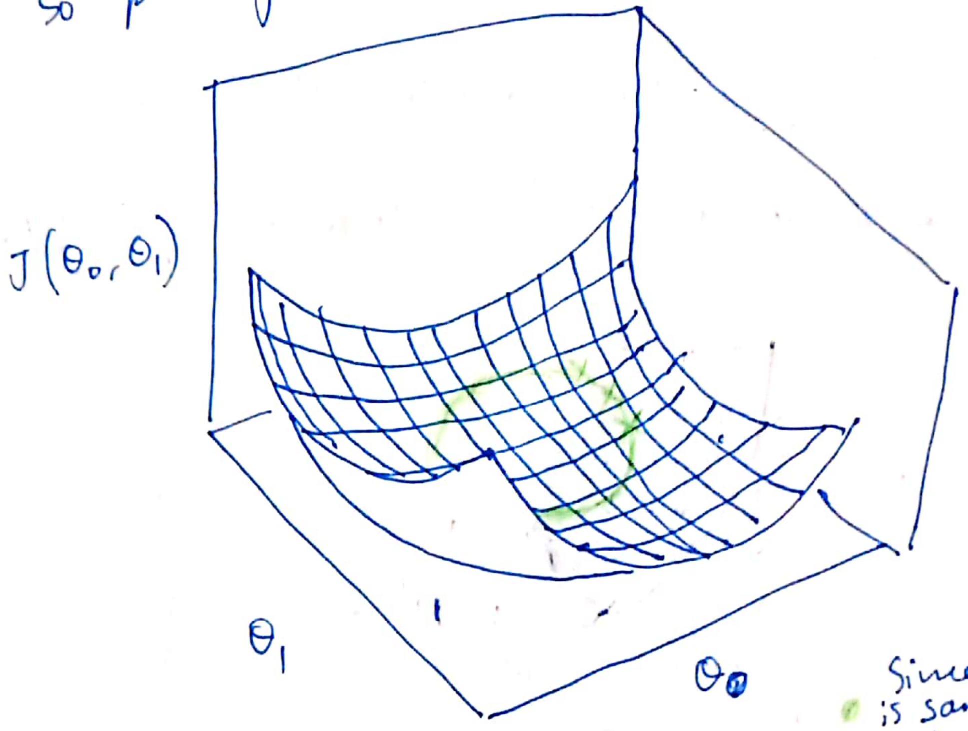


Hence when $\theta_1 = 1$, we get minimum value of $J(\theta_1)$. This is the best fit of the data



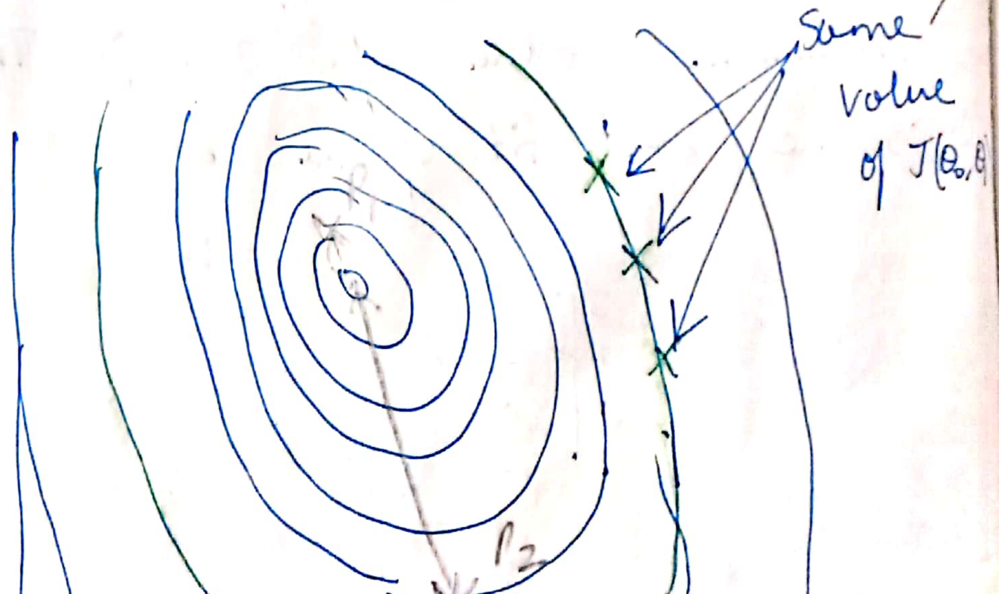
Intuition II

Now we consider both θ_1 & θ_2
So ploting $J(\theta_0, \theta_1)$ similar to previous example



