Linear Regression

Eun Yi Kim

Artificial Inteligence
& Computer Vision
Laboratory

Agenda

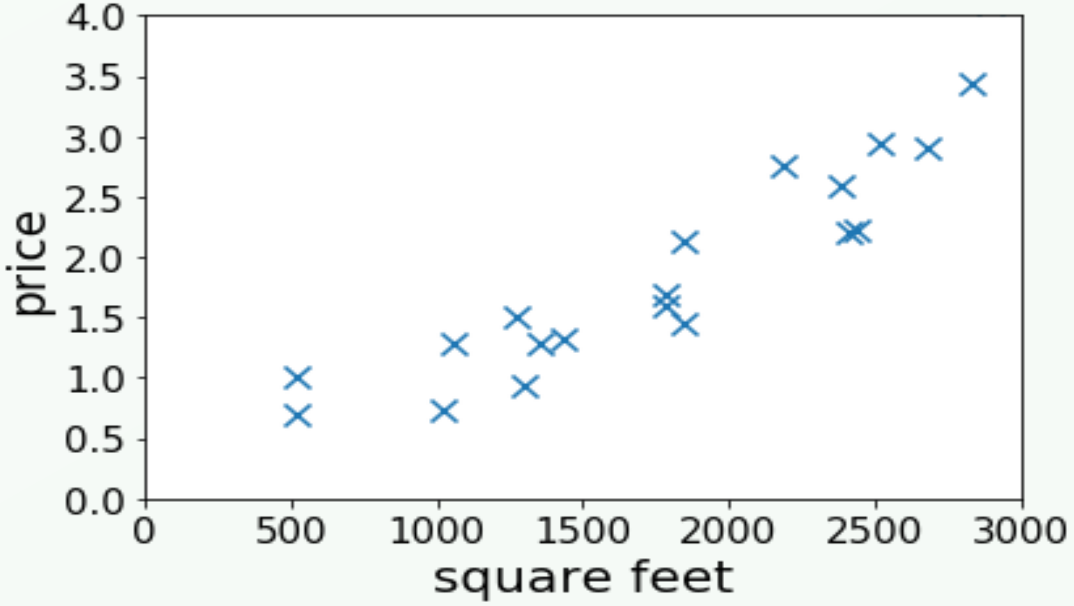
- Linear regression with one variable
  - ✓Model representation
  - ✓Cost function
  - ✓Gradient decent
  - ✓Gradient decent for linear regression
- Linear regression with multiple variable
  - ✓ Multiple features
  - ✓ Gradient decent for multiple variables
  - ✓ Gradient decent in practice
  - ✓ Features and Polynomial regression

Linear regression with one variable

Model Representation

Housing Prices

(in 100,000s of dollars)



- Supervised Learning

  : Given the “right answers” for each example in the data

- Regression

  : Predict real-valued output (price)

- Classification

  : Discrete-valued output

Model Representation

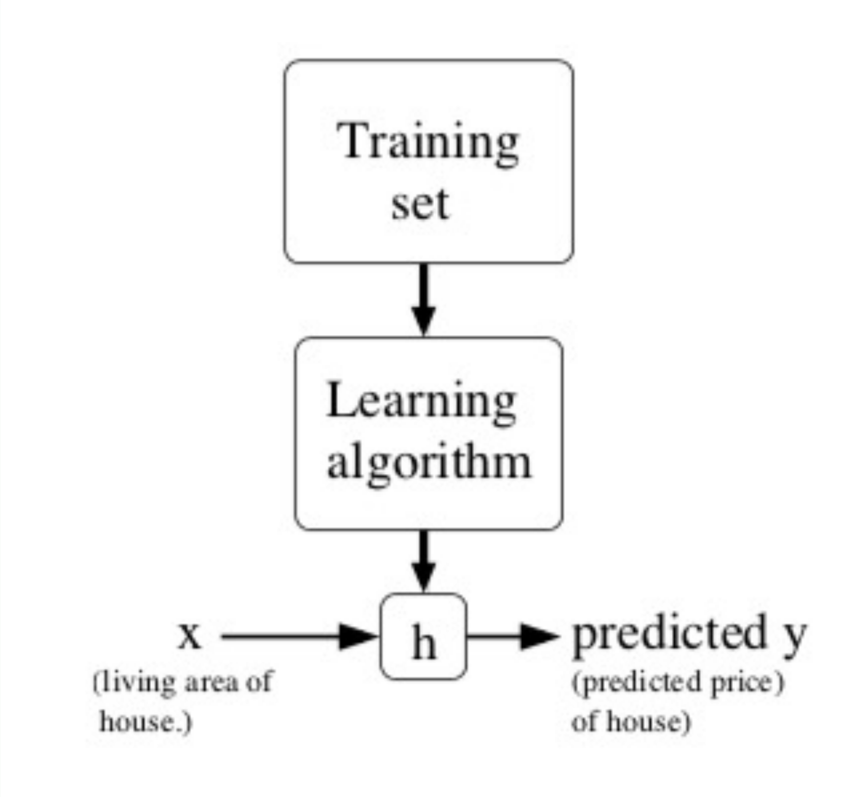
Training set of housing prices

Living area ($feet^2$)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

Notation:

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

Model Representation



How do we represent *h*?

Linear regression with one variable

: Univariate linear regression

Cost Function

Training set of housing prices

Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

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

θ_i's : Parameters

How to choose θ_i's **?**

Cost Function

Artificial Inteligence & Computer Vision Laboratory

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

$\theta_0 = 1.5$

$\theta_1 = 0$

$\theta_0 = 0$

$\theta_1 = 0.5$

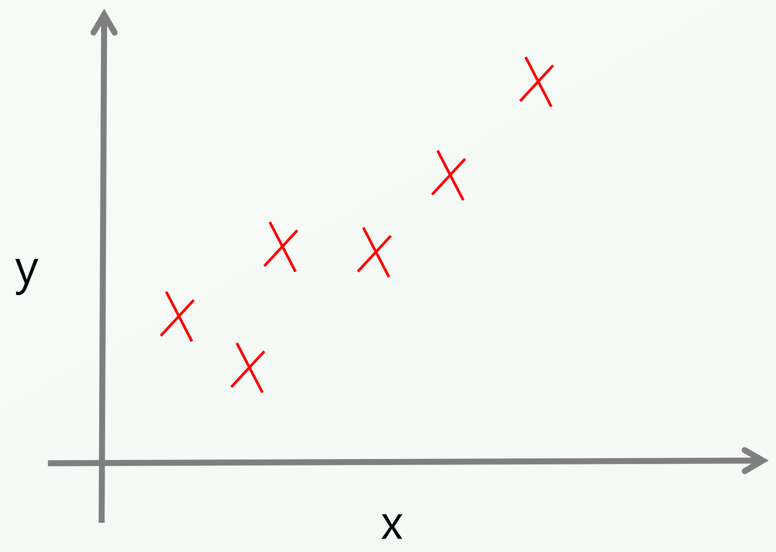
$\theta_0 = 1$

$\theta_1 = 0.5$

Cost Function

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

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



Idea: Choose θ_0 and θ_1 so that

$h_\theta(x)$ is close to y

for our training example (x, y)

Cost Function (intuition)

Simplified

Hypothesis:

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

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

Parameters:

$$\theta_0, \theta_1$$

Cost function:

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

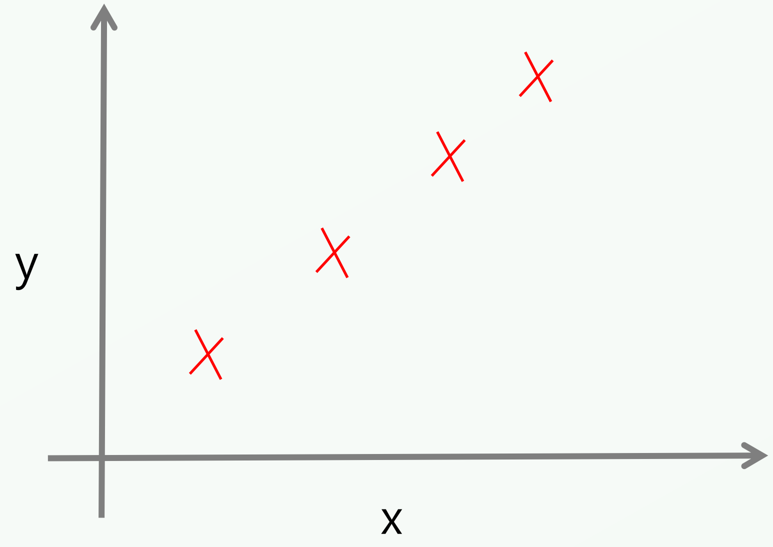
Goal:

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

Cost Function (intuition)

