Linear regression

Ngô Minh Nhựt

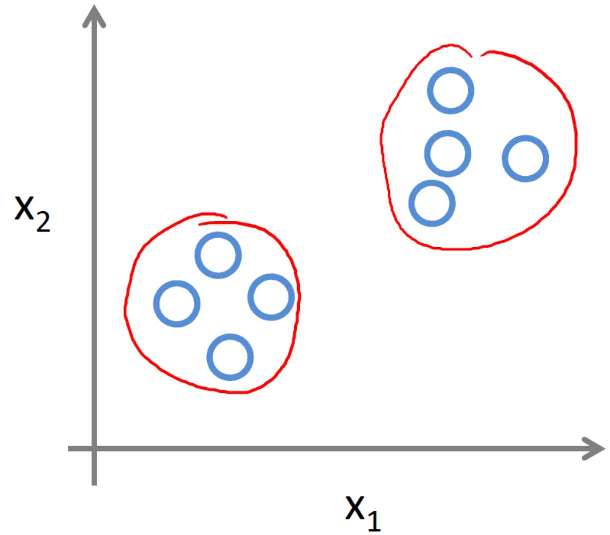
2021

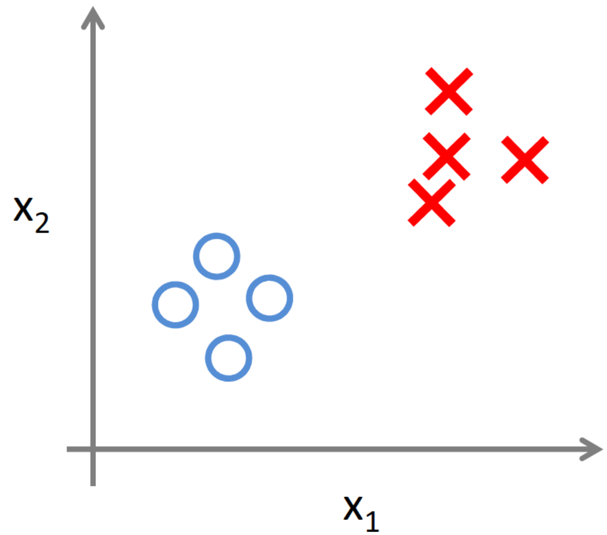
Linear regression with one variable

Part 1

Machine learning

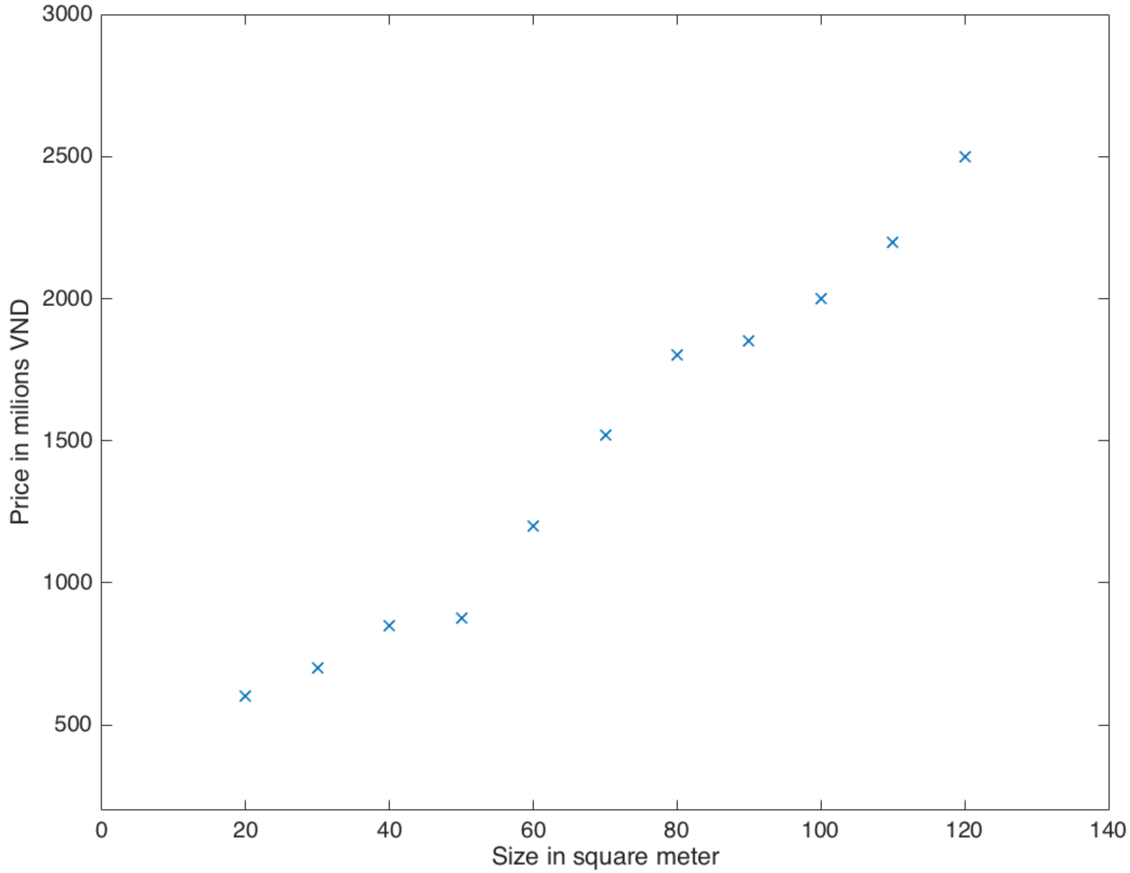
- Machine learning is about to find out structure of observed data or relationship inside them





