REGRESSION ANALYSIS : LINEAR

BY – MAUAJAMA FIRDAUS & TULIKA SAHA

MACHINE LEARNING

- It is the science of getting computer to learn without being explicitly programmed.

- Machine learning is an area of artificial intelligence concerned with the development of techniques which allow computers to "learn".

- More specifically, machine learning is a method for creating computer programs by the analysis of data sets.

APPLICATIONS

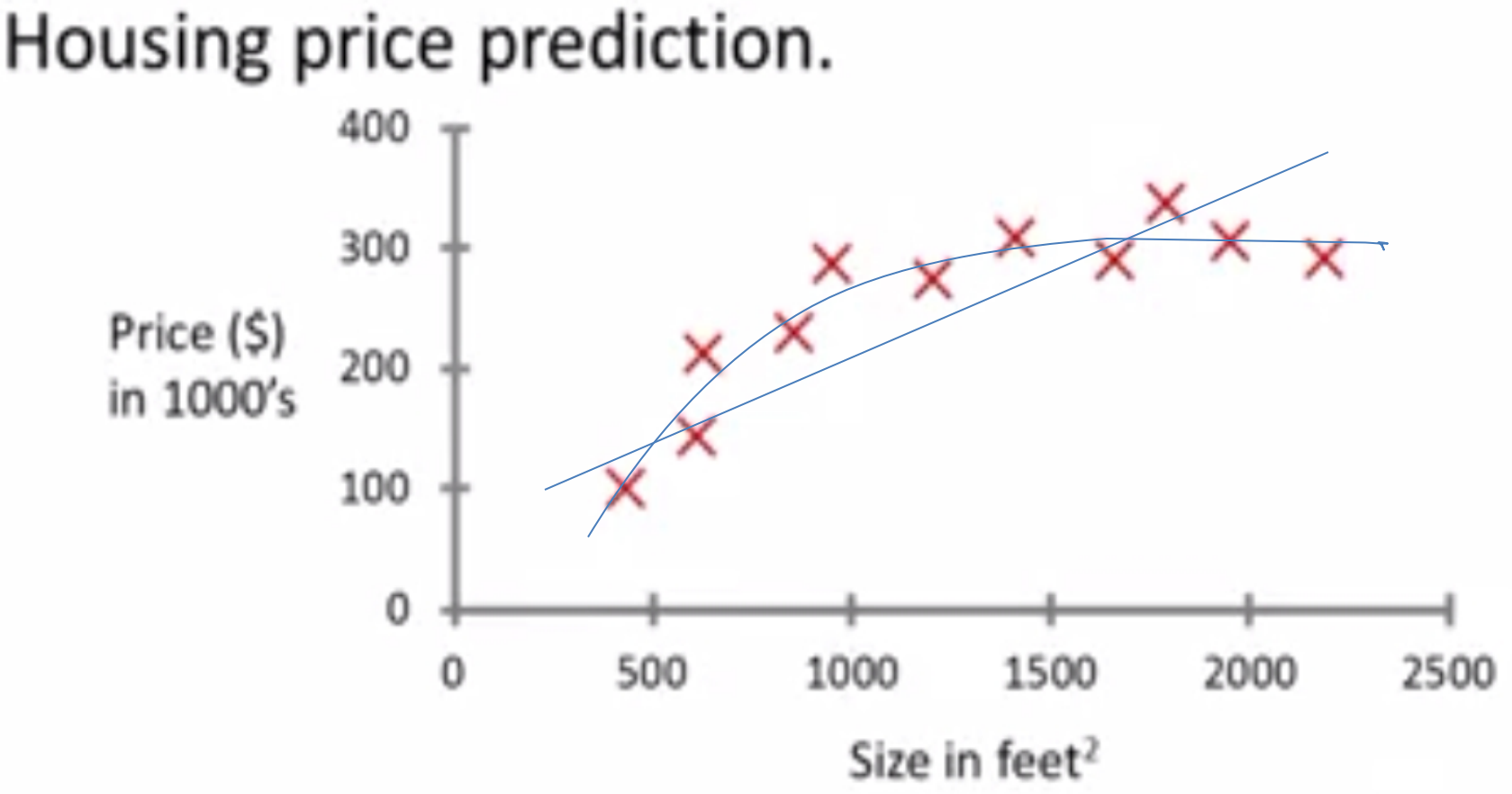
- Search Engines like Google, Bing etc.
- Facebook photo tagging application.
- Self Customizing Programs and many more.

SUPERVISED LEARNING

- Given the “right answer” for each of our examples in the training set.
- The task of the algorithm is to find many more such right answers for new examples.
- Consists of two problems-
    - Regression Problem
    - Classification Problem

REGRESSION PROBLEM

- The term “regression” refers to the fact that we are trying to predict a continuous-valued output.

