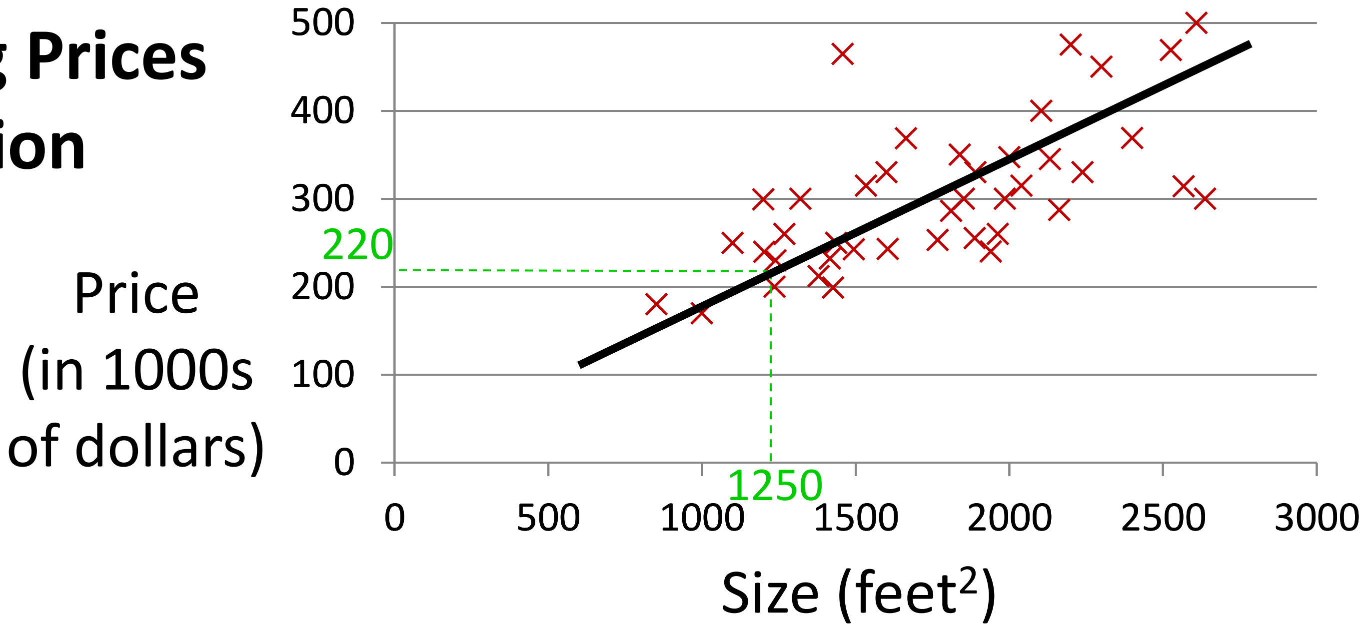
Types of Learning

Instructors: Dr. Durgesh Singh

CSE Discipline, PDPM IIITDM, Jabalpur -482005

Linear regression with one variable

Ex. Housing Prices prediction



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Classification : Discrete-valued output

Training set of housing prices

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

m

Notation:

m = Number of training examples

x's = "input" variable / features

y's = "output" variable / "target" variable

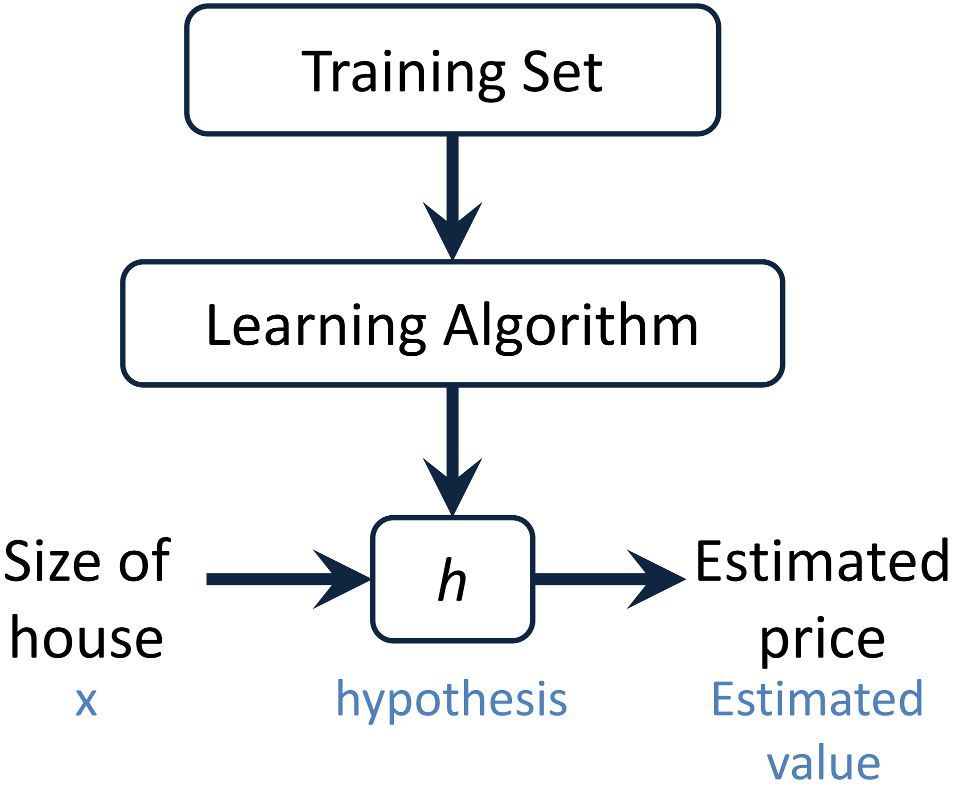
(x, y) – one training example

$(x^{(i)}, y^{(i)})$ – ith training example

$x^{(1)} = 2104$

$x^{(2)} = 1416$

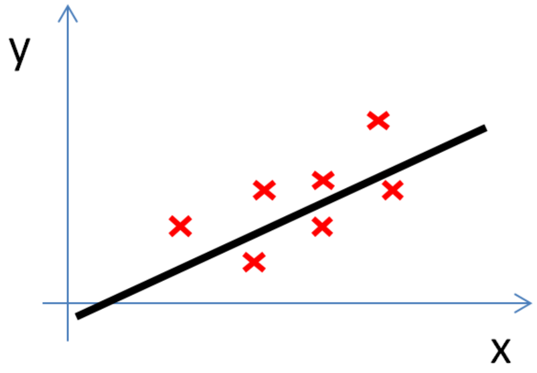
$y^{(1)} = 460$



h maps from x's to y's

How do we represent *h* ?

$h_\Theta(x) = \Theta_0 + \Theta_1 x$



Linear regression with one variable.
Univariate linear regression.

One variable

Linear regression with one variable

Cost function

Training Set

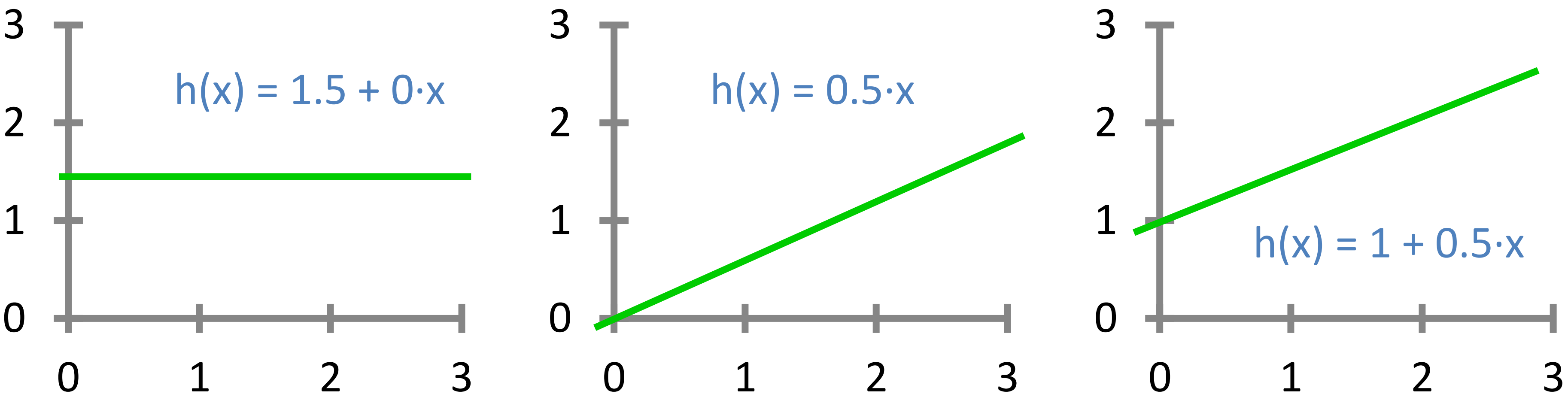
Size in feet^2 (x)	**Price ($) in 1000's (y)**
2104	460
1416	232
1534	315
852	178
…	…