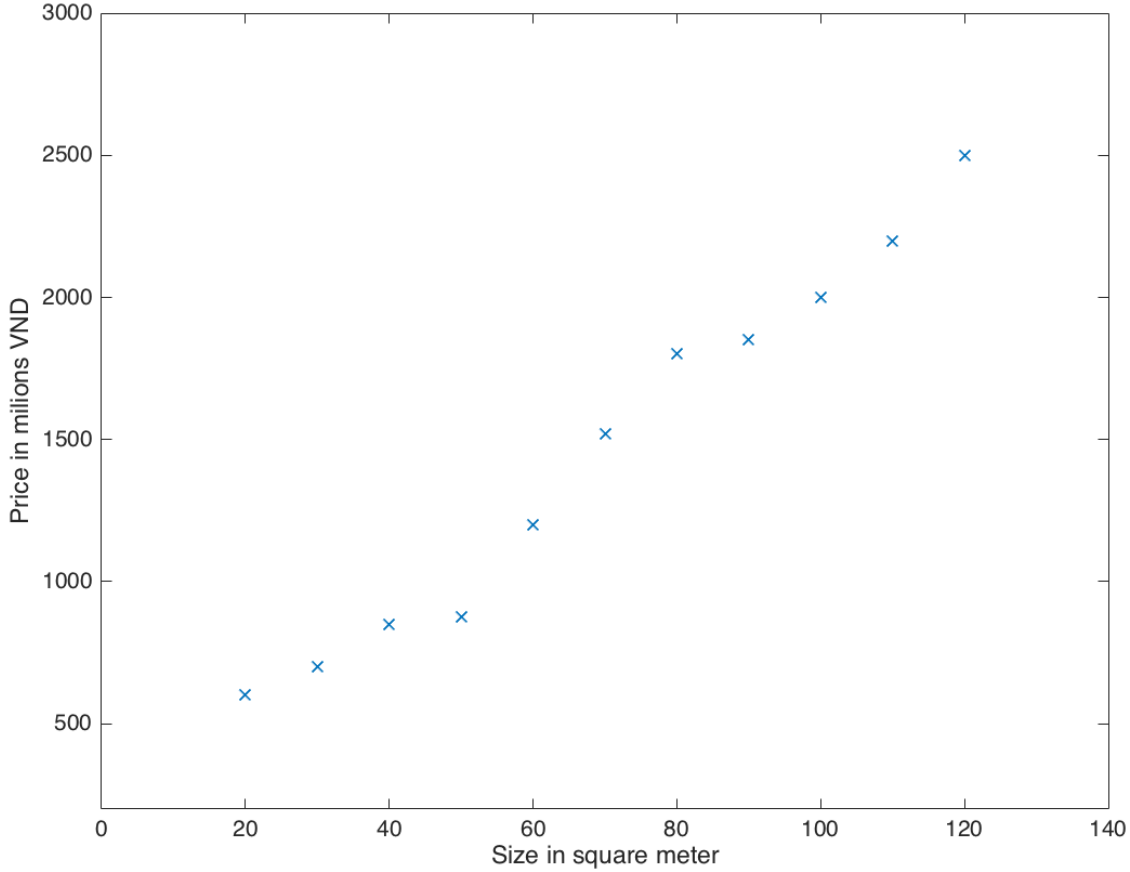
Source: Andrew Ng

Supervised learning



- Learn: provided with input and corresponding output
- Inference: infer output for new input

Linear regression



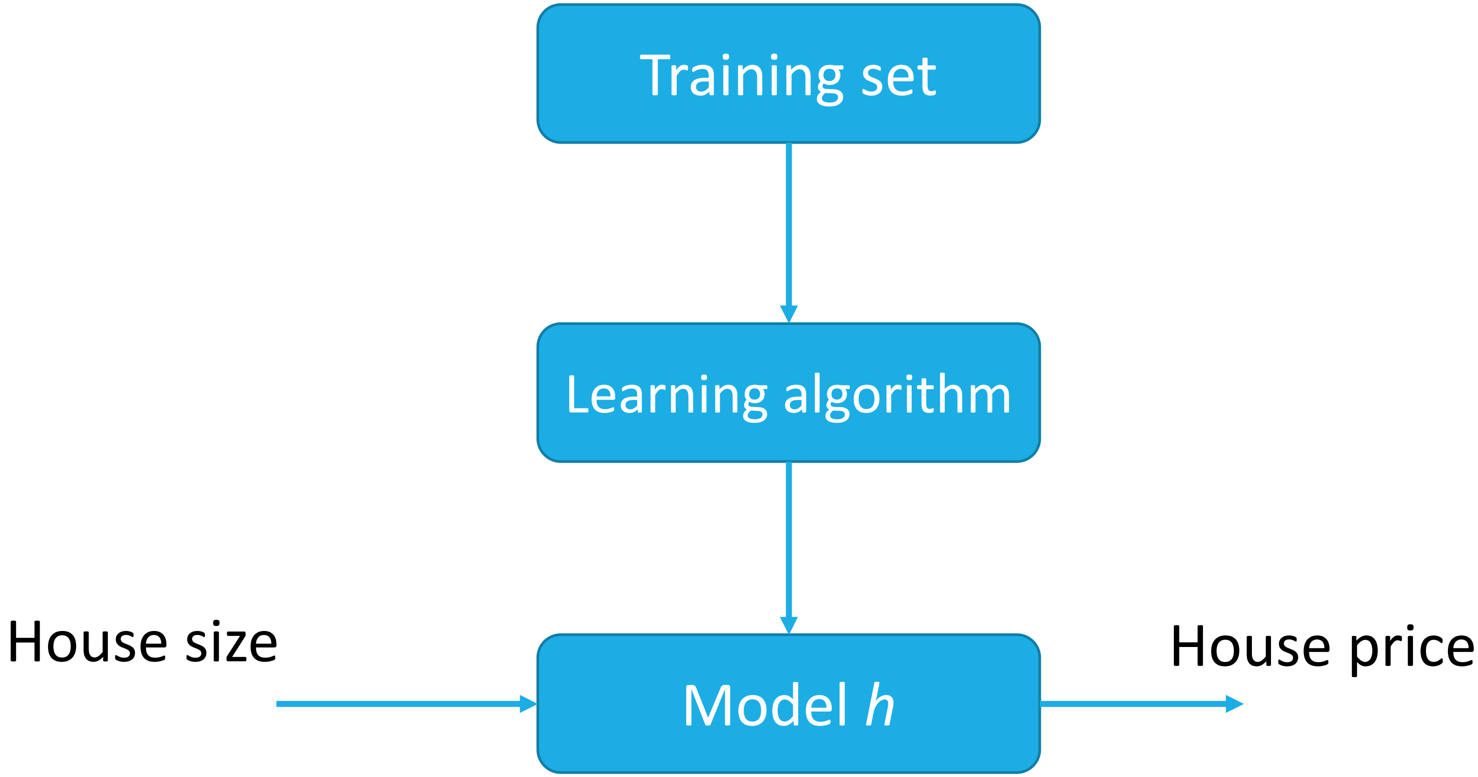
- Learn: provided with input and corresponding output
- Inference: infer output for new input
- **Output: continuous real value**

Training set

House price with respect to size

x: Size (m2)	y: Price (millions VND)
20	600
50	876
80	1800
100	2000
…	…

- m: number of samples
- x: input
- y: output/label
- (x, y): training sample
- $(x^{(i)}, y^{(i)})$: i^{th} sample



- h: mapping from size to price
- Linear regression: linear mapping
- In this part: linear regression with one variable

Hypothesis

- Training set: $\left(x^{(i)}, y^{(i)}\right),\ i = 1, 2, \dots m$
- Hypothesis : $h(x) = \theta_0 + \theta_1 x$
  - x: input
  - h(x): output
- Goal: identify θ_0 and θ_1 so that model $\mathrm{h(x)}$ fits with training set most
  - Given x, identify $h(x)$, so that $h(x)$ closes to y most
    - x: input
    - h(x): estimated output
    - y: realistic output