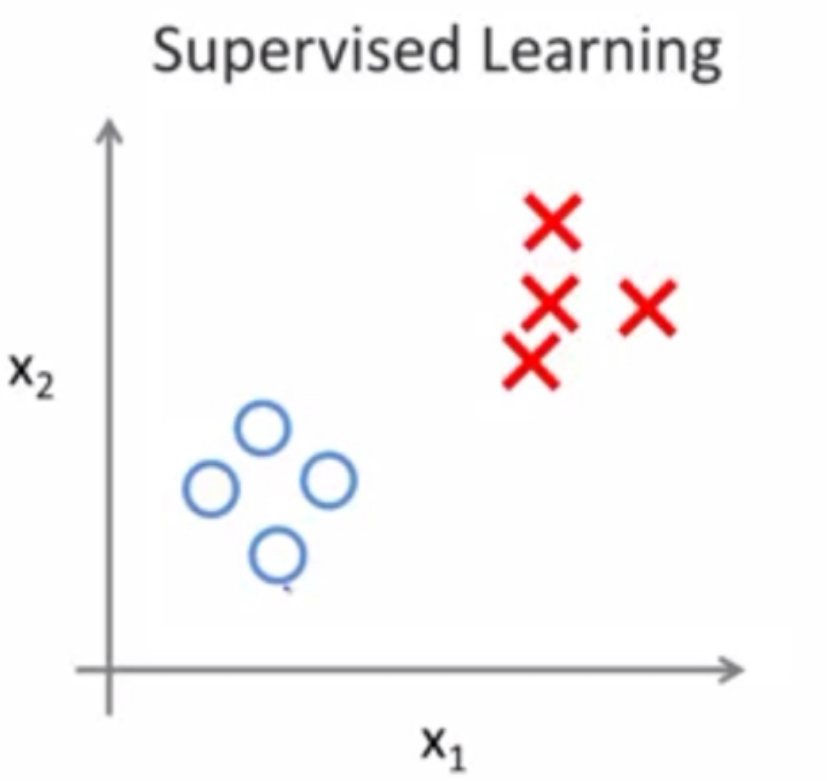


CLASSIFICATION PROBLEM

- It refers to the fact that we are trying to predict discrete valued outputs.

UNSUPERVISED LEARENING

- Given a set of data or examples in the training set without prior information of what the data is all about.
- The task of the algorithm is to find the structure among the data.
- Clustering Problem is an example of Unsupervised Learning.


Supervised Learning
x_2
x_1


Unsupervised Learning
x_2
x_1

LINEAR REGRESSION

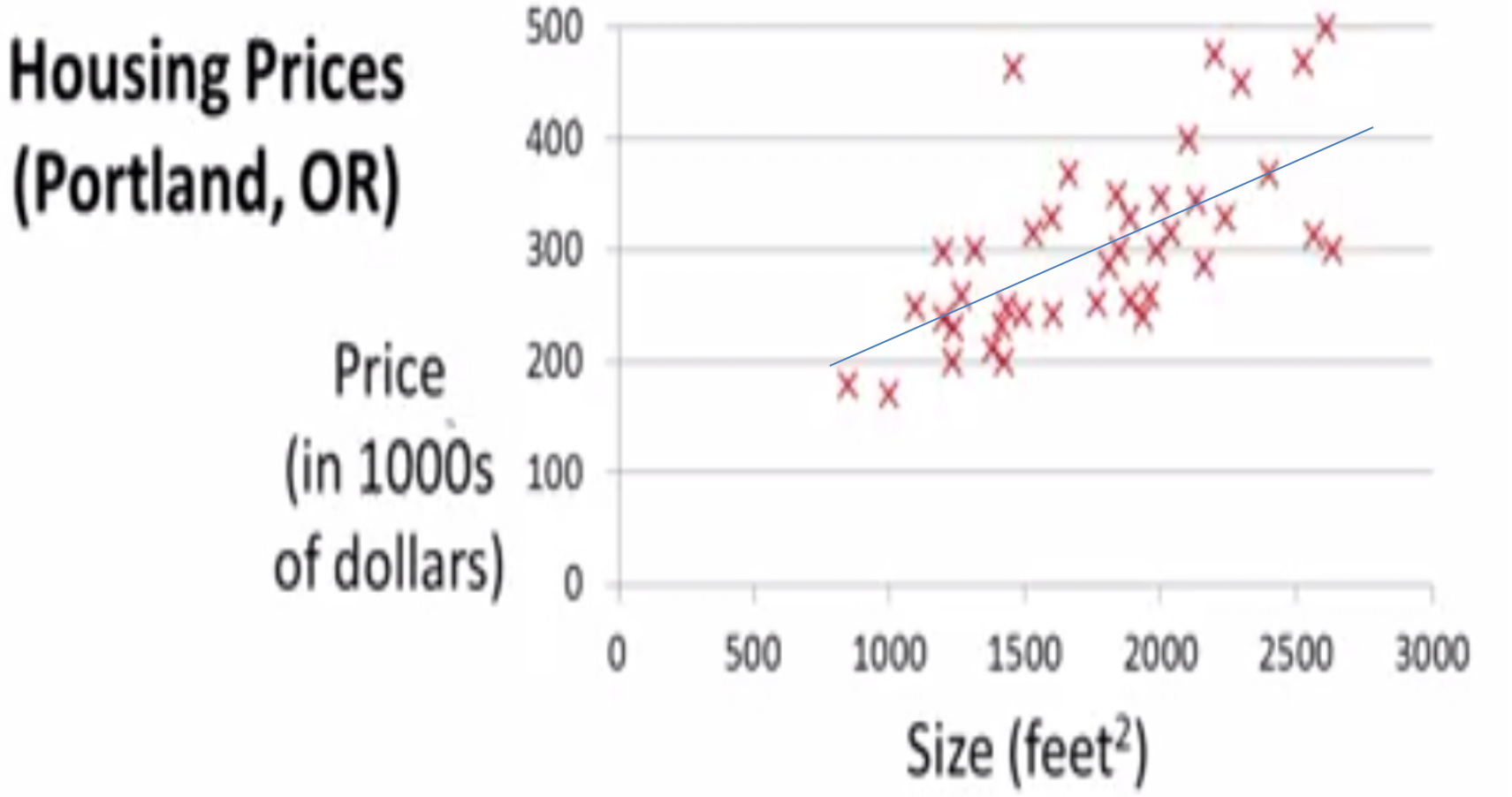
Why are we looking at this algorithm ?