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

θ_i's: Parameters

How to choose θ_i's ?

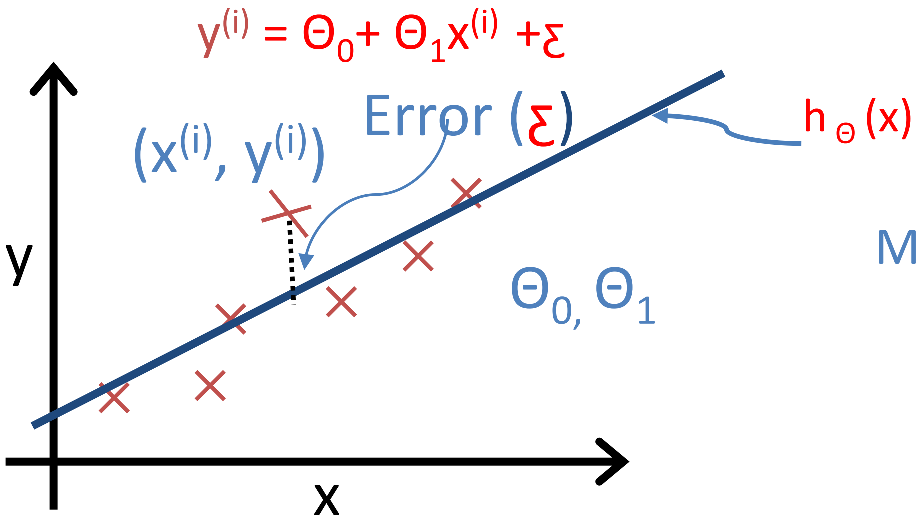
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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

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

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



$$\underset{\Theta_0\,\Theta_1}{\text{Minimize}} \ \frac{1}{2m}\sum_{i=1}^{m}\left(h_\Theta(x^{(i)}) - y^{(i)}\right)^2$$

$$h_\Theta(x) = \Theta_0 + \Theta_1 x^{(i)}$$

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\Theta(x^{(i)}) - y^{(i)}\right)^2$$

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

$\underset{\Theta_0\,\Theta_1}{\text{Minimize}}\ J(\Theta_0, \Theta_1)$: Cost Function

Squared error function

Simplified

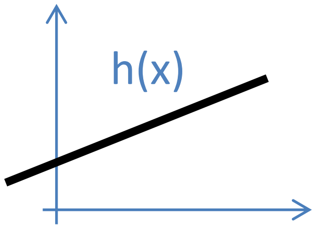
Hypothesis:

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

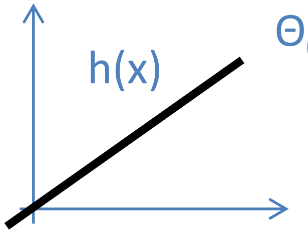
$$h_\theta(x) = \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$



$$\theta_1$$



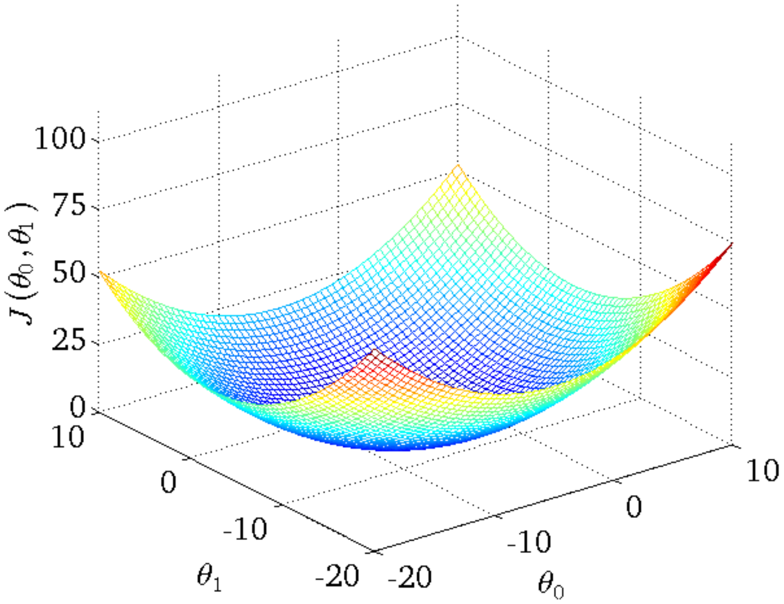
Cost Function:

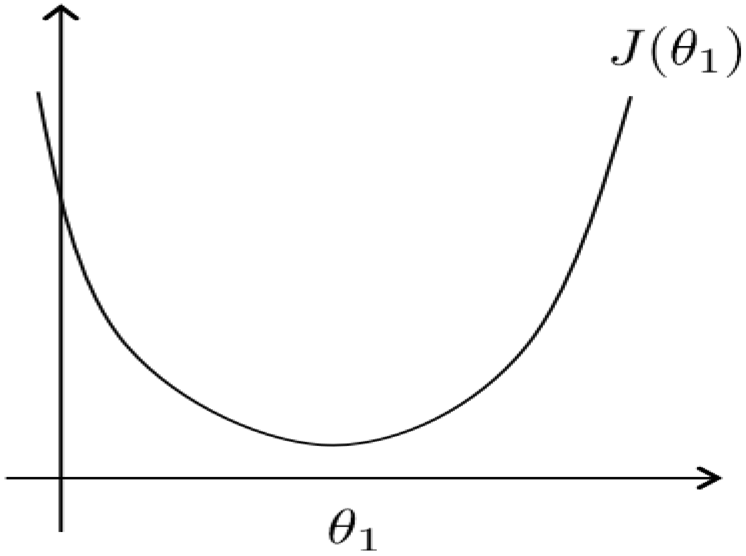
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

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

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

$$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$$


100
75
50
25
0
$J(\theta_0,\theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0


$J(\theta_1)$
θ_1

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

Linear regression with one variable

Gradient descent

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

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

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

Gradient descent algorithm

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$
$$\}$$

Learning rate

Simultaneously update
Θ_0 & Θ_1

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

Gradient descent algorithm

repeat until convergence {

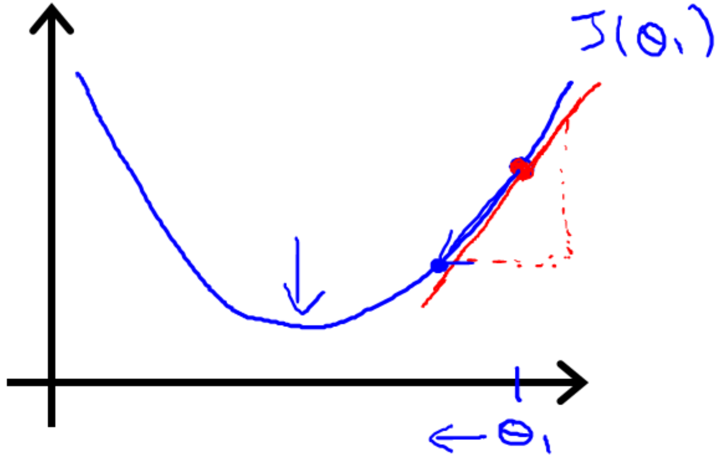
$$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}$$ (simultaneously update $j = 0$ and $j = 1$)