Hypothesis

House price with regard to size

x: Size (m2)	y: Price (millions VND)
20	600
50	876
80	1800
100	2000
…	…

Hypothesis : $h(x) = \theta_0 + \theta_1 x$

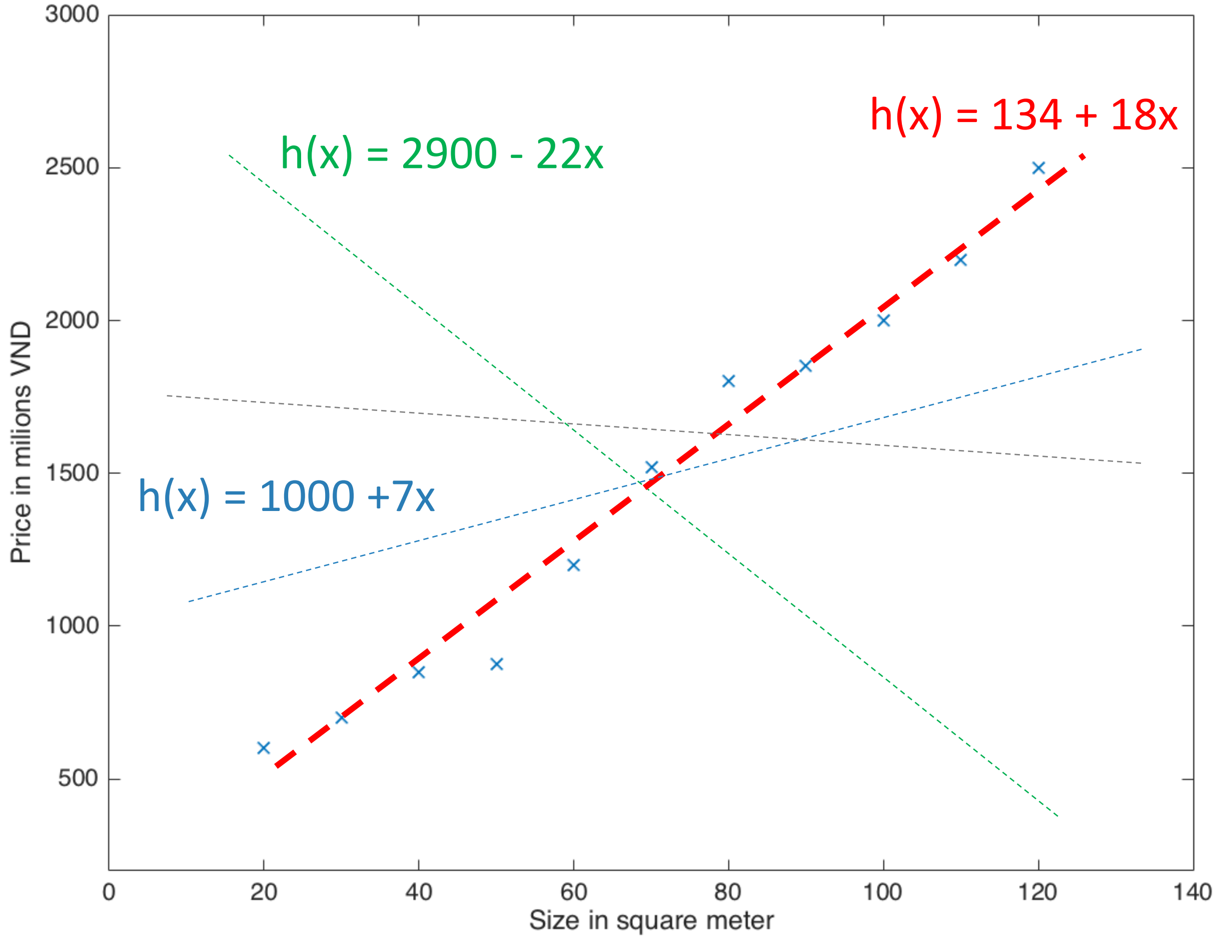
x: input

h(x): output

θ_0, θ_1: parameters

Learning: find out $\boldsymbol{\theta_0}, \boldsymbol{\theta_1}$ from training set

Hypothesis


h(x) = 134 + 18x
h(x) = 2900 - 22x
h(x) = 1000 +7x
Price in milions VND
Size in square meter
3000
2500
2000
1500
1000
500
0
20
40
60
80
100
120
140

Cost function

- Model error with one sample
  - $\frac{1}{2}(h(x)-y)^2 = \frac{1}{2}(\theta_0+\theta_1 x - y)^2$
- Cost function
  - $\mathrm{J}(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h\left(x^{(i)}\right)-y^{(i)}\right)^2$
  - $\mathrm{J}(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_0+\theta_1 x^{(i)}-y^{(i)}\right)^2$
- Goal:
  - Identify θ_0, θ_1 so that $J(\theta_0,\theta_1)$ reaches minimal