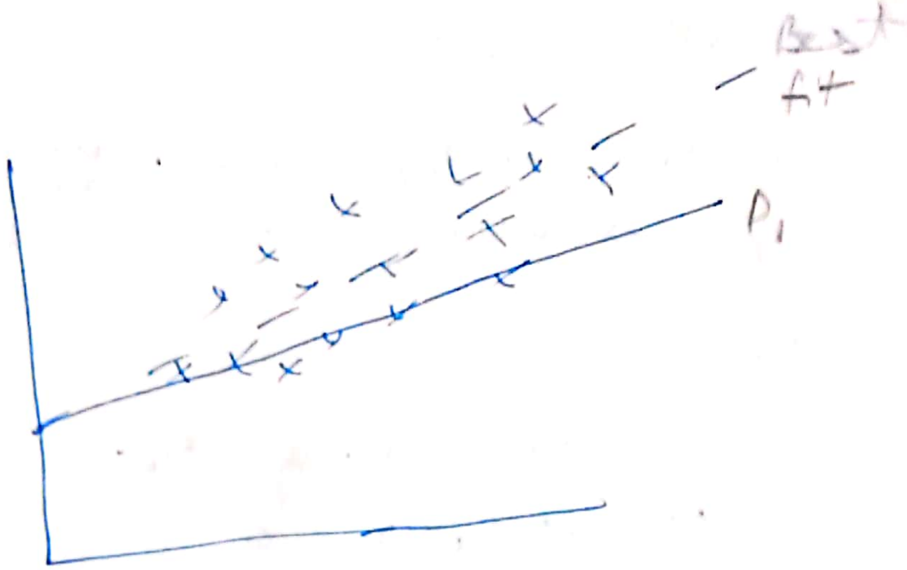
Since height is same along that circle

We can represent this in contour plot:

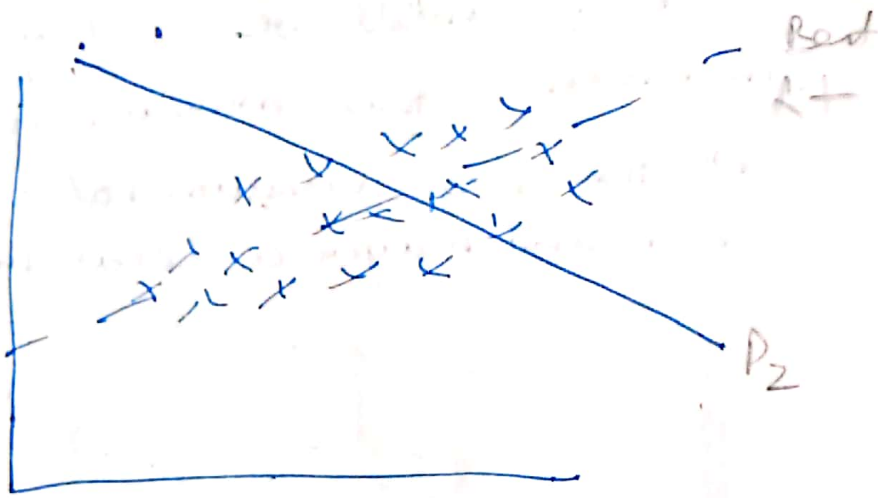


This is the top view of the 3D graph

P_1



P_2



∴ Closer to the centre, better the fit. So we need an algorithm that helps us find the centre. Therefore we use <u>GRADIENT DESCENT</u>