$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$h_\theta(x)$

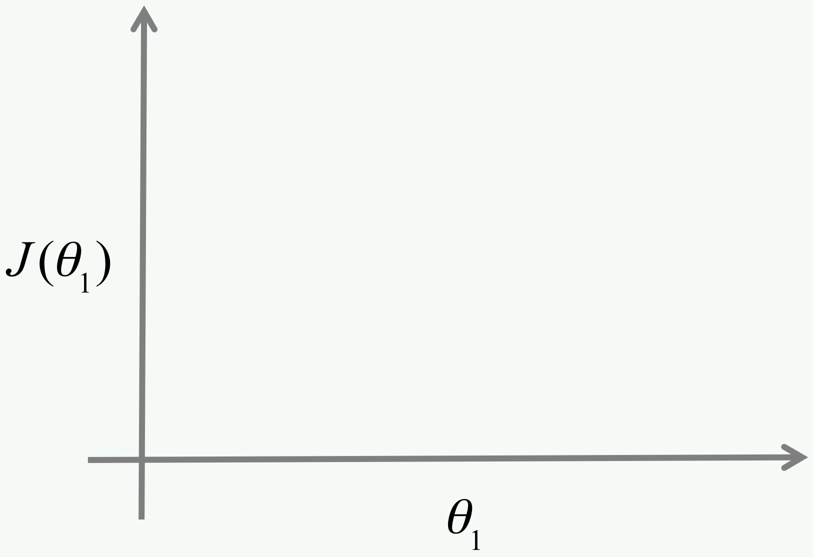
Idea: Choose θ_0 and θ_1 so that

$h_\theta(x)$ is close to y

for our training example (x, y)

$J(\theta_1)$

(function of the parameter θ_1)



Cost Function (intuition)

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

Parameters: θ_0, θ_1

Cost function:

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

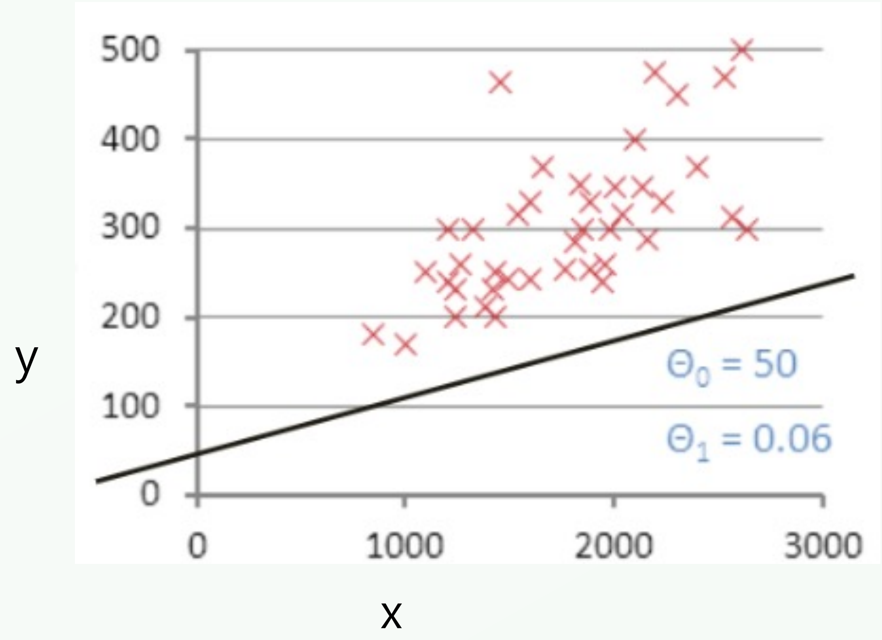
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\, J(\theta_0, \theta_1)$

Cost Function (intuition)

Artificial Inteligence
& Computer Vision
Laboratory

$$h_\theta(x)$$

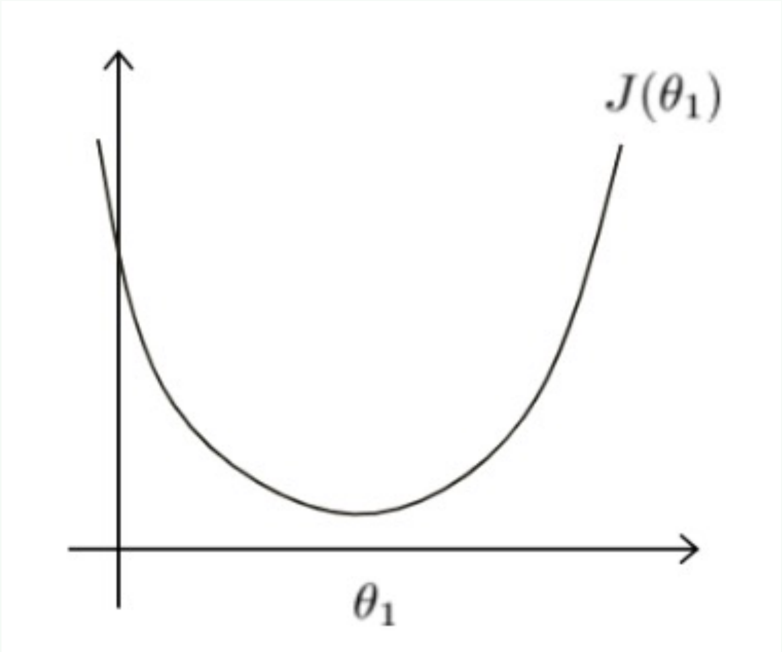
(for fixed θ_1, this is a function of x)



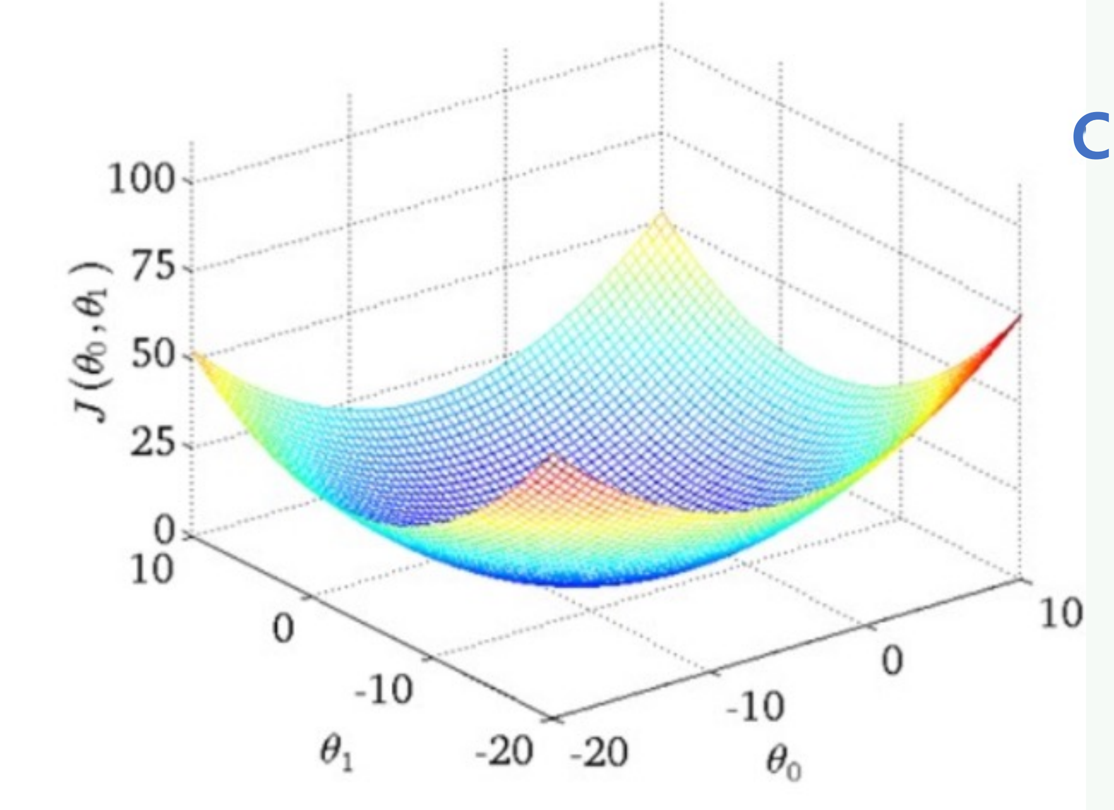
$$h_\theta(x) = 50 + 0.06x$$

$$J(\theta_1)$$

(function of the parameter θ_1)



Cost Function (intuition)



Contour plots

Cost Function (intuition)

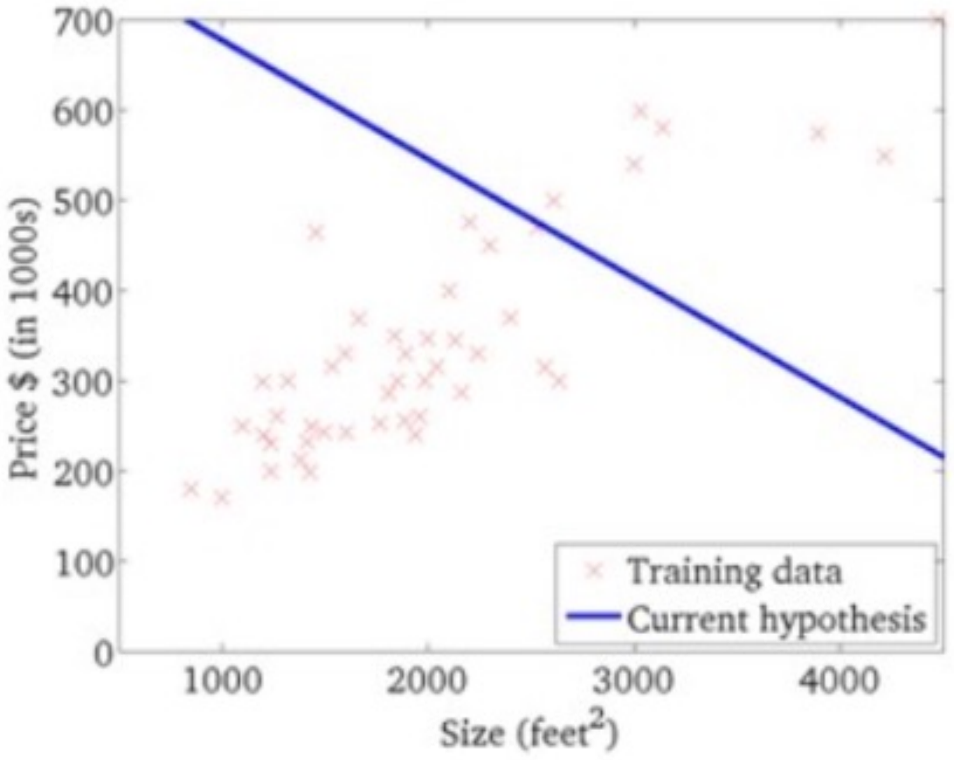
Artificial Inteligence & Computer Vision Laboratory