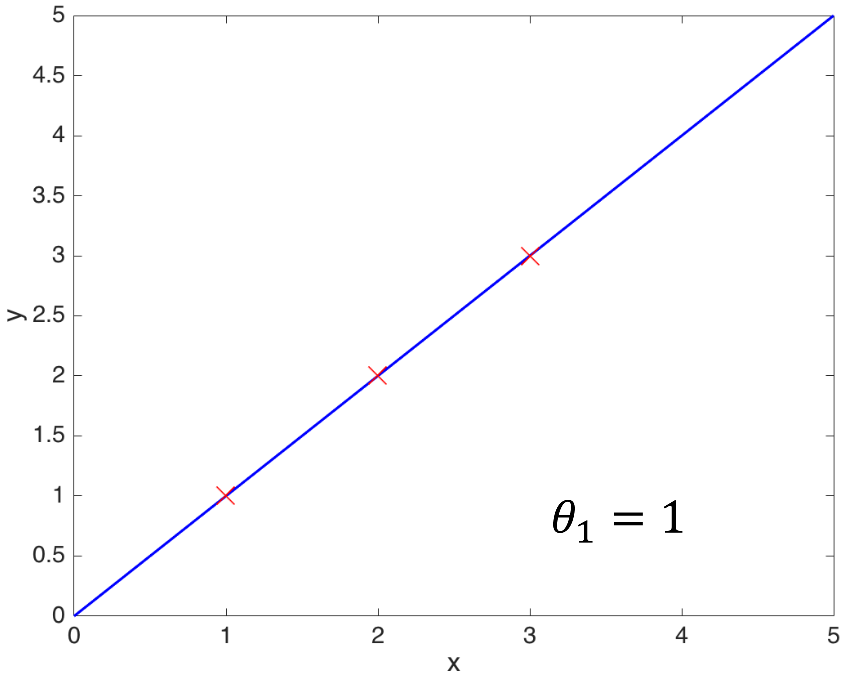
Cost function

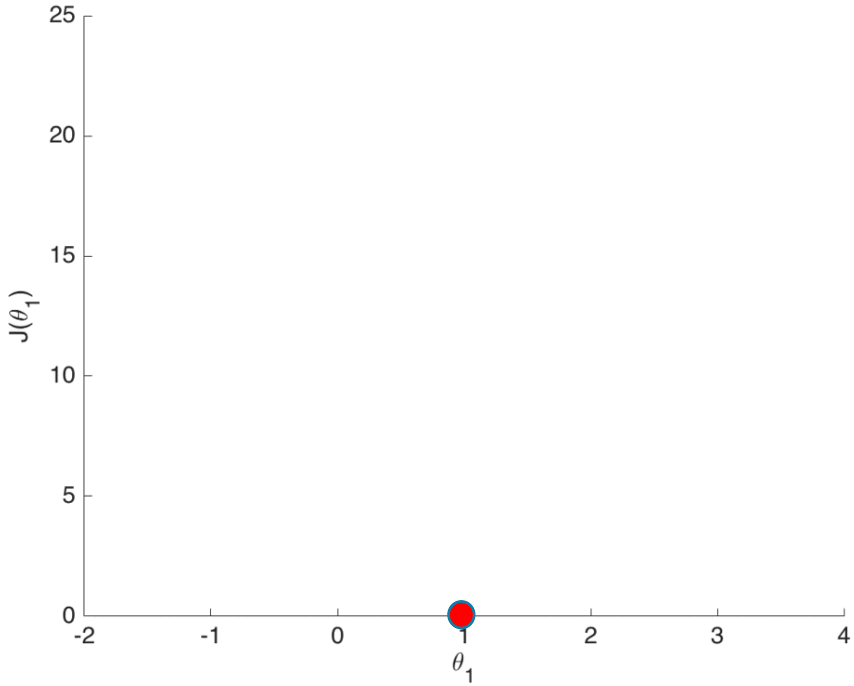
- Hypothesis : $h_\theta(x) = \theta_0 + \theta_1 x$
- Parameters: θ_0, θ_1
- Cost function:

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

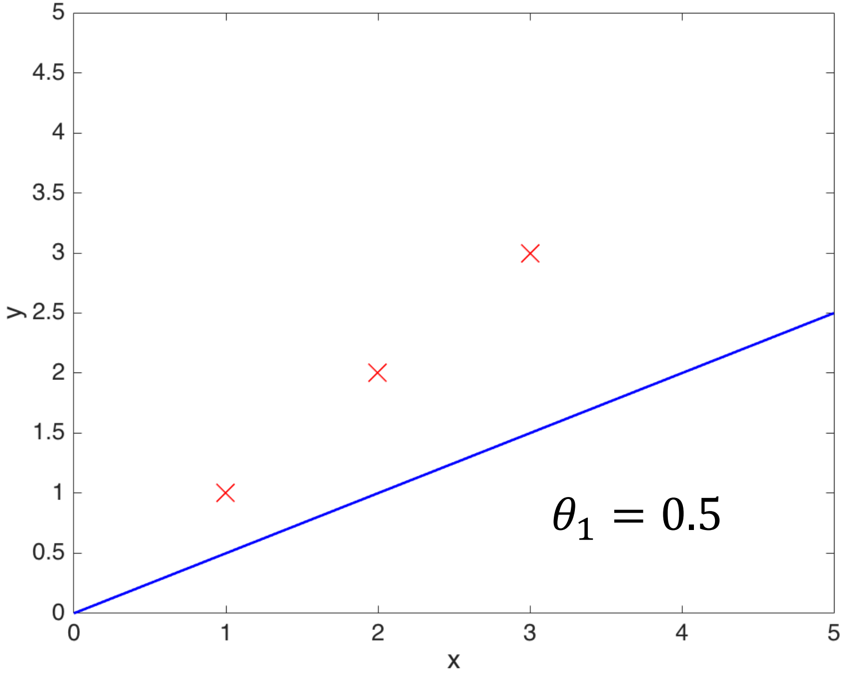
- Goal: find out θ_0, θ_1 so that $\mathrm{J}(\theta_0, \theta_1)$ reaches minimal
  - $Minimize_{\theta_0,\theta_1}\mathrm{J}(\theta_0, \theta_1)$
- To demonstrate: $\theta_0 = 0,\ h_\theta(x) = \theta_1 x$
  - $\mathrm{J}(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