}

Learning rate

derivative

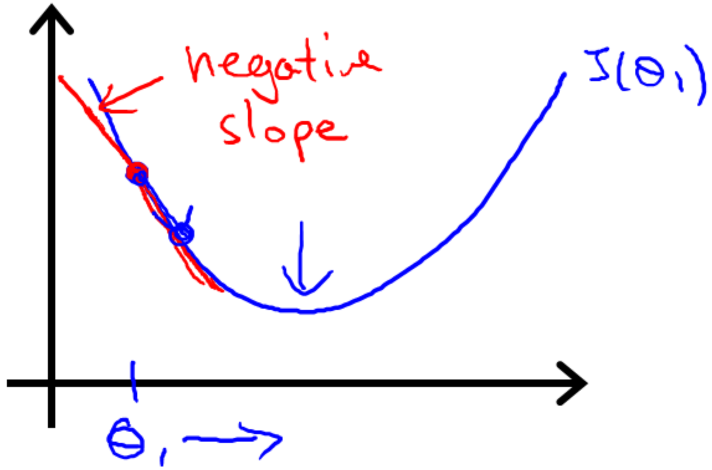
The partial derivative is the slop of the tangent line to this curve at that point.



$$\Theta_1 := \Theta_1 - \alpha \boxed{\frac{\partial}{\partial \theta_1} J(\theta_1)}$$

≥ 0

$$\Theta_1 := \Theta_1 - \alpha(positive\ number)$$



$$\Theta_1 := \Theta_1 - \alpha \boxed{\frac{\partial}{\partial \theta_1} J(\theta_1)}$$

≤ 0

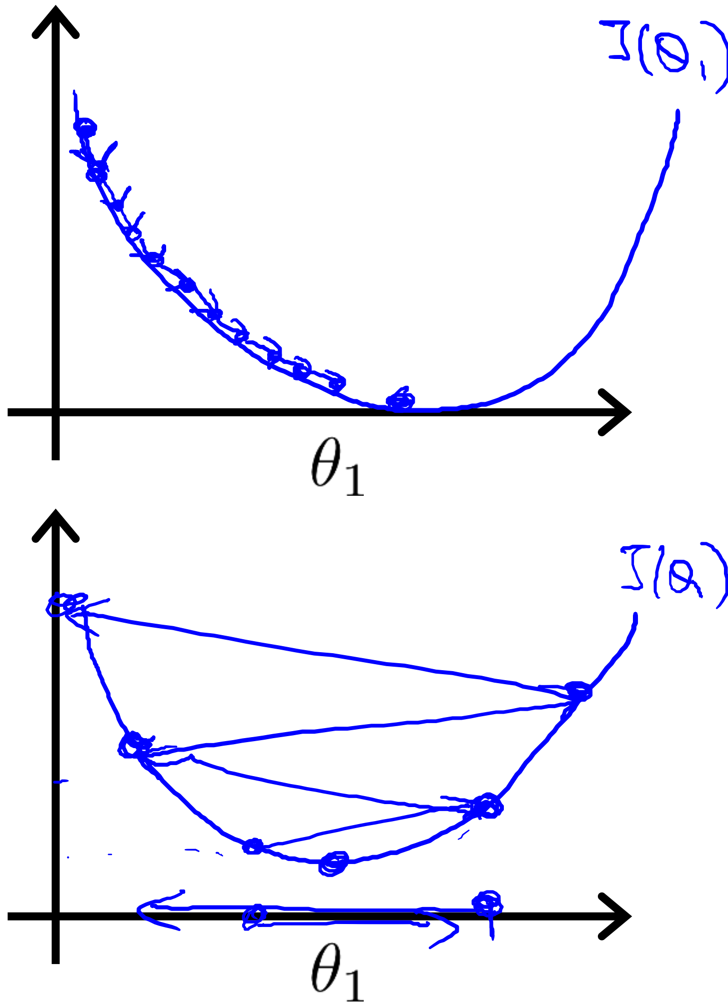
$$\Theta_1 := \Theta_1 - \alpha(negative\ number)$$