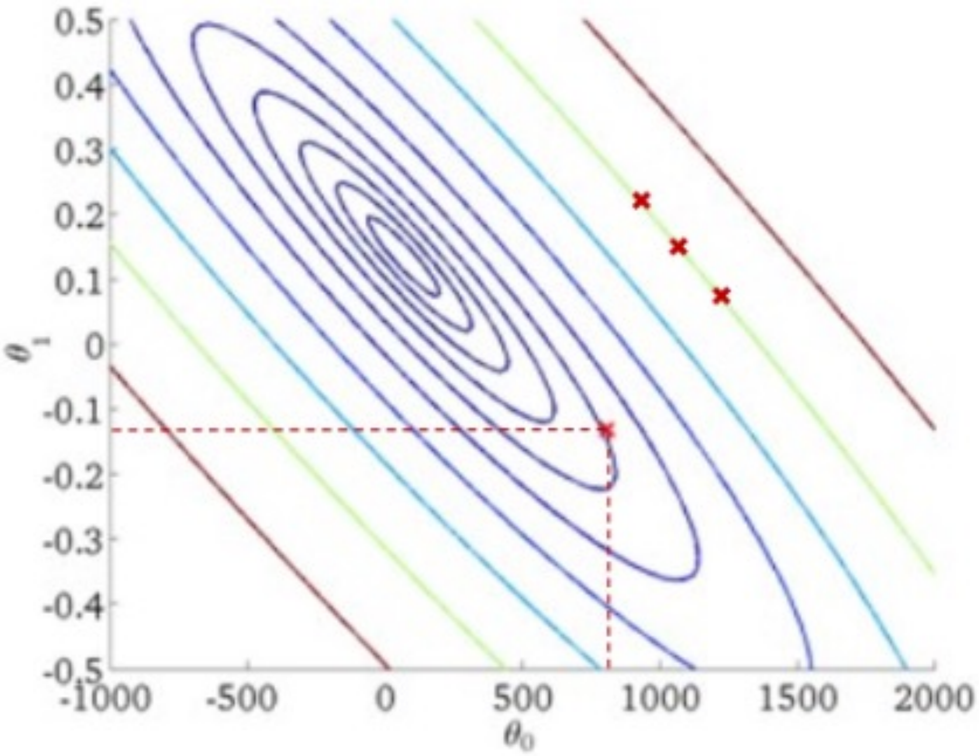
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)

$J(\theta_1)$

(function of the parameter θ_0, θ_1)



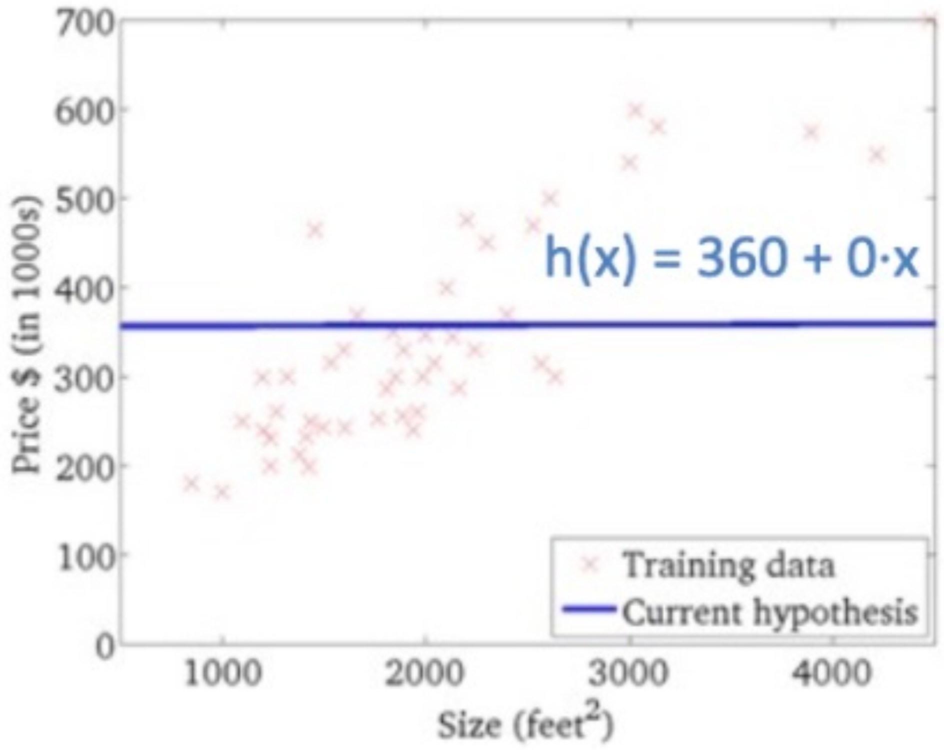


Cost Function (intuition)

Artificial Inteligence & Computer Vision Laboratory

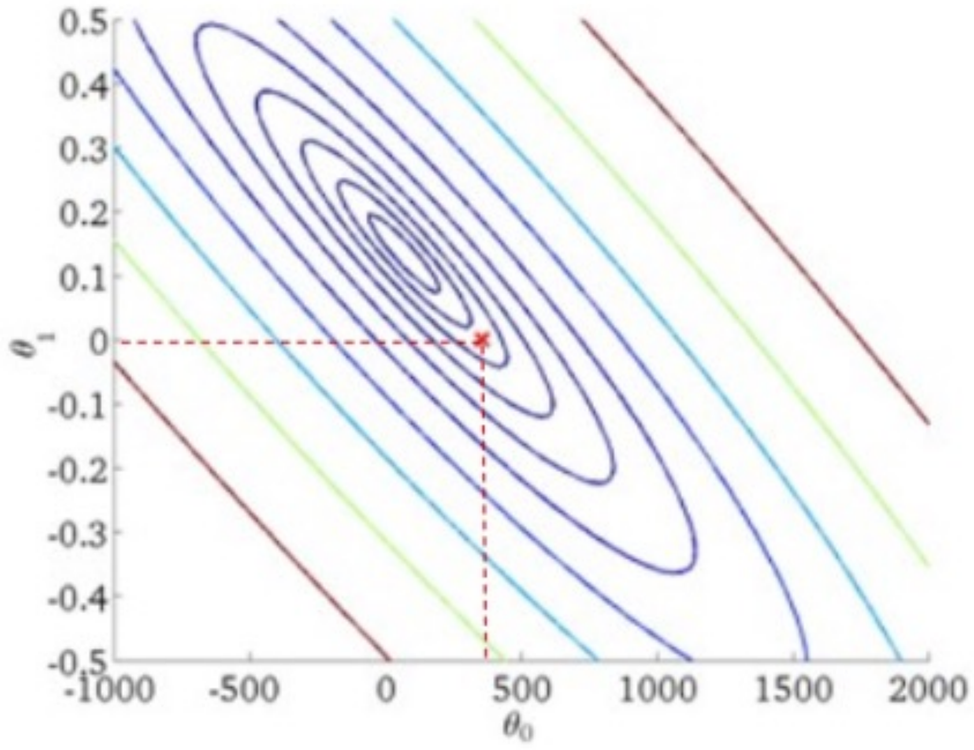
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_1)$

(function of the parameter θ_0, θ_1)



$\Theta_0 = 360$

$\Theta_1 = 0$

Cost Function (intuition)

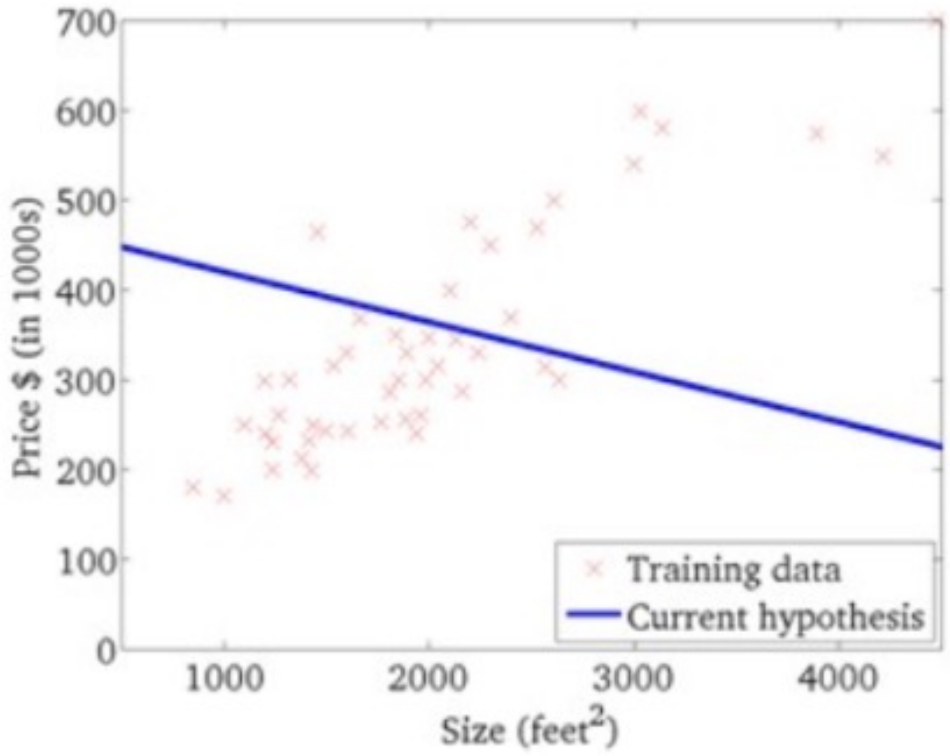
Artificial Inteligence & Computer Vision Laboratory