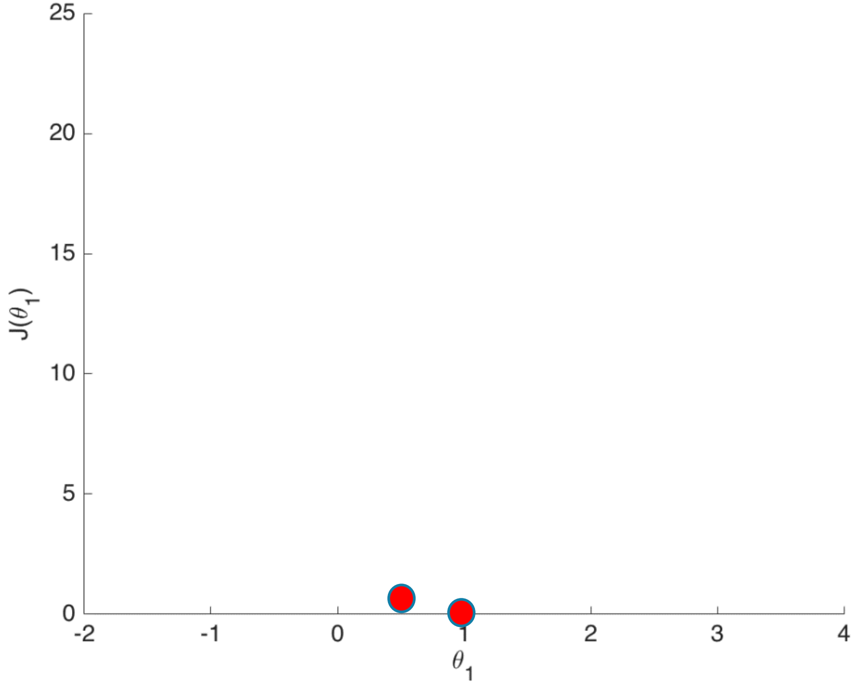
Cost function



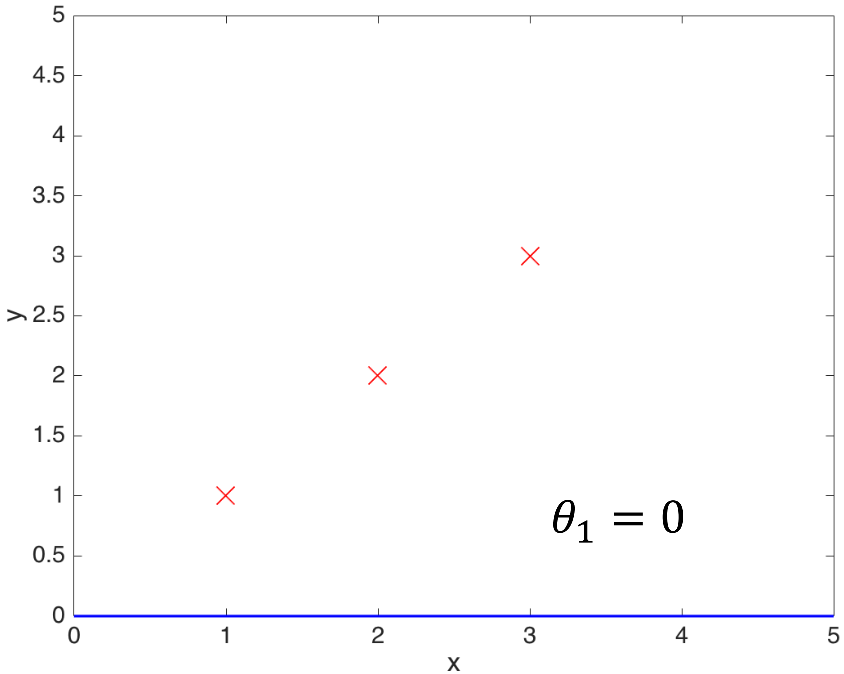


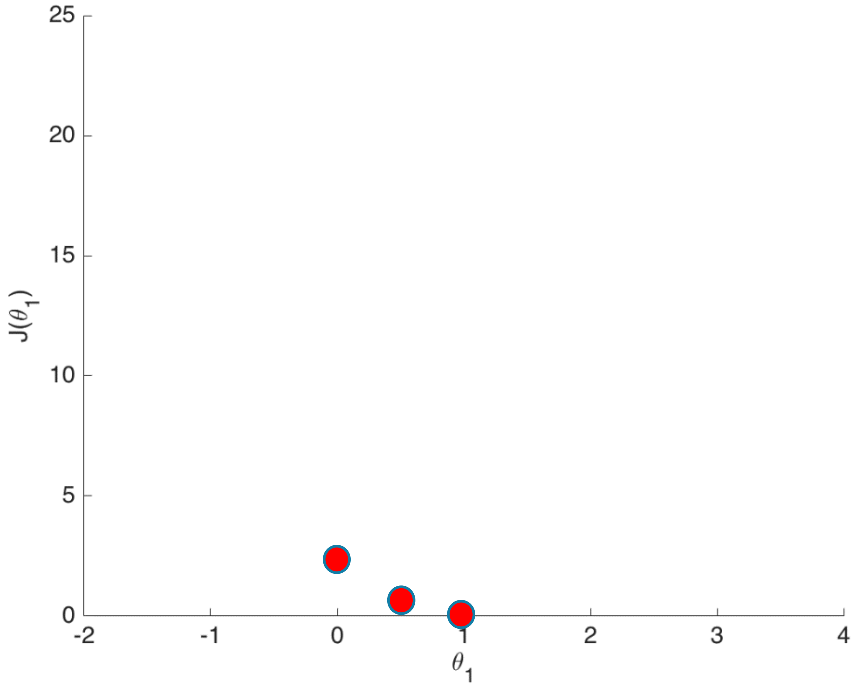
Cost function



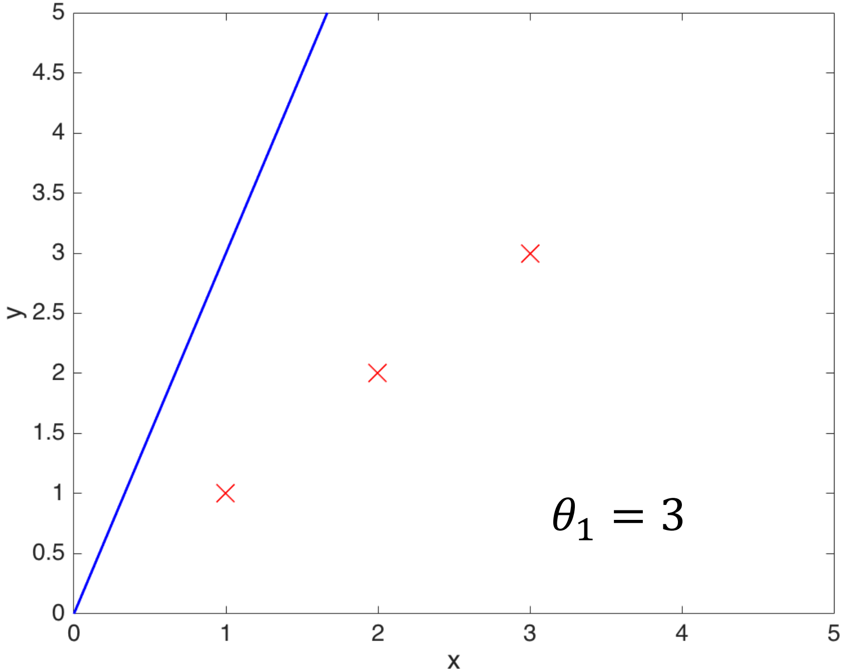


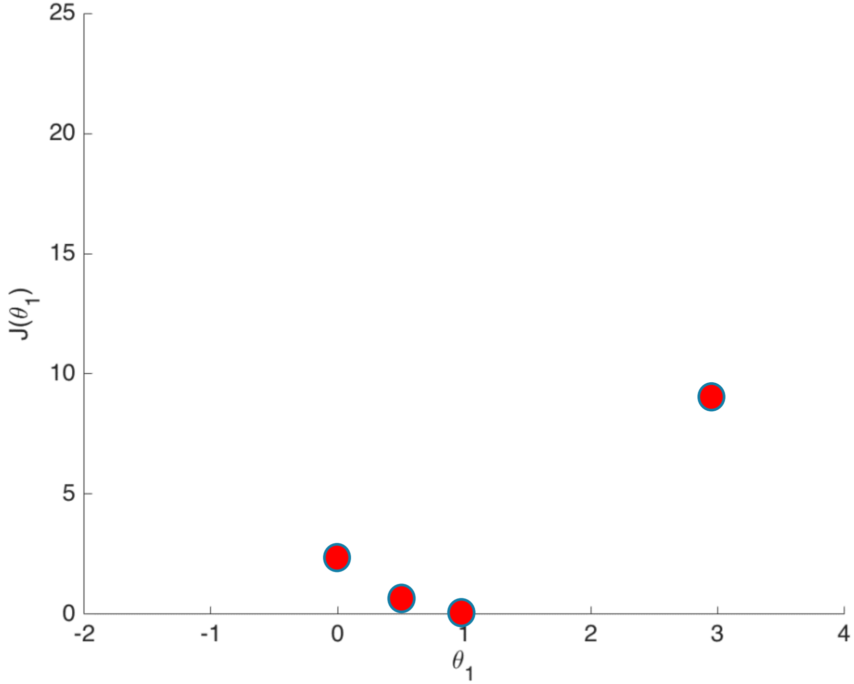
Cost function



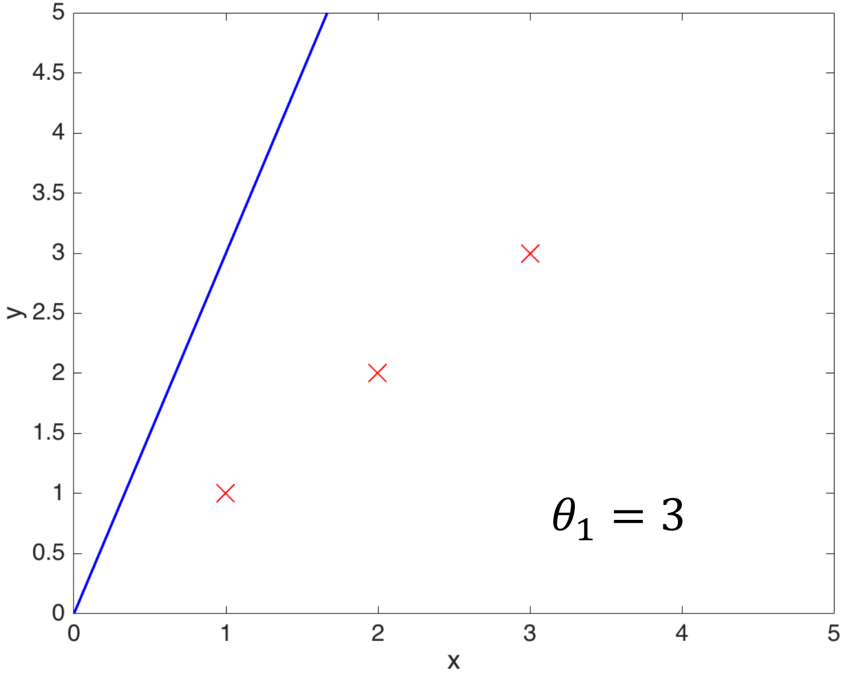


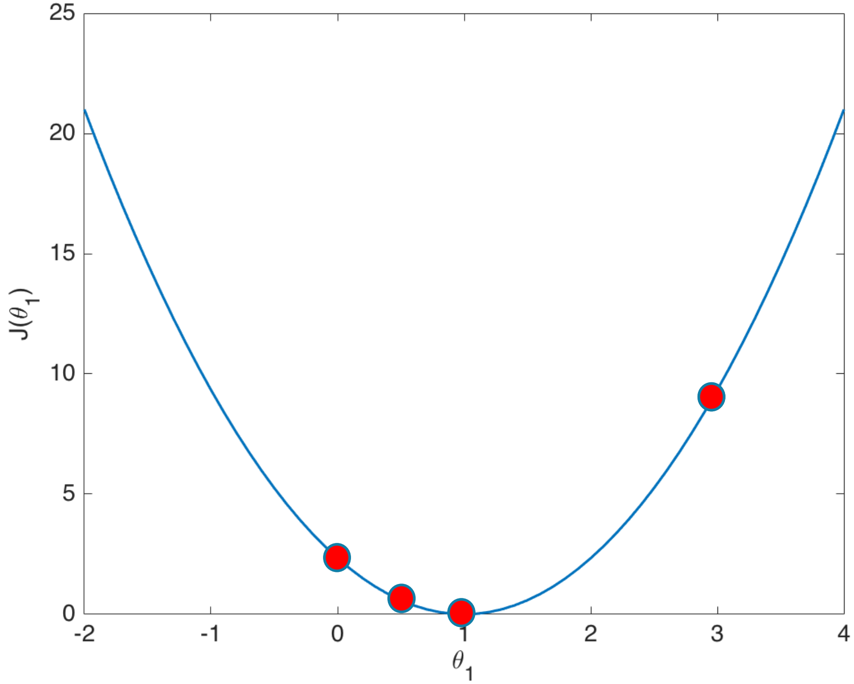
Cost function



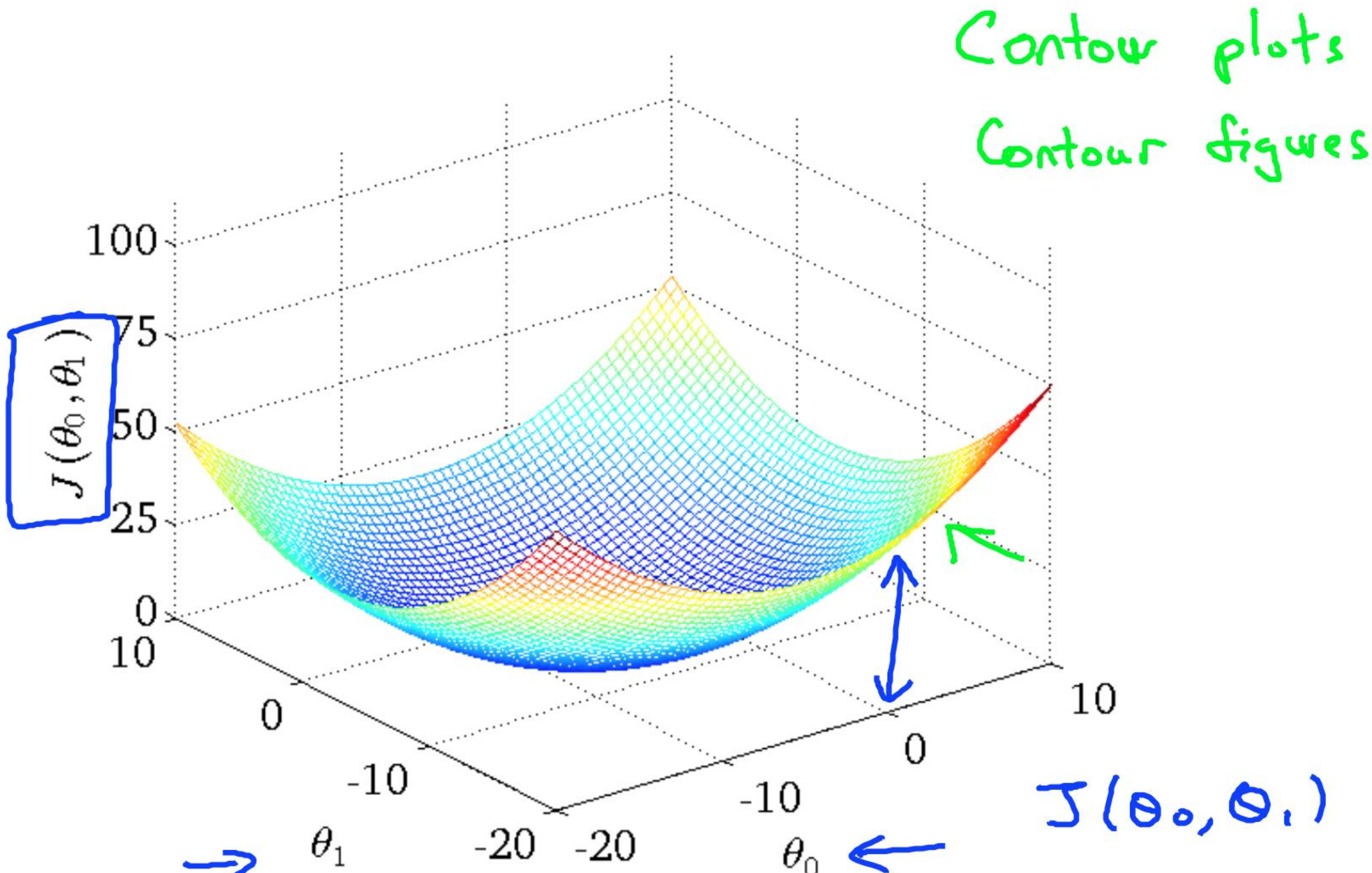


Cost function





Cost function



Source: Andrew Ng

Gradient descent algorithm

- Cost function
  - $J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$
- Goal
  - Identify θ_0, θ_1 so that $J(\theta_0, \theta_1)$ reaches minimal

Gradient descent algorithm

- Cost function
  - $J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$
- Goal
  - Identify θ_0, θ_1 so that $J(\theta_0, \theta_1)$ reaches minimal
- To find θ_0, θ_1
  - Starting from certain point θ_0, θ_1 (e.g., $\theta_0 = 0, \theta_1 = 0$)
  - Change values of θ_0, θ_1 until $J(\theta_0, \theta_1)$ reaches minimal
- How to change values of θ_0, θ_1?

Gradient descent algorithm

- Partial derivative
  - $\frac{dJ}{d\theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$
  - $\frac{dJ}{d\theta_1} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$

Gradient descent algorithm

- Partial derivative