- Machine Learning – a field of predictive modelling
- Linear Regression
  - Developed in the field of statistic
  - A model that attempts to show the relationship between two variables with a linear equation
  - Borrowed by Machine Learning
- Finds it's application in
  - Evaluating Trends and Sales Estimates
  - Analyzing the Impact of Price Change and many more

LINEAR REGRESSION

- Basic Framework
  - Dependent variable **y** , also called the output/explained variable which is to be predicted
  - Independent variables $\mathbf{x_i}$, also called the input/explanatory variables that is to be used for making predictions.
- Regression is the general task of attempting to predict the value of output variable **y** from the input variables $\mathbf{x_i}$.

MODEL REPRESENTATION



MODEL REPRESENTATION

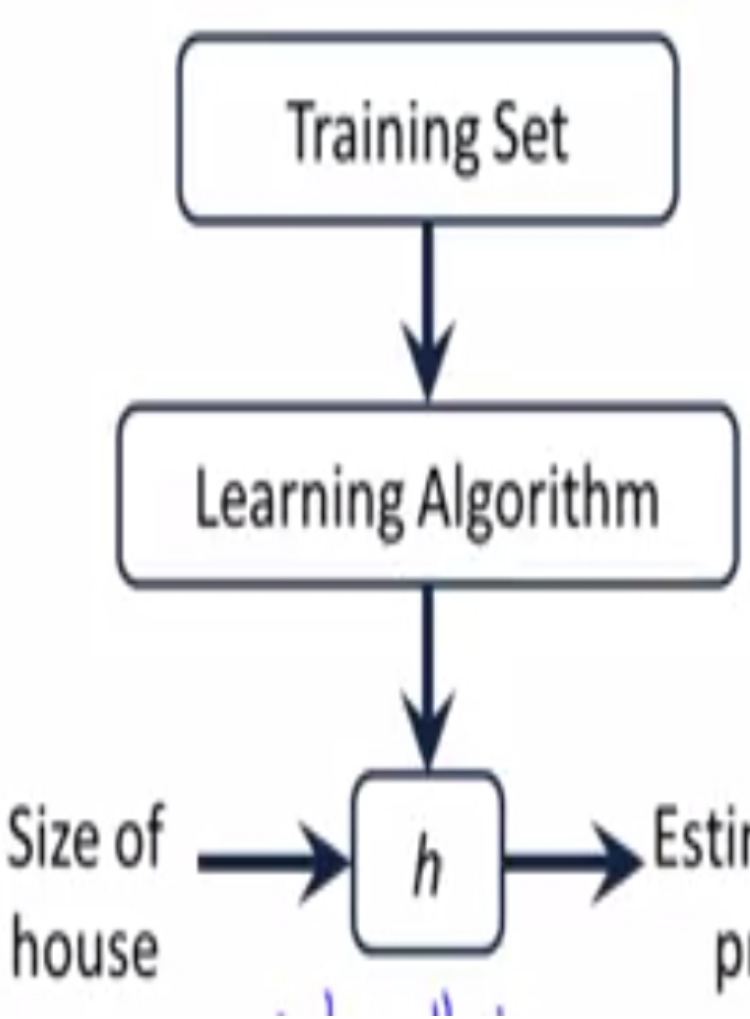
Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

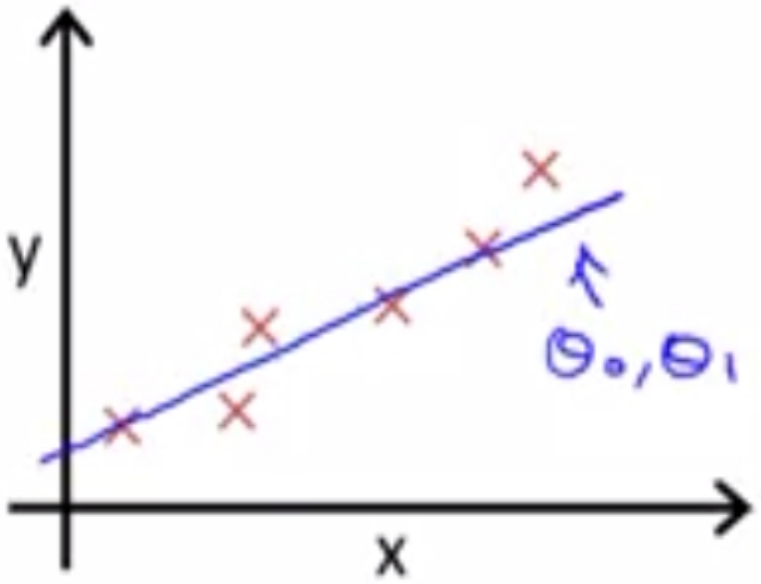
Notation:

- **m** = Number of training examples
- **x**'s = "input" variable / features
- **y**'s = "output" variable / "target" variable

MODEL REPRESENTATION



How do we represent h ?