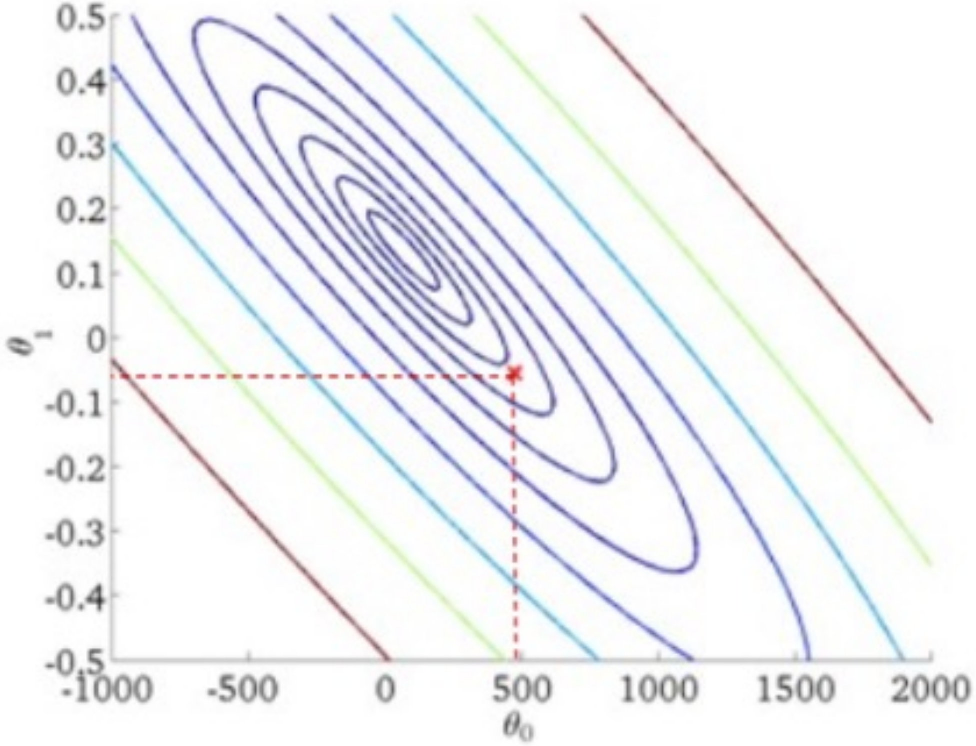
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)

$J(\theta_1)$

(function of the parameter θ_0, θ_1)





Cost Function (intuition)

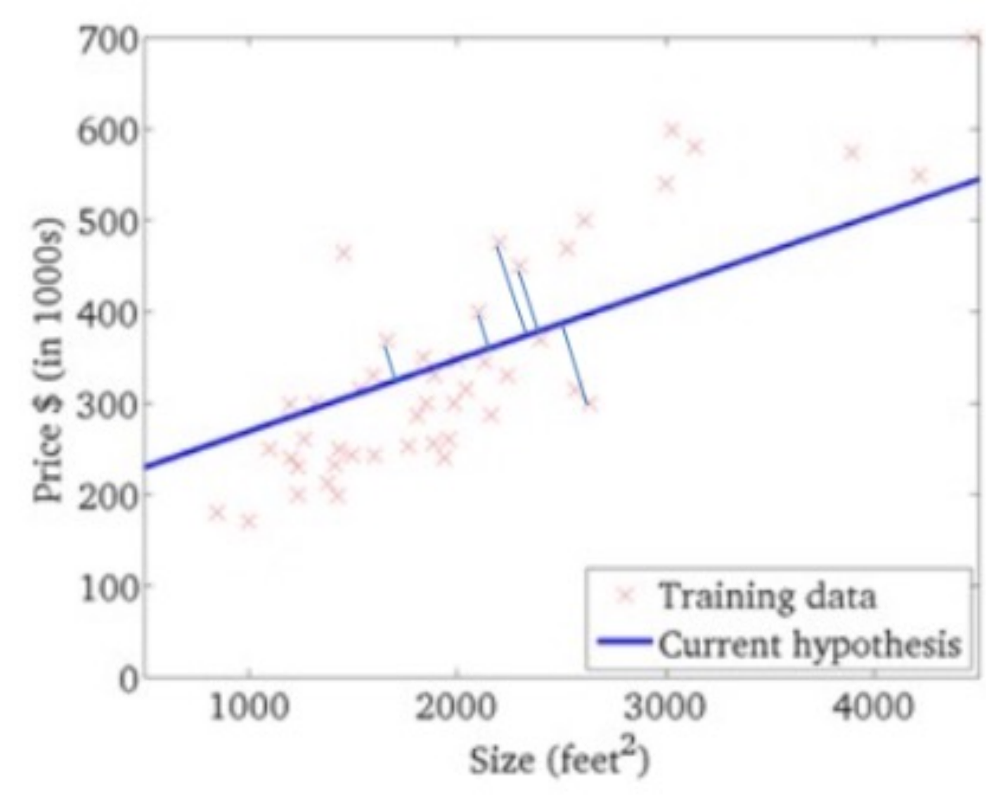
Artificial Inteligence & Computer Vision Laboratory

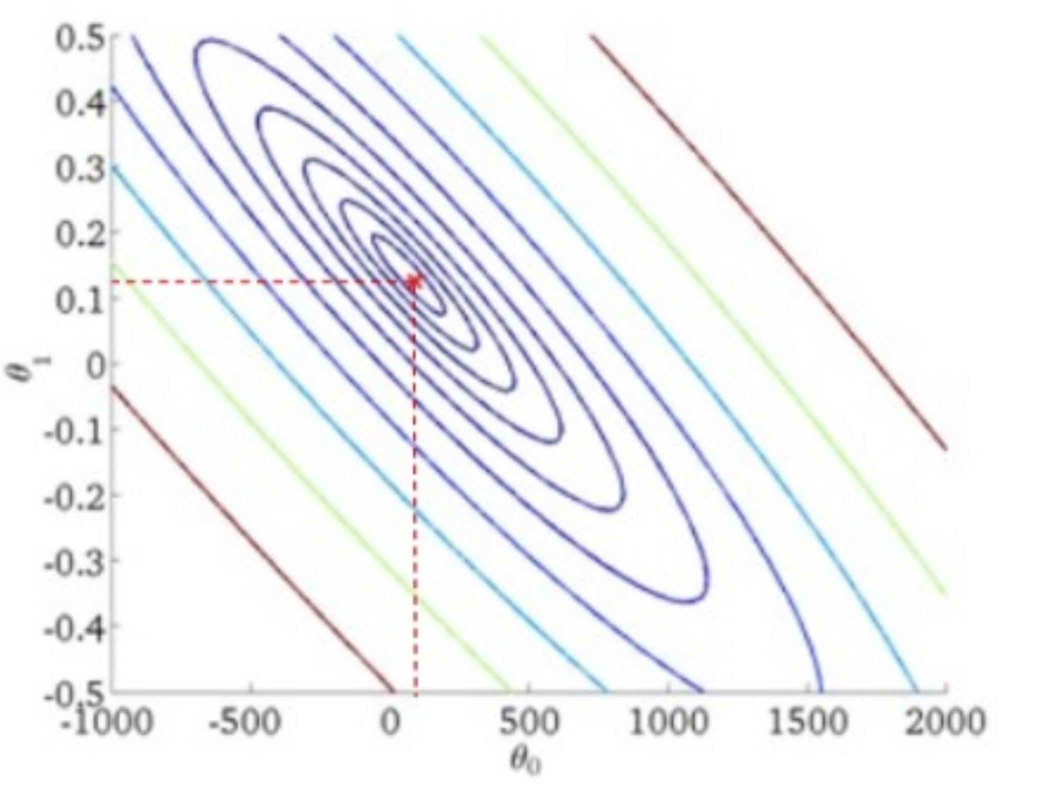
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)

$J(\theta_1)$

(function of the parameter θ_0, θ_1)