Role of Learning Rate

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

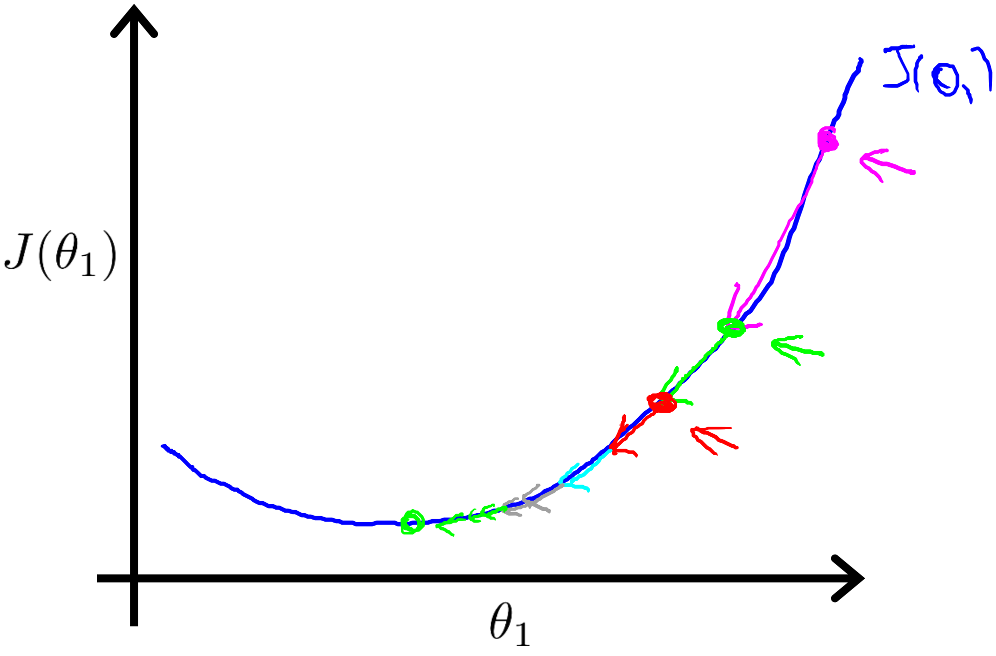
If α is too small, gradient descent can be slow.

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



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.

Linear regression with one variable

Gradient descent for linear regression

Gradient descent algorithm

repeat until convergence {

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

$$(\text{for } j = 1 \text{ and } j = 0)$$

}

Linear Regression Model

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

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

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

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

$$j = 0 : \frac{\partial}{\partial\theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})$$

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

Gradient descent algorithm

$$\text{repeat until convergence } \{$$

$$\frac{d}{d\theta_0} \cdot J(\theta_0, \theta_1)$$

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

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

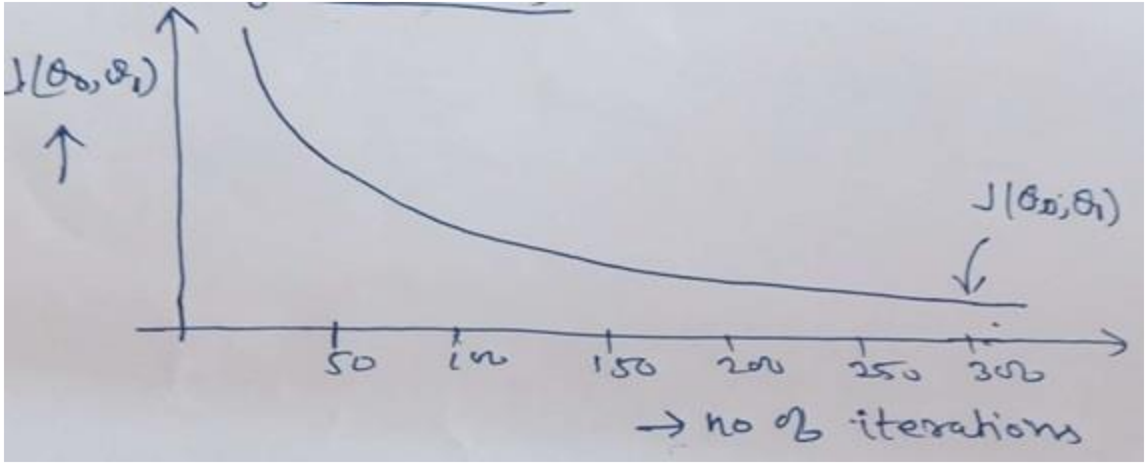
$$\}$$

$$\frac{d}{d\theta_1} \cdot J(\theta_0, \theta_1)$$

update θ_0 and θ_1 simultaneously

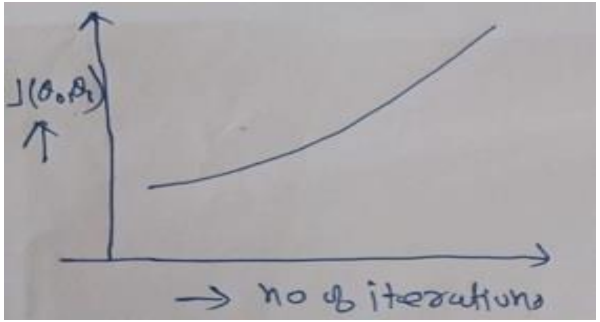
How to choose learning rate (α)

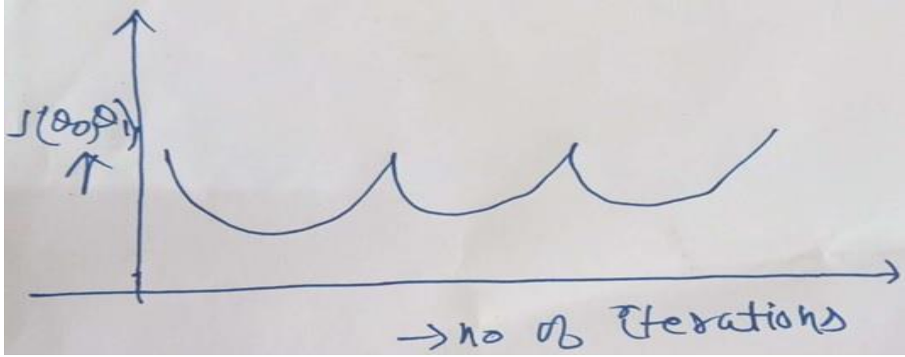
If gradient decent is working correctly then $J(\theta_0, \theta_1)$ should decrease after every iteration.



How to choose learning rate (α)

If gradient decent is not working correctly then..





- **Cause:** Large α
- **Solution:** use small α
- **To choose α, Try as,**

0.001, 0.003, 0.01, 0.03, 0.1, …….., 1.0

Linear Regression with Multiple Variable

- Ex. House Price Prediction

Size in feet^2	No. of bed rooms	No. of floors	Age of House in Years	Price ($) in 1000's (y)
2104	5	1	45	450
1416	3	2	40	232
1534	3	2	30	315
….	…	-----		…
…..	….	----		….

Linear Regression with Multiple Variable

Ex. House Price Prediction

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example

$x_j^{(i)}$ =value of feature j in i^{th} training example.

Hypothesis

Simple Linear Regression: $h_\theta(x) = \theta_0 + \theta_1 x$

For multi-variate linear regression:

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

- For convenience of notation, define $x_0 = 1$
- Hypothesis:

$$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Gradient descent

Repeat $\{$

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

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Gradient Descent

Previously (n=1):

Repeat $\{$

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

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

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

simultaneously update θ_j for $j = 0, \ldots, n$

$\}$

Gradient Descent

Previously (n=1):

Repeat {

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

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

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

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

simultaneously update θ_j for $j = 0, \ldots, n$

}

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

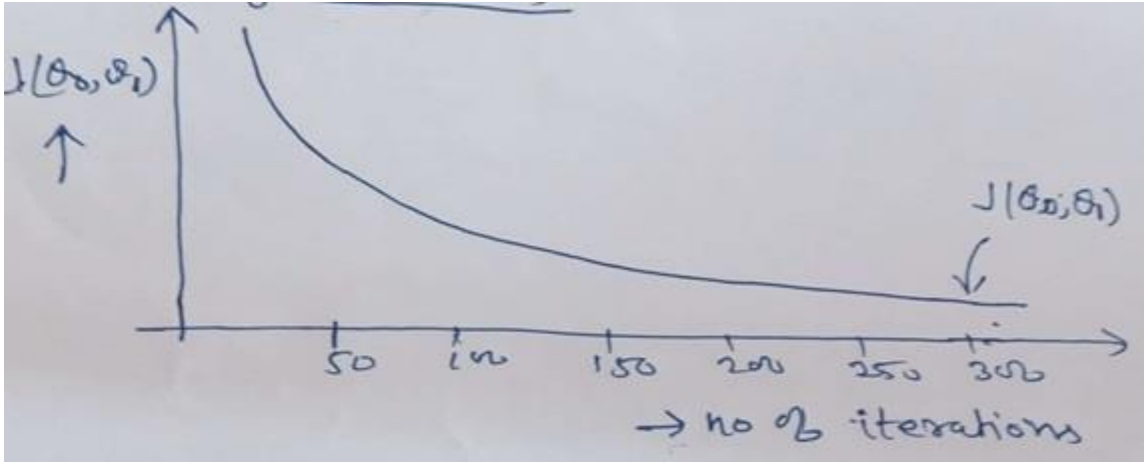
...

Is gradient descent working properly?

- Plot how $J(\theta)$ changes with every iteration of gradient descent
- For sufficiently small learning rate, $J(\theta)$ should decrease with every iteration
  - If not, learning rate needs to be reduced
- However, too small learning rate means slow convergence

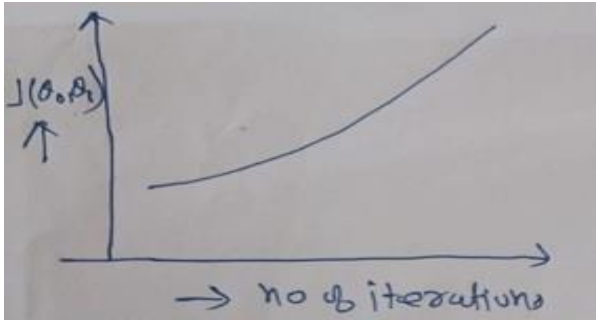
How to choose learning rate (α)

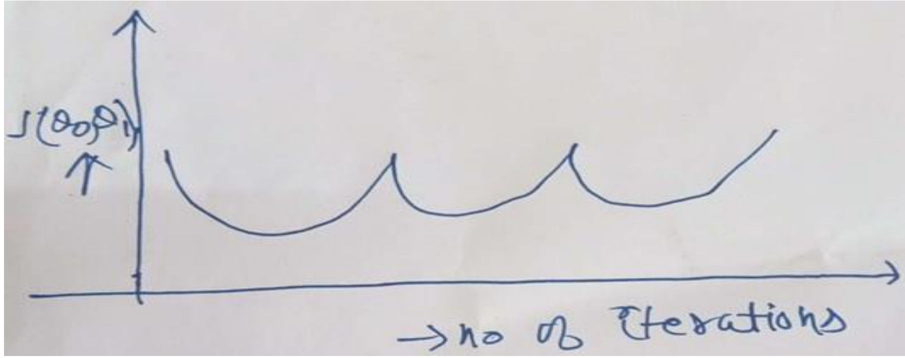
If gradient decent is working correctly then J(θ) should decrease after every iteration.



How to choose learning rate (α)

If gradient decent is not working correctly then..





- **Cause:** Large α
- **Solution:** use small α
- **To choose α, Try as,**

0.001, 0.003, 0.01, 0.03, 0.1, 1.0


Thank You
Gracias
Danke
Merci
Grazie