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

MODEL REPRESENTATION

Given a training set ,

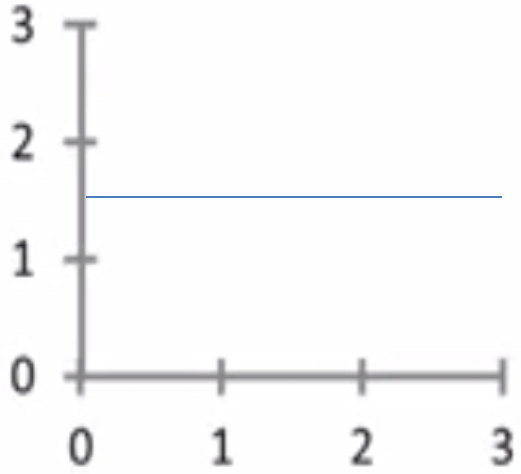
Hypothesis : $h_\theta(x) = \theta_0 + \theta_1 x$

where $\theta_i's : \text{Parameter}$

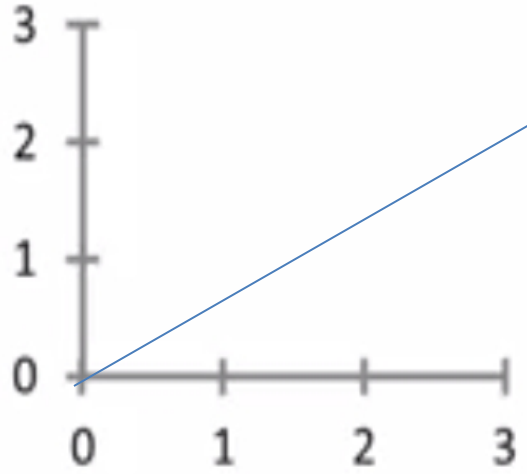
How to choose $\theta_i's$

MODEL REPRESENTATION

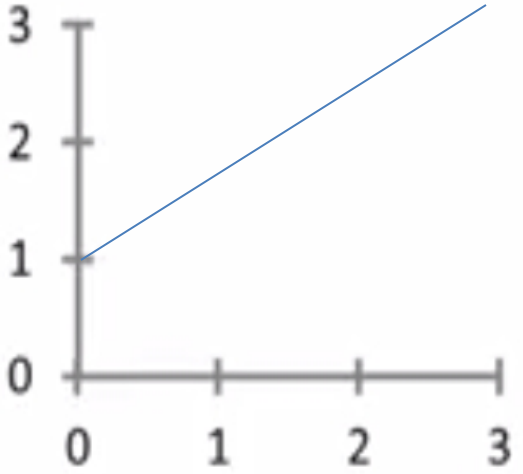
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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