Gradient Descent

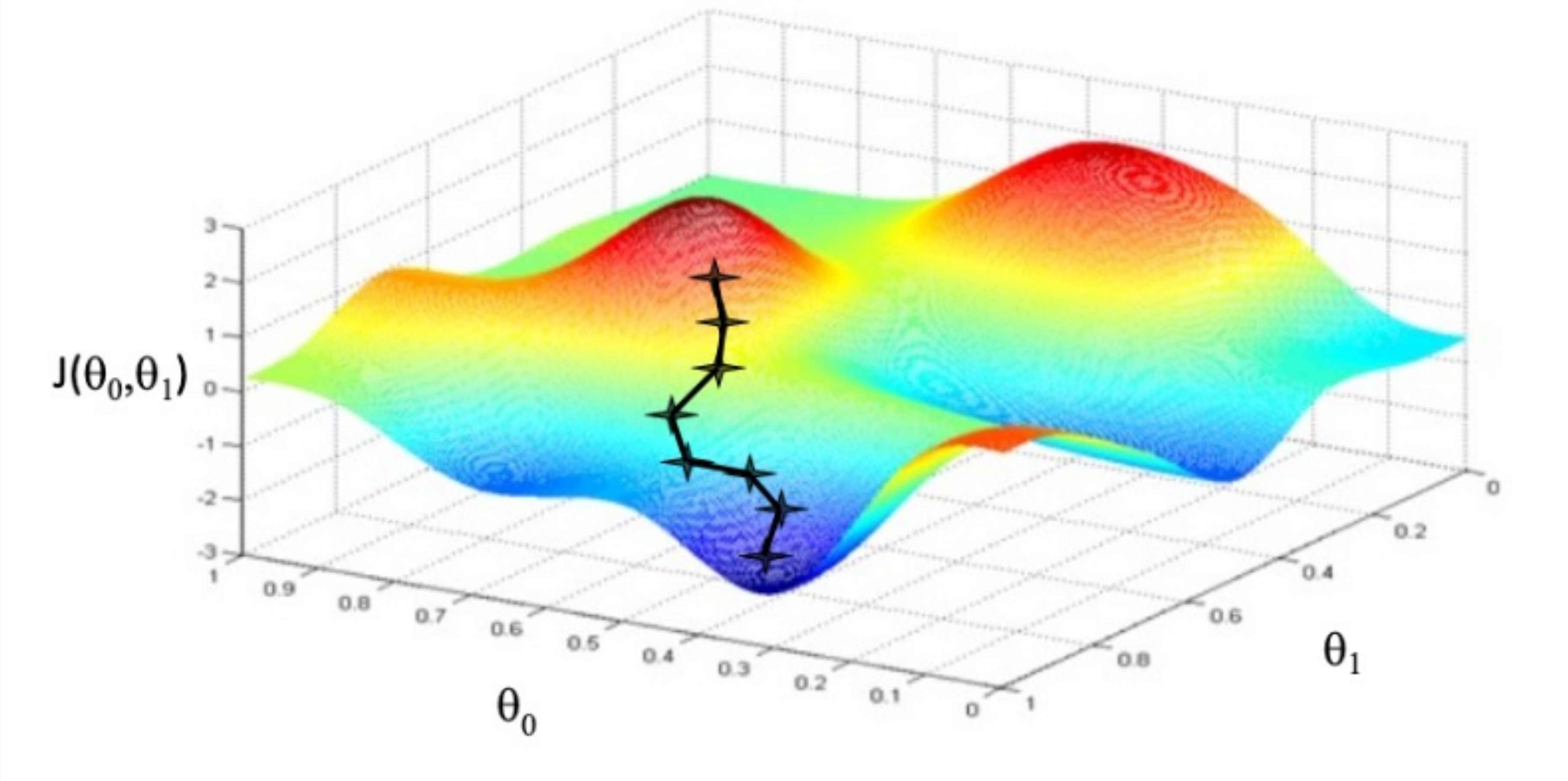
Have some function $J(\theta_0, \theta_1)$

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

Outline:

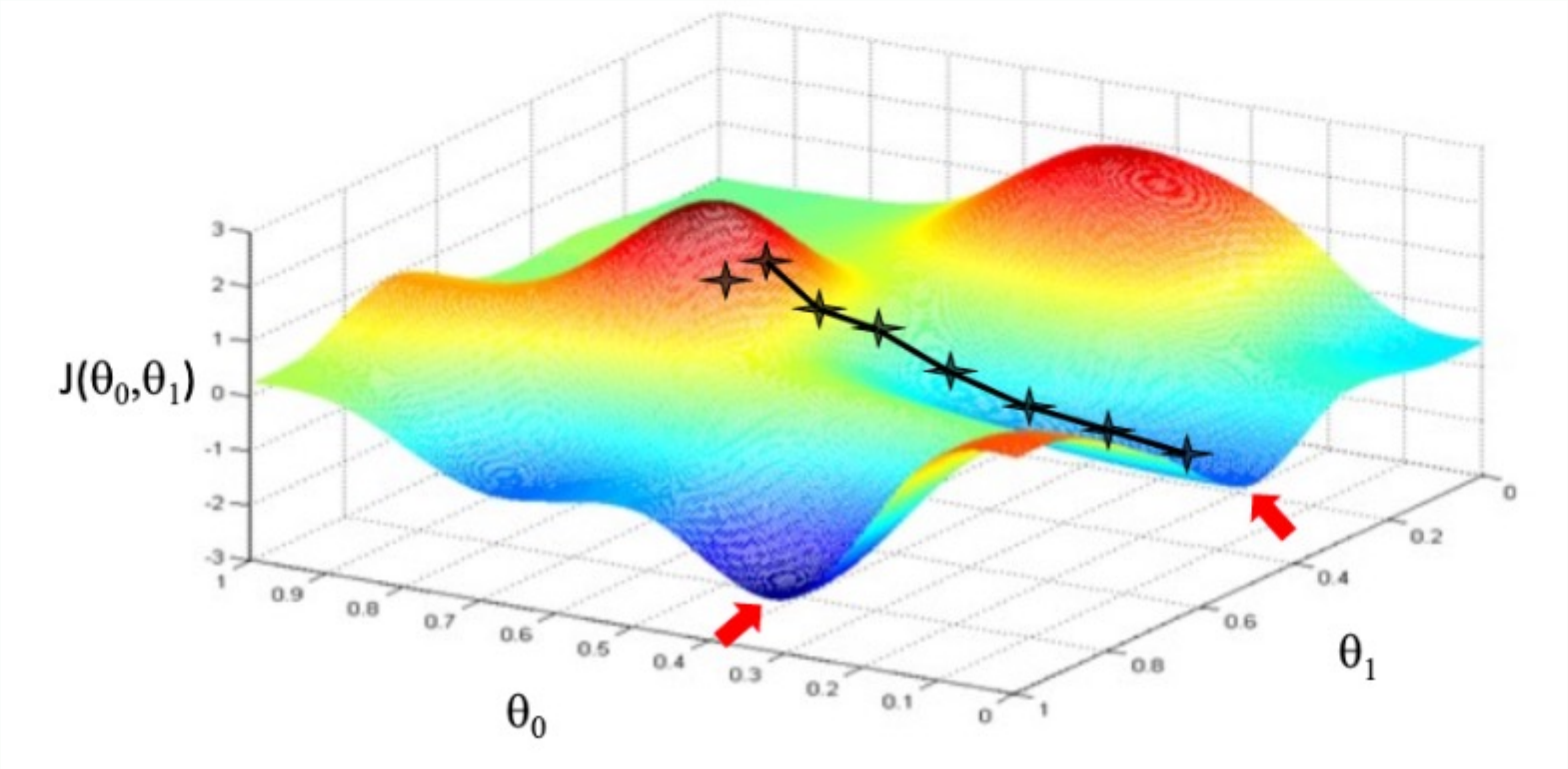
- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Gradient Descent