  - $\frac{dJ}{d\theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$
  - $\frac{dJ}{d\theta_1} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$
- Gradient vector
  - $\left(\frac{dJ}{d\theta_0}, \frac{dJ}{d\theta_1}\right)$

Gradient descent algorithm

- Partial derivative
  - $\frac{dJ}{d\theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$
  - $\frac{dJ}{d\theta_1} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$
- Gradient vector
  - $\left(\frac{dJ}{d\theta_0}, \frac{dJ}{d\theta_1}\right)$
- Going backward gradient vector makes function decrease

Gradient descent algorithm

Loop until convergence

{

$$\theta_0 = \theta_0 - \alpha \frac{dJ}{d\theta_0}$$

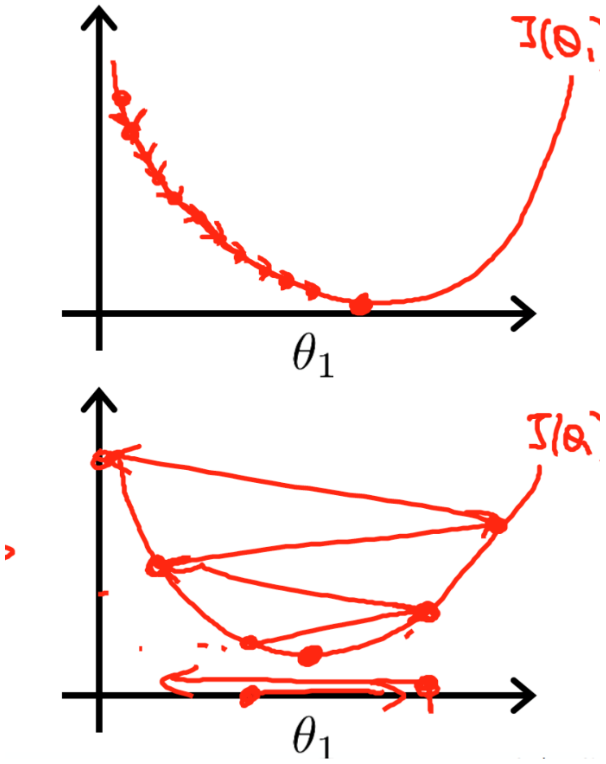
$$\theta_1 = \theta_1 - \alpha \frac{dJ}{d\theta_1}$$

}

- α is learning rate
- Steps
  - Step 1: calculate gradient vector
  - Step 2: update elements of vector θ
  - Step 3: recalculate cost

Learning rate

- If learning rate α is too small, algorithm converges slowly
- If learning rate α is too large, algorithm may not converge



Source: Andrew Ng

Multivariate linear regression

Part 2

Training set

x: Size (m2)	y: Price (millions VND)
20	600
50	876
80	1800
100	2000
…	…

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

Training set