Gradient Descent

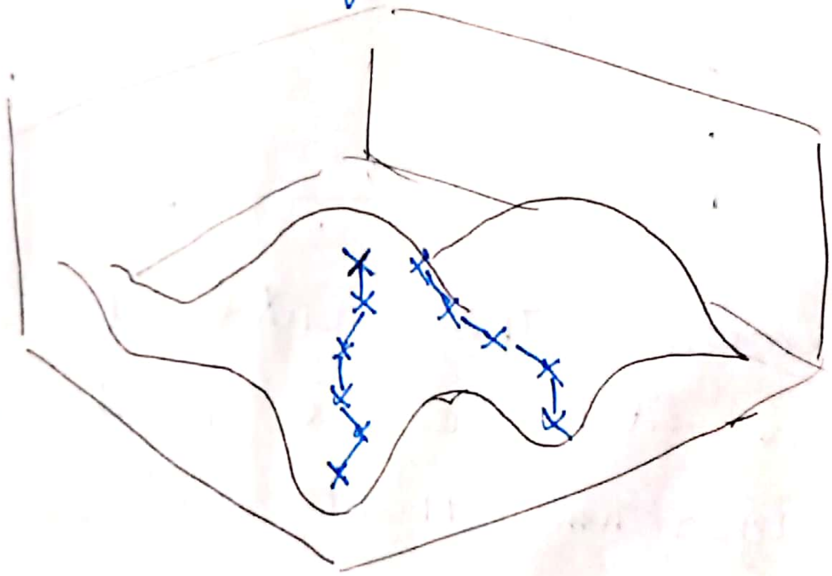
- Can solve function with n number of θ parameters $J(\theta_1, \theta_2 \dots \theta_n)$

Logic:

- Start with some θ_0, θ_1

  Eg: let $\theta_0 = 0$ & $\theta_1 = 0$

- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ untill we end up at a minimum (this minimum may not be the most minimum value in some cases and maybe a local minimum)



- Start
- Look around, which is lower point?
- Take a small step toward the point of maximum descent

Gradient Descent Algorithm

When the value starts changing by very little with every iteration

repeat until convergence {

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

}

assignment operator

learning rate - the size of the step taken

Here $j = 0$ and $j = 1$

∴ Simultaneosly update θ_0 & θ_1

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

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

$\theta_0 := temp0$

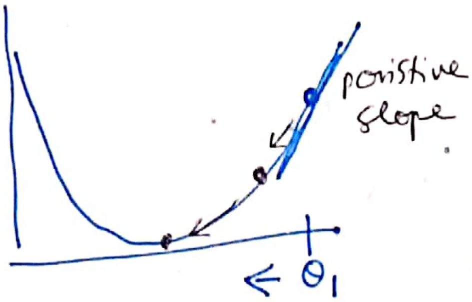
$\theta_1 := temp1$

Scanned with CamScanner

Intuition

consider only θ_1 ∴ $J(\theta_1)$

①

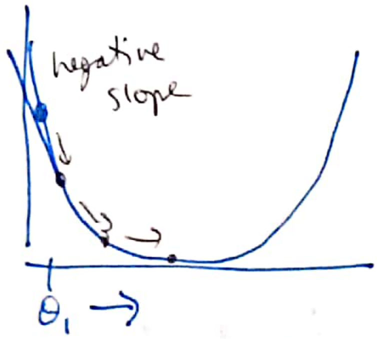


$$\theta_1 = \theta_1 - \alpha \underbrace{\boxed{\frac{d}{d\theta_1} J(\theta_1)}}_{}$$

slope

$$\therefore \theta_1 = \theta_1 - \alpha\,(\text{positive value})$$

∴ θ_1 is decreasing



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

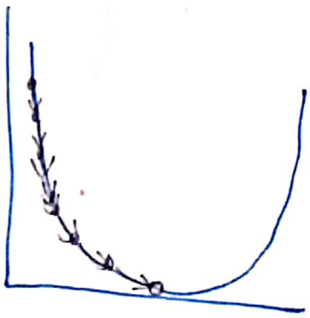
$$\therefore \theta_1 = \theta_1 - \alpha\,(\text{negative value})$$

$$\therefore \theta_1 = \theta_1 + \alpha\,(\text{positive value})$$

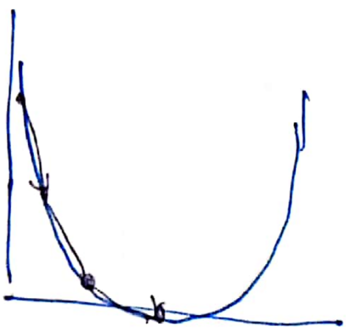
∴ θ_1 is increasing

∴ $\frac{d}{d\theta_1} J(\theta_1)$ controls where θ_1 changes (direction of descent)