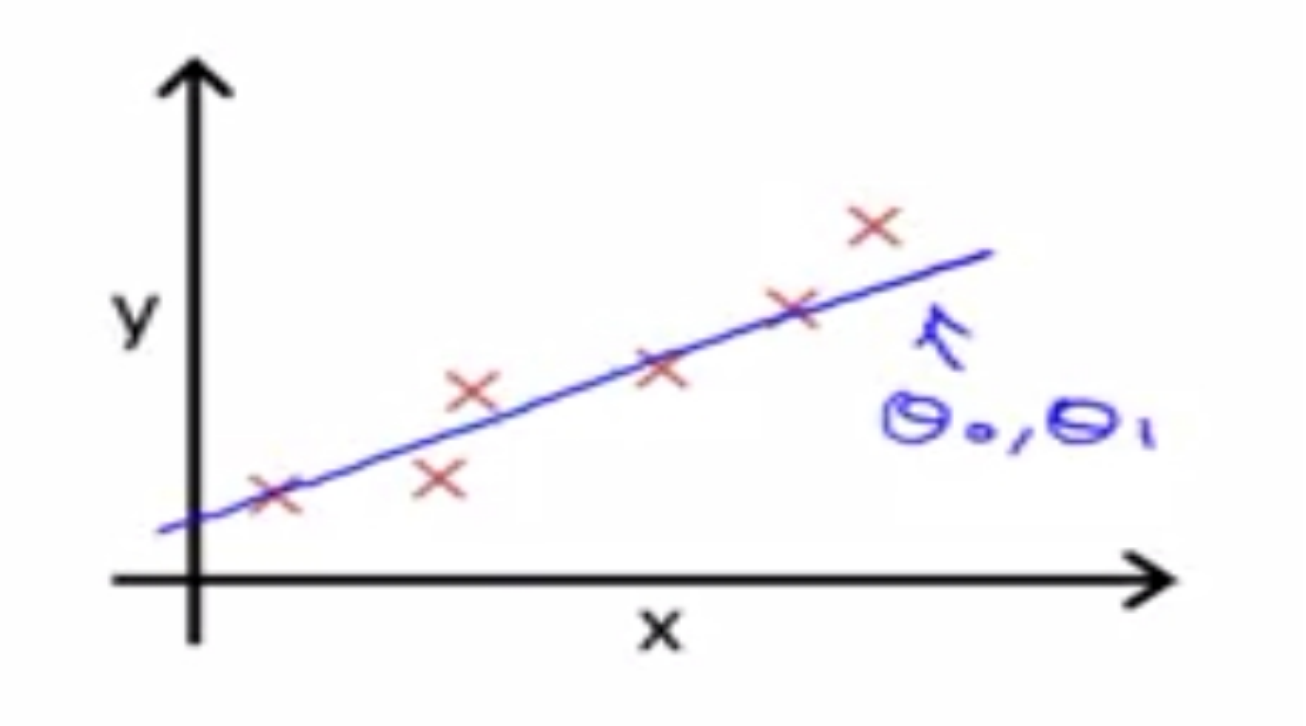


$\theta_0 = 0$
$\theta_1 = 0.5$



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

COST FUNCTION



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

COST FUNCTION

So going with the idea ,

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

We want to minimise the squared error function

i.e. $(\mathbf{h_\theta}(\mathbf{x^{(i)}}) - \mathbf{y^{(i)}})^2$ i.e the difference between the predicted value ($h_\theta(x)$) and the actual label (**y**) should be as minimum as possible.

COST FUNCTION

Therefore, the cost function becomes

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

where summation(Σ) represents sum over my training set from i to m .

and where (${}^{1}/_{2m}$) represents that minimising one half of something shall give us same values of θ_0 and θ_1 as minimising the entire thing.

COST FUNCTION

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

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

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

So, overall we want to find the value of θ_0 & θ_1 that minimises the entire expression.

GRADIENT DESCENT ALGORITHM

- Gradient Descent algorithm is basically used to minimise some function J(say a cost function).
- Problem Setup
  - Have some function J(,)
  - Want to θ_0 θ_1
- Outline $\min_{\theta_0,\theta_1} J(\theta_0,\theta_1)$
  - Start with some & (say = 0 , = 0)
  - Keep changing θ_0 θ_1 to reduce J(,) until we hopefully end up at a minimum θ_0 θ_1 θ_0 θ_1

GRADIENT DESCENT ALGORITHM

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta_0, \theta_1) \quad , \quad (\text{for } j = 0 \ \& \ j = 1) \ \}$$

where **:=** is the assignment operator

α is the learning rate that controls how big a step we can take downhill while creating descent

- Simultaneous update

$$\text{temp } 0 := \theta_0 - \alpha \frac{\delta}{\delta \theta_0} J(\theta_0, \theta_1)$$

$$\text{temp } 1 := \theta_1 - \alpha \frac{\delta}{\delta \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp } 0$$

$$\theta_1 := \text{temp } 1$$

GRADIENT DESCENT INTUITION

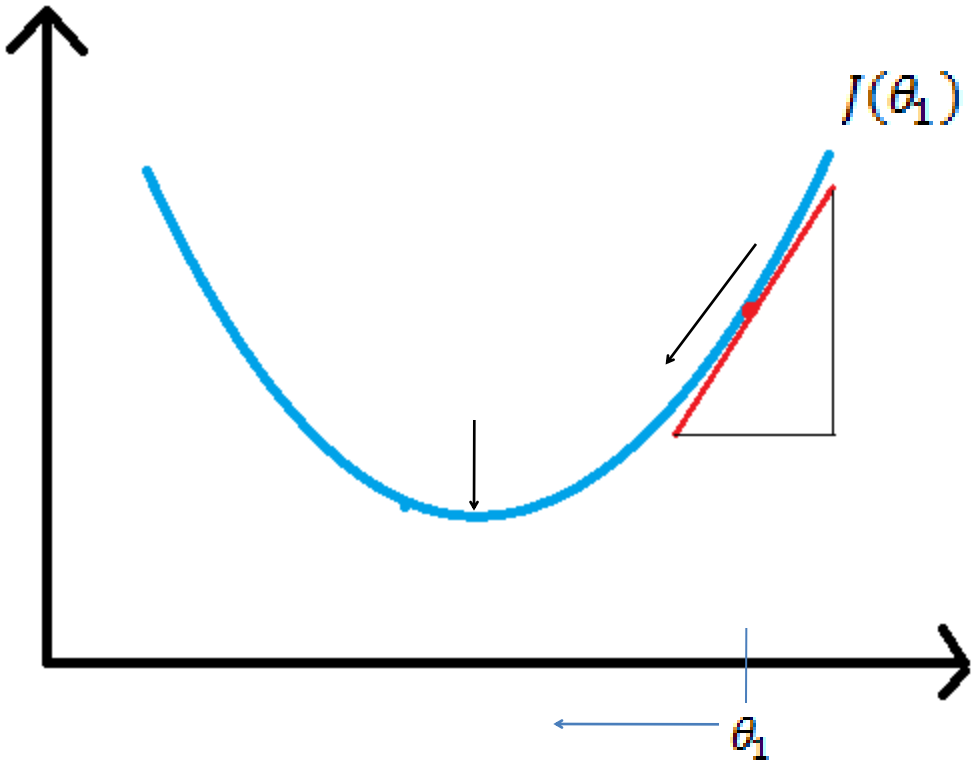
For eg :) say we have only one parameter, so,

$\min_{\theta_1} J(\theta_1)$ & $\theta_1 \epsilon R$, hence the GD equation becomes