Gradient Descent

Artificial Inteligence
& Computer Vision
Laboratory



Gradient Descent

Gradient descent algorithm

assignment
a:=b

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

Gradient descent algorithm

repeat until convergence {

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

}

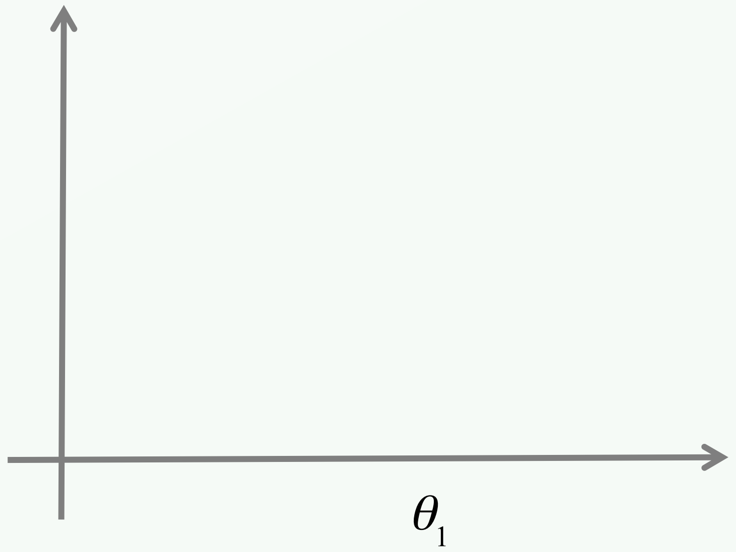
Learning rate

derivative

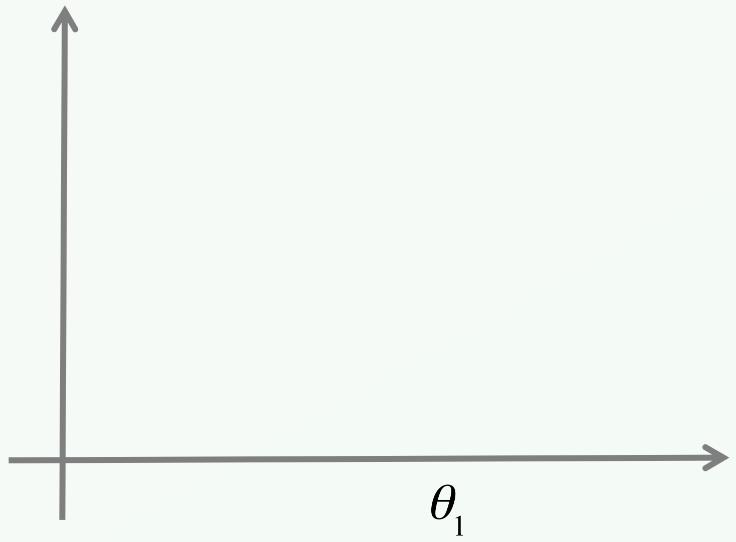
Gradient Descent

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

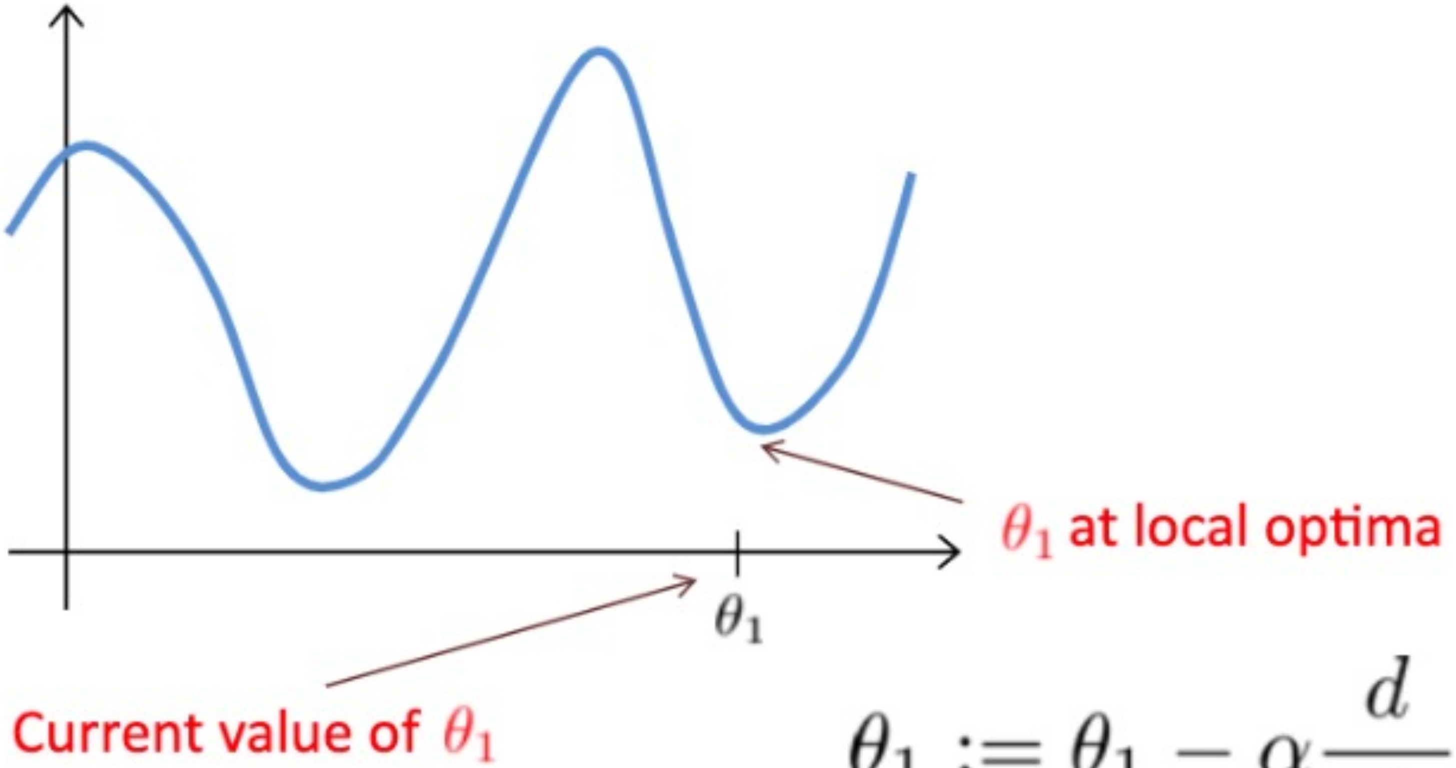
- If α is too small, gradient descent can be slow



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



Gradient Descent



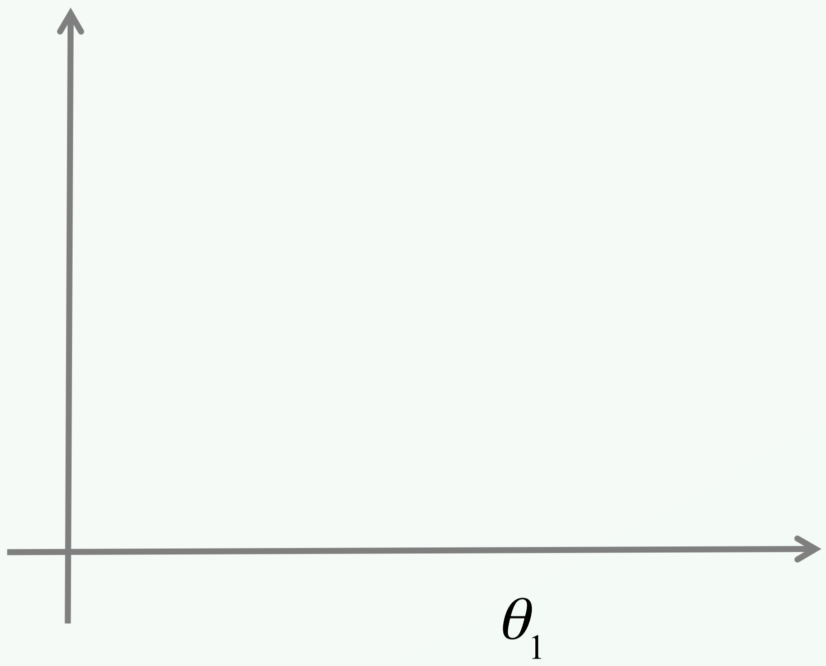
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Gradient Descent

- 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 for linear regression

Artificial Inteligence
& Computer Vision
Laboratory

Gradient descent algorithm

repeat until convergence {

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

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

}

Linear Regression Model

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

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

Gradient Descent for linear regression

Gradient descent algorithm

repeat until convergence {

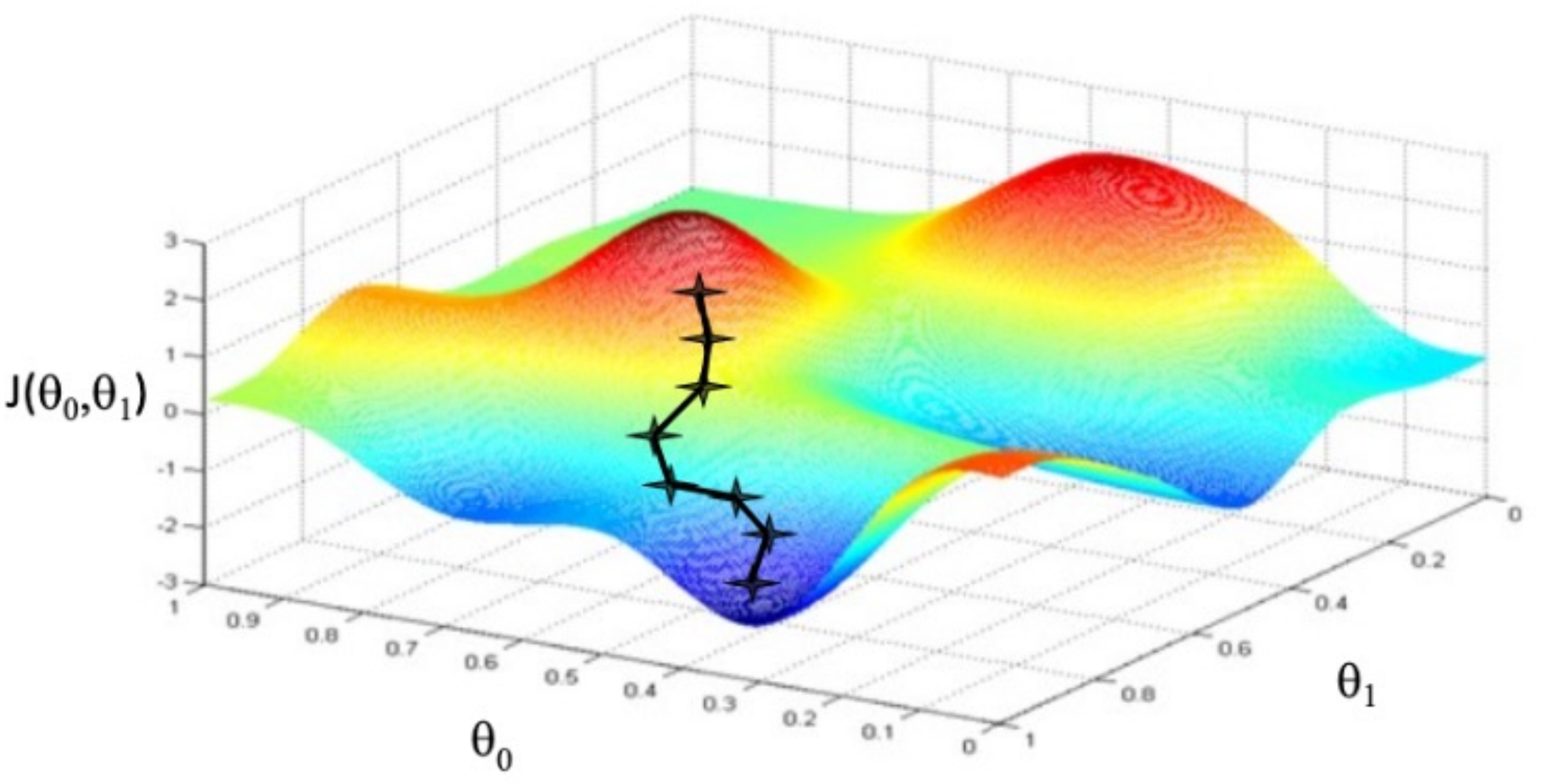
$$\theta_0 := \theta_0 - \alpha \overbrace{\boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)}}^{\frac{d}{d\theta_0}\cdot J(\theta_0,\theta_1)}$$

$$\theta_1 := \theta_1 - \alpha \underbrace{\boxed{\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\cdot x^{(i)}}}_{\frac{d}{d\theta_1}\cdot J(\theta_0,\theta_1)}$$

}

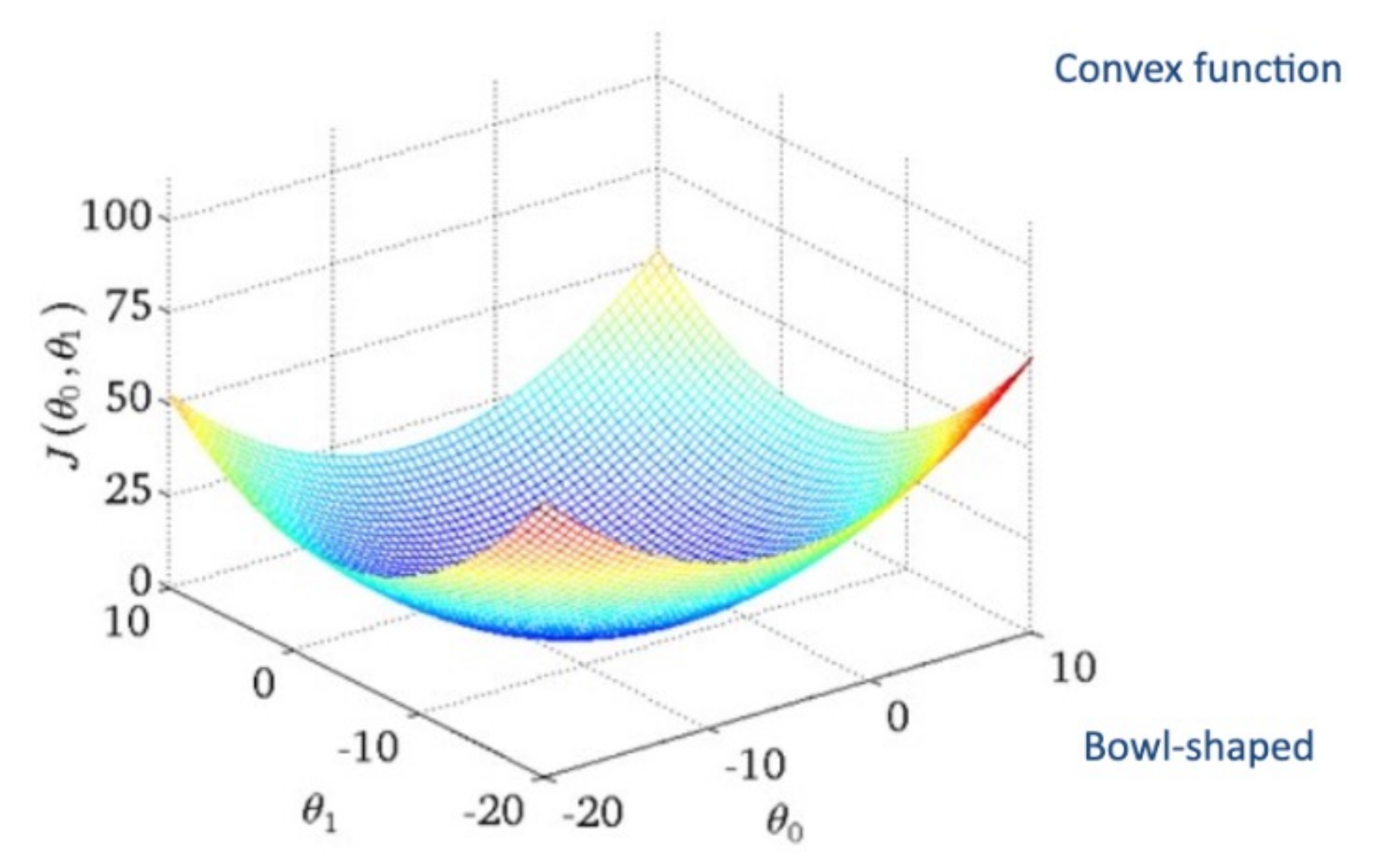
update θ_0 and θ_1 simultaneously

Gradient Descent for linear regression



Gradient Descent for linear regression

Artificial Inteligence & Computer Vision Laboratory



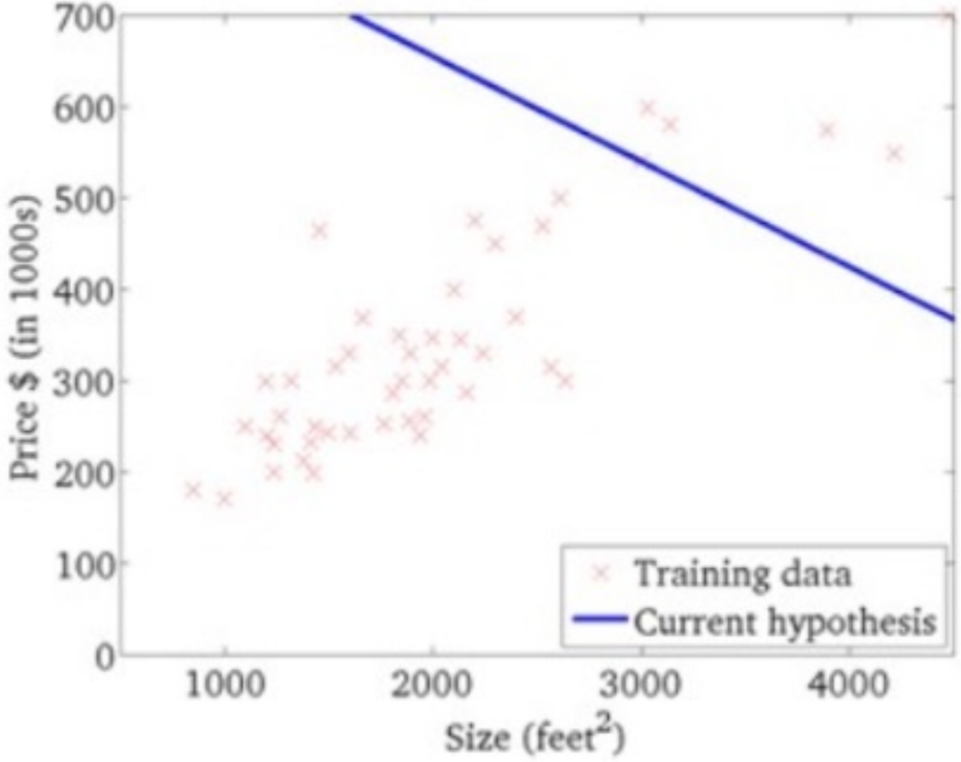
Gradient Descent for linear regression

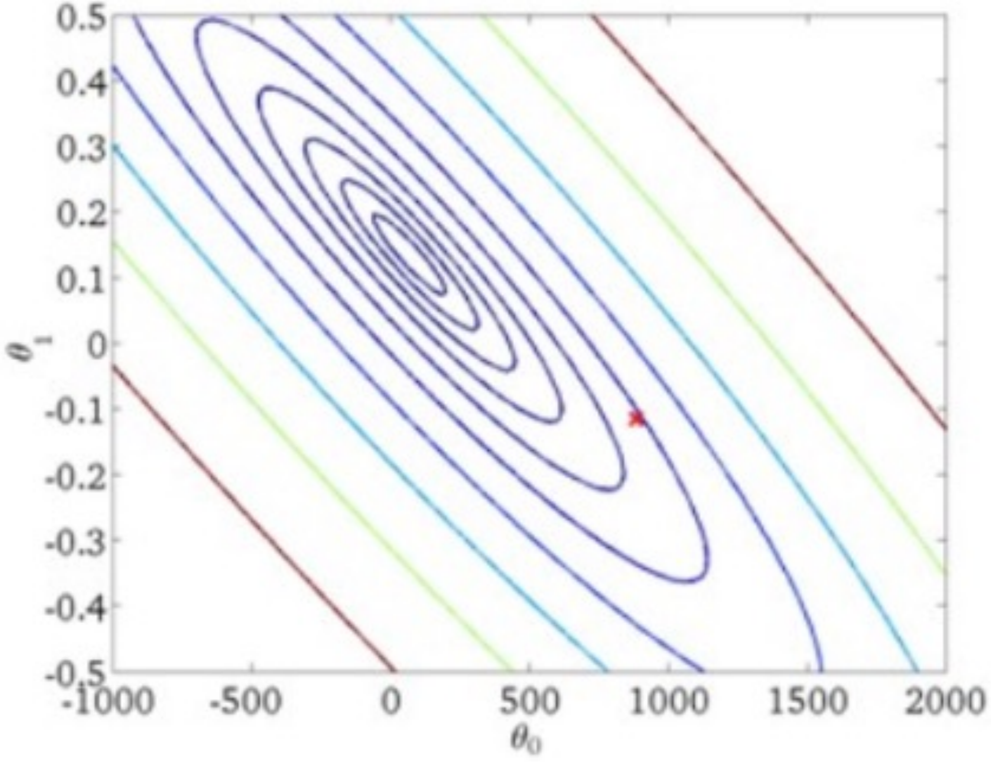
Artificial Inteligence
& Computer Vision
Laboratory