②

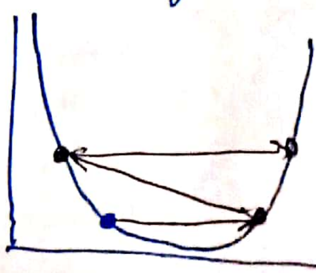


For small α
- gradient descent is slow



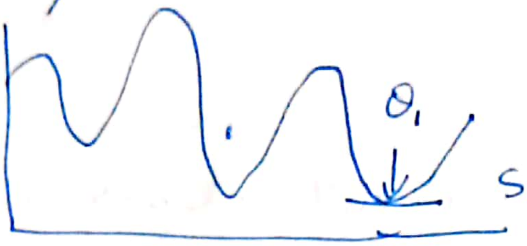
For large α
- gradient descent is fast

For too large α



- the steps are too big so it keeps overshooting & never converges

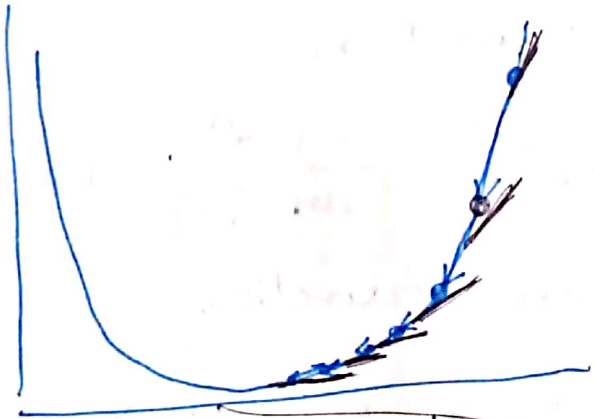
(3) If θ_1 is already at local minimum,



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

$\therefore \theta_1$ doesn't change

(4) As gradient descent occurs, the slope gradually decreases, so the step size also decreases



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

- This slope reduces with descent so overall step size decreases as we approach minimum

Gradient Descent for Linear Regression

So far,

- Gradient Descent algorithm,

  repeat untill convergence {

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

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

  }

- Cost function

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

- Hypothesis function

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

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

Simplifying this for θ_0 & θ_1,

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

∴ Gradient descent algorithm for linear regression (univariate)

repeat until convergence {

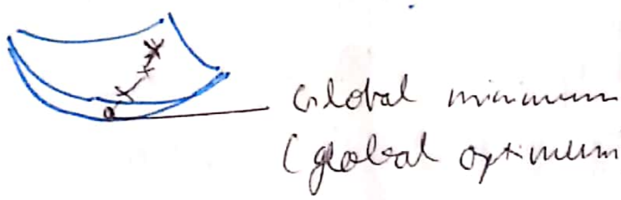
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

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

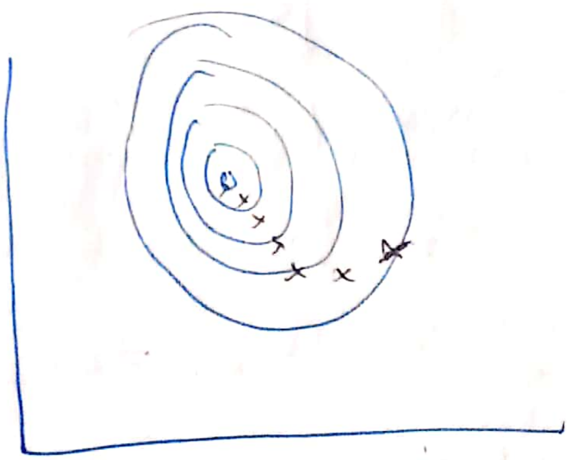
}

Note: update θ_0 & θ_1 simultaneously

Note: we had a problem where gradient descent would go to local ~~maximum~~ minimum, but in linear regression, we get a 'convex function' that has a global ~~maxi~~ minimum



Example



This is also called Batch gradient descent

Batch: Each step of gradient descent uses all the training examples (entire batch)