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

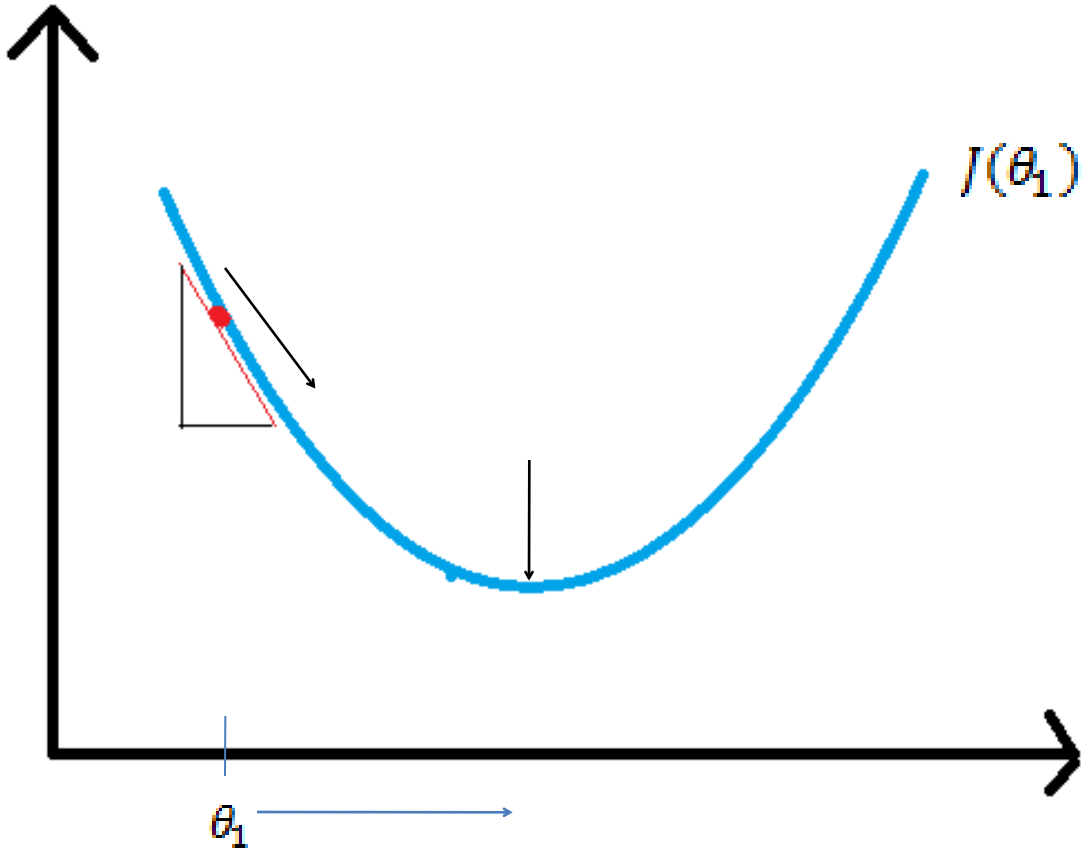
$\frac{d}{d\theta_1} J(\theta_1)$ is the derivative term which basically denotes the slope of the line that is just tangent to the function $J(\theta_1)$ at the point θ_1

GRADIENT DESCENT INTUITION



$$\frac{d}{d\theta_1} J(\theta_1) \geq 0 \qquad \theta_1 := \theta_1 - \alpha\,(positive\ no.)$$