x_1: Size (m^2)	x_2: Age (year)	x_3: Number of floor (m)	y: Price (millions VND)
20	5	1	600
20	8	3	876
80	10	3	1800
70	7	5	2000
…	…	…	…

- $(x^{(i)}, y^{(i)})$: i^{th} sample
- $x^{(i)}$: input of i^{th} sample
- $x^{(i)}_j$: j^{th} feature i^{th} sample
- n: number of features

Hypothesis

- Single variable

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

single value

- Multiple variables

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

vector

Hypothesis

- Multiple varibles

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

$x \in \mathbb{R}^n, \qquad \theta \in \mathbb{R}^{n+1}$

Given $x_0 = 1$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}, \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \cdots \\ \theta_n \end{bmatrix}$$

$$\rightarrow h_\theta\,(x) = \theta^T x, \qquad \theta, x \in \mathbb{R}^{n+1}$$

Cost function

- Hypothesis
  - $h_\theta(x) = \theta^T x$
- Parameters
  - $\theta = [\theta_0, \theta_1, \theta_2, \ldots \theta_n]^T$
- Cost function
  - $\mathrm{J}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)^2$

Gradient descent algorithm

- Gradient vector
  - $\frac{dJ}{d\theta_j} = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})\, x_j^{(i)}$
  - $j = 0, 1, 2, \dots, n$
- Repeat until convergence