$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)

(function of the parameter θ_0, θ_1)





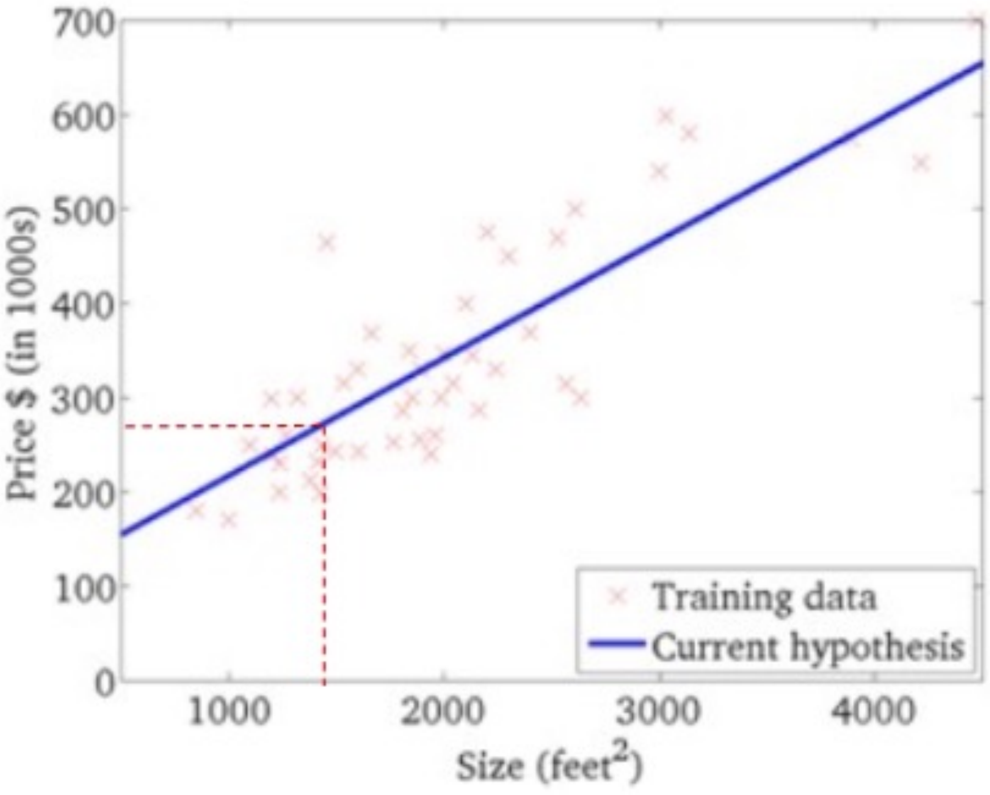
Gradient Descent for linear regression

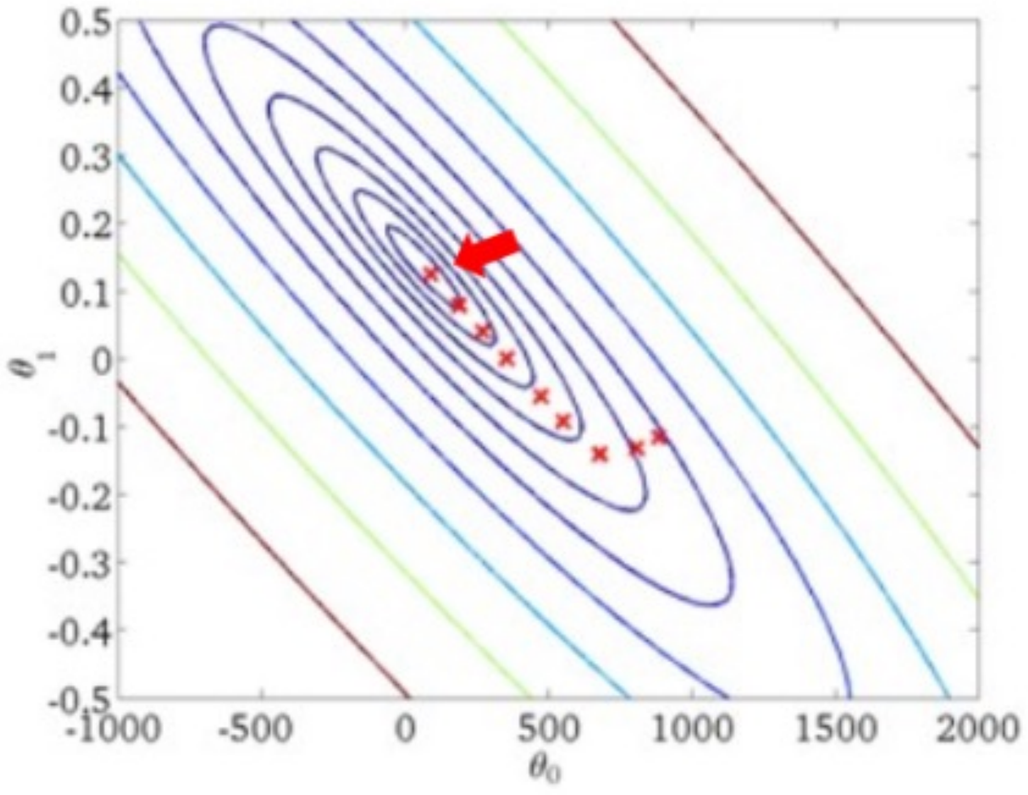
Artificial Inteligence & Computer Vision Laboratory

$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)

(function of the parameter θ_0, θ_1)





“Batch” Gradient Descent

“Batch”: Each step of gradient descent uses all the training examples

Linear regression with multiple variable

Multiple Features (variables)

Artificial Inteligence & Computer Vision Laboratory

Training set of housing prices

Size ($feet^2$)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Notation:

- **n** = number of features
- $\mathbf{x}^{(i)}$ = input (features) of i^{th} training example
- $\mathbf{x}_j^{(i)}$ = value of feature j in i^{th} training example

Hypothesis

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

Hypothesis

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

For convenience of notation, define $x_0 = 1$

Gradient Descent for multiple variables

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:

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

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 for multiple variables

Artificial Inteligence & Computer Vision Laboratory

Gradient Descent

Previously (n=1):

Repeat $\{$

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

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

(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 $j = 0, \ldots, n$)

$\}$

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

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

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

$\ldots$

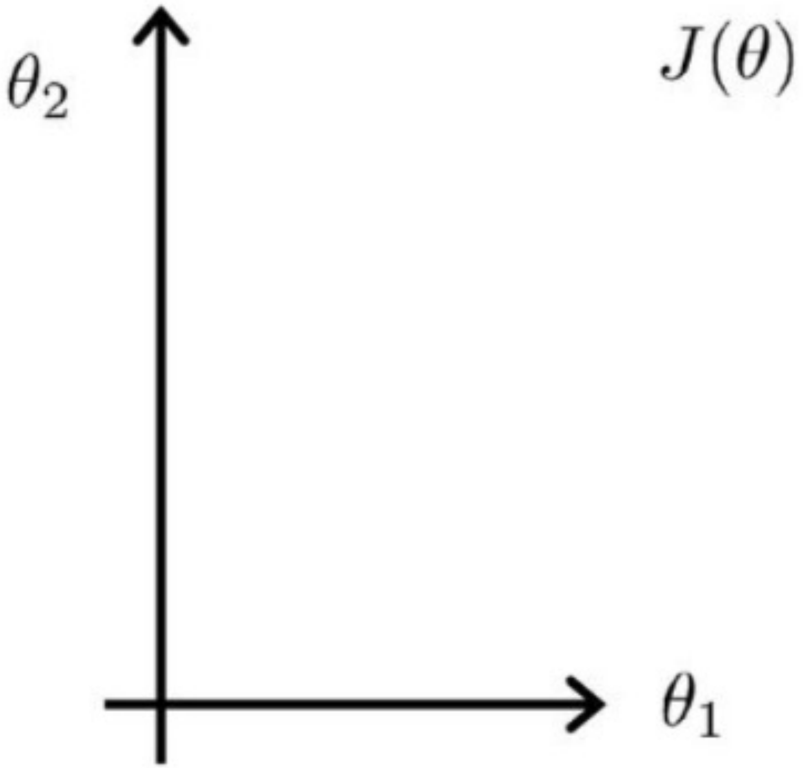
Gradient Descent in practice : Feature scaling

Artificial Inteligence & Computer Vision Laboratory

Idea: Make sure features are on a similar scale