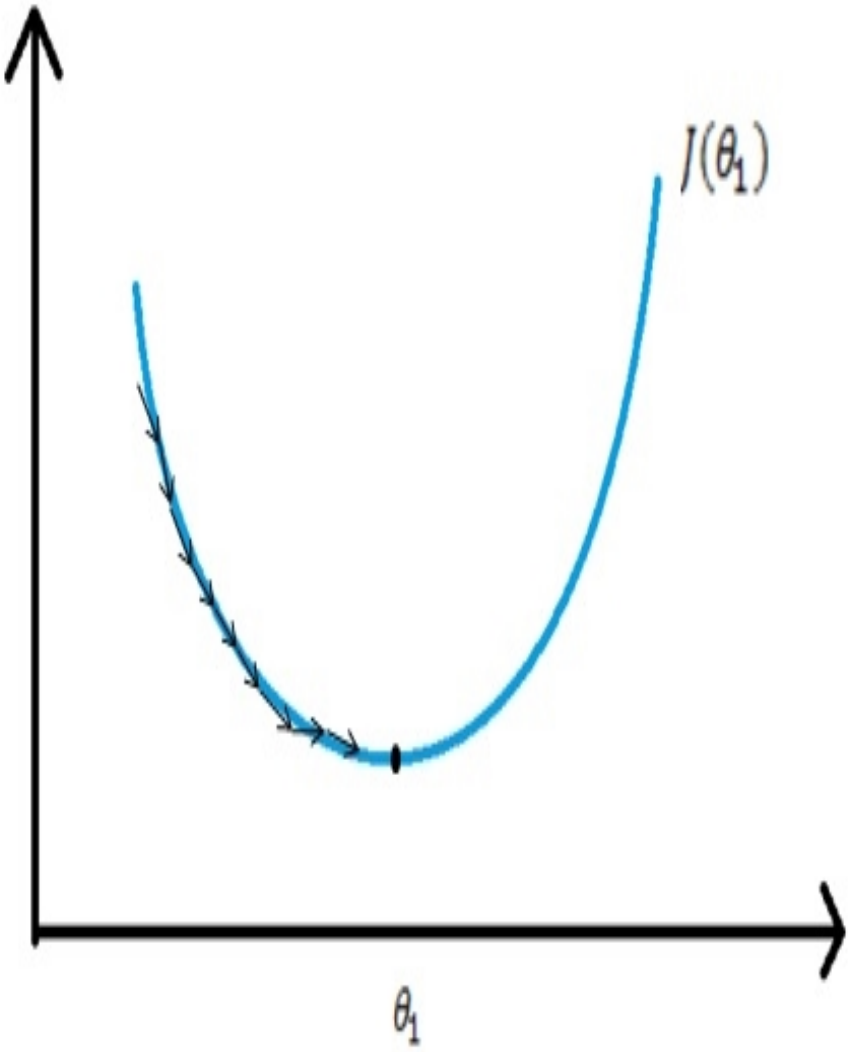
GRADIENT DESCENT INTUITION



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

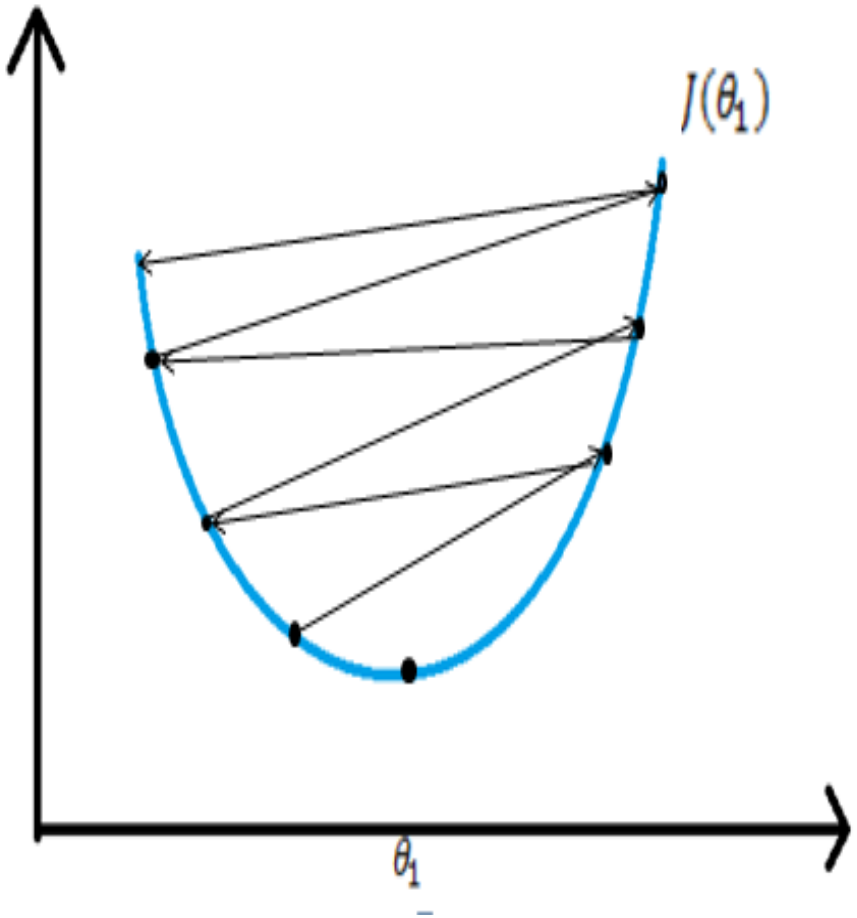
$$\theta_1 := \theta_1 - \alpha\,(negative\ no.)$$

GRADIENT DESCENT INTUITION



If α is too small , then gradient descent will converge slowly because very small and hence lot of steps are being taken before it gets anywhere close to the global minimum.

GRADIENT DESCENT INTUITION



If α is too large, then gradient descent will overshoot the minimum because the steps taken are huge . It may fail to converge .

GRADIENT DESCENT FOR LINEAR REGRESSION

So, applying GD algorithm to minimise the squared error cost function i.e. $\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$

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

&

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

Calculating the derivative part,

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

$$\frac{\delta}{\delta\theta_j} J(\theta_0, \theta_1) = \frac{\delta}{\delta\theta_j} {}^{1}/_{2m} \left(\sum_{i=1}^{m} (\theta_0 + \theta_1 x - y^{(i)})^2\right)$$

GRADIENT DESCENT FOR LINEAR REGRESSION

So for θ_0 , or **j** = 0 :

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

for θ_1 , or **j** = 1 :

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

GRADIENT DESCENT FOR LINEAR REGRESSION

Now , by applying GD algorithm to the previous equation(derivative part) ,

repeat until convergence {

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

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

} update θ_0 & θ_1 simultaneously

FEATURE SCALING

- If one feature has a range say 0-2000 & another say 0-5 i.e. a 2000 to 5 ratio, then the contours of the cost funtion takes up a very skewed elliptical shape & the GD may end up taking a long time & can oscillate back & forth before it can finally find its way to the global minimum.
- A useful thing to do is scale the features i.e. the different features take on similar range of values.
- So, by feature scaling we get the features in a range of **-1** to **+1** .

LINEAR REGRESSION WITH MULTIPLE VARIABLES

for multiple features ,

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots\dots\dots + \theta_n x_n$$

For convenience of notation , define $x_0 = 1$

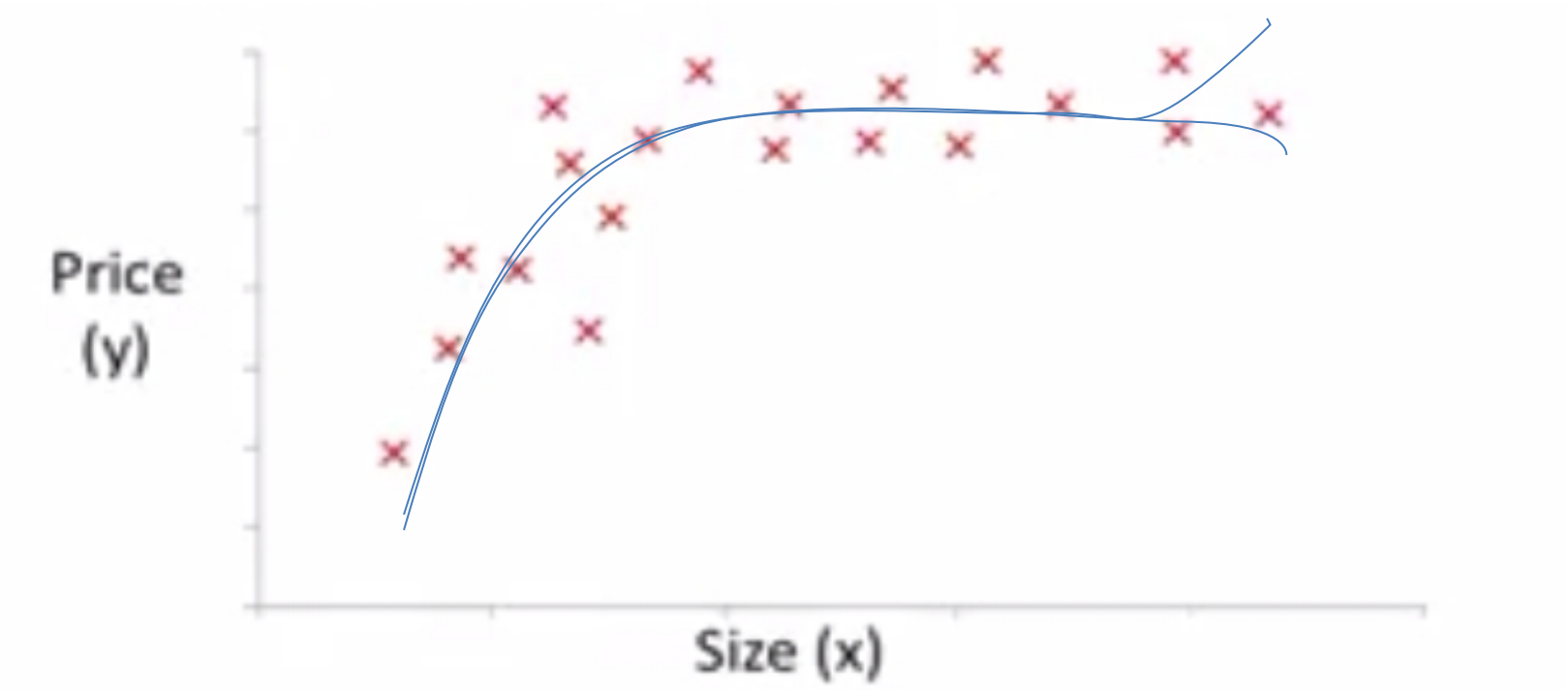
$$X = \begin{bmatrix} x_0 \\ x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad , \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdot \\ \cdot \\ \cdot \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

So, $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots\dots\dots + \theta_n x_n$

$$\therefore \quad h_\theta(x) = \theta^T x$$

POLYNOMIAL REGRESSION

- It allows to use the machinery of linear regression to fit very complicated even non-linear function to the data.



Quadratic model : $\theta_0 + \theta_1 x + \theta_2 x^2$

Cubic model : $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

POLYNOMIAL REGRESSION

The form of hypothesis is :

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$$h_\theta(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3$$

where $x_1 = \text{size}$, $x_2 = (\text{size})^2$, $x_3 = (\text{size})^3$

- Feature Scaling is required here

NORMAL EQUATION

- GD is basically an algorithm for linear regression that takes multiple iterations to reach the global minimum or to minimise a cost function $J(\theta)$.
- Normal Equation is a method to solve for θ analytically i.e. in one step we get the optimum value .
- Intuition : If 1D , $(\theta \in \mathbb{R})$

$$J(\theta) = a\theta^2 + b\theta + c$$

To minimise the above quadratic function, we calculate the derivative of the function and set it to 0 i.e. $\frac{d}{d\theta} J(\theta) = 0$ & solve for θ .

NORMAL EQUATION

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$\theta = (X^T X)^{-1} X^T Y$$

GRADIENT DESCENT Vs NORMAL EQUATION

GRADIENT DESCENT

- Need to choose α .
- Needs many iterations
- Works well even if n is large where n is the no. of features.

NORMAL EQUATION

- No need to choose α .
- Needs no iteration
- Slow if n is large because it needs to compute $(\mathbf{X}^{\mathbf{T}}\boldsymbol{X})^{-1}$ which is **n*n** matrix .