$$\{$$

$$\theta_j = \theta_j - \alpha \frac{dJ}{d\theta_j}$$

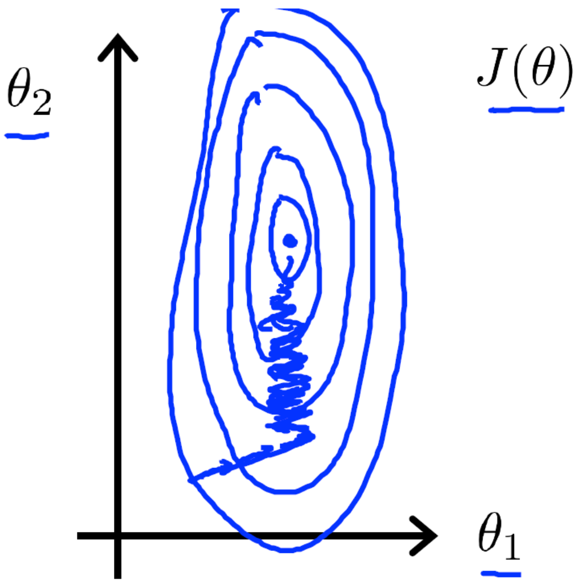
$$\}$$

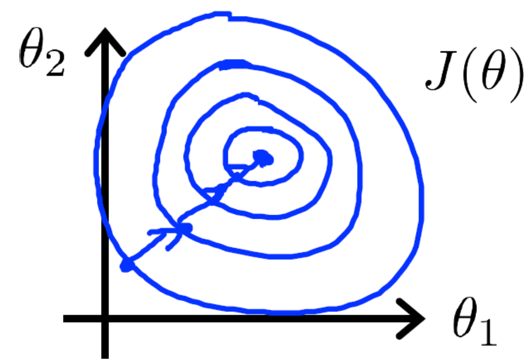
Feature normalization

x_1: Size (m^2)	x_2: Age (year)	x_3: Number of floor (m)	y: Price (millions VND)
20	5	1	600
20	8	3	876
80	10	3	1800
70	7	5	2000
…	…	…	…

Feature normalization

Scale values of features into the same range





x_1 = 20, 30, …, 100
x_2 = 1, 2, …, 10

$$x_1 = \frac{x_1}{100}$$

$$x_2 = \frac{x_2}{10}$$

Source: Andrew Ng

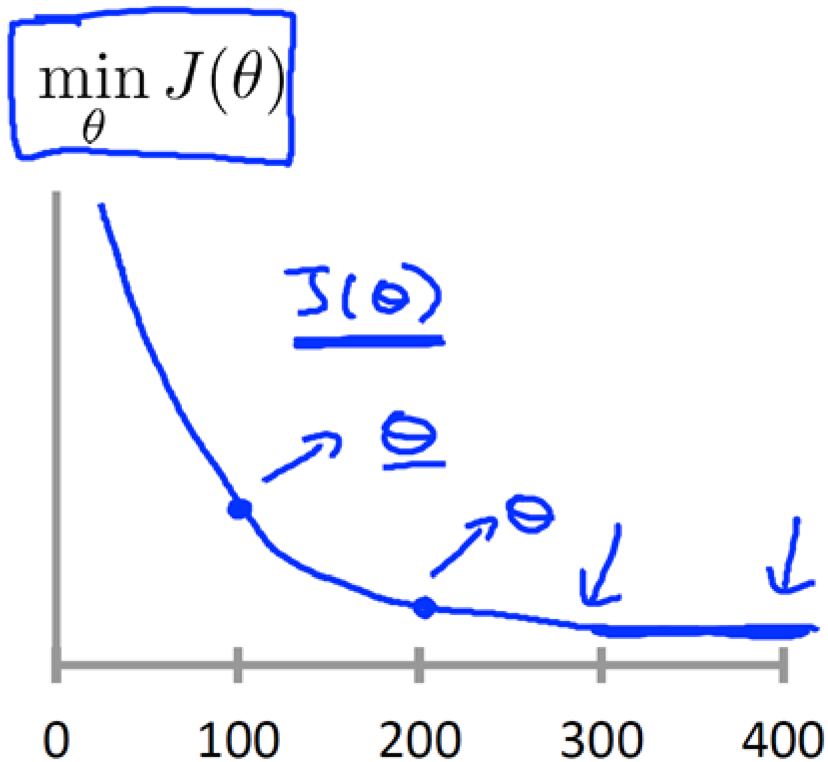
Feature normalization

- Mean-based normalization

$$x_j = \frac{x_j - \mu_j}{s_j}$$

  - μ_j: mean
  - s_j: standard deviation
- Range after normalization: $-1 \leq x_j \leq 1$

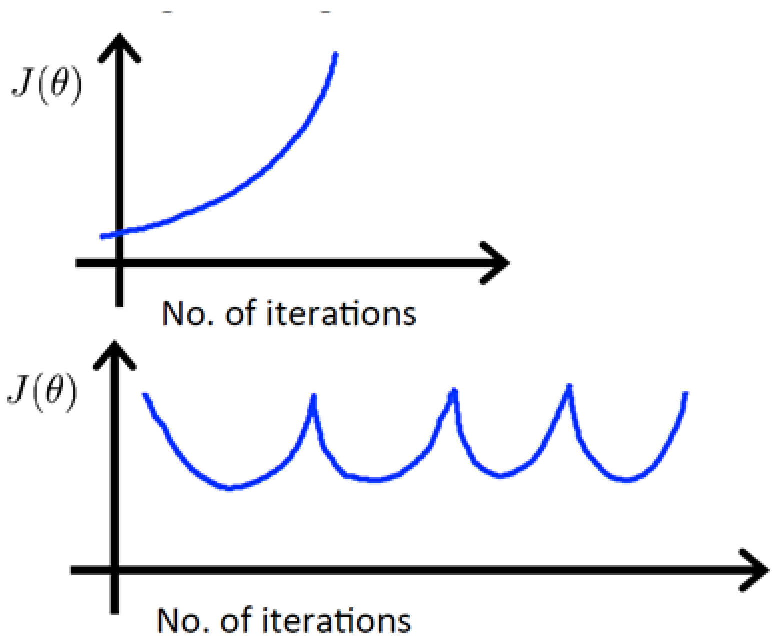
Learning rate

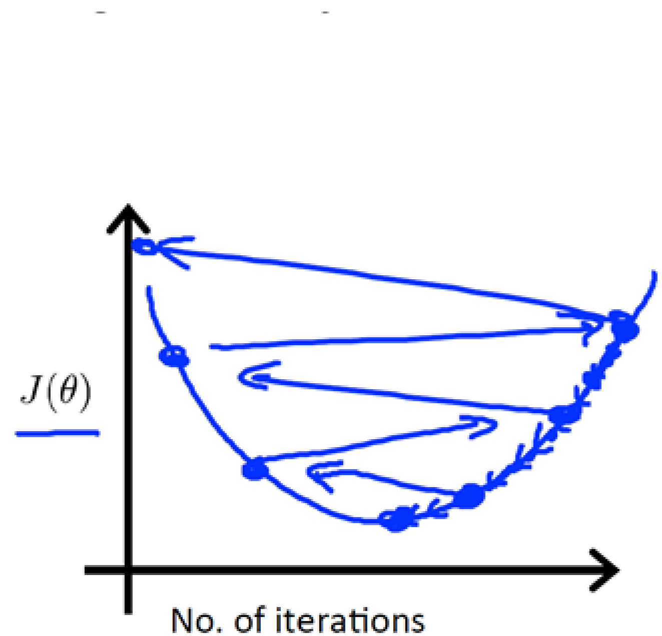


- Check if J decreases after each step of updating
- J converges when J decreases an amount smaller that 0.001 (ε) after a step

Source: Andrew Ng

Learning rate





- With large learning rate, J may not converge
- With small learning rate, J converges slowly
- Try with: …, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, …

Source: Andrew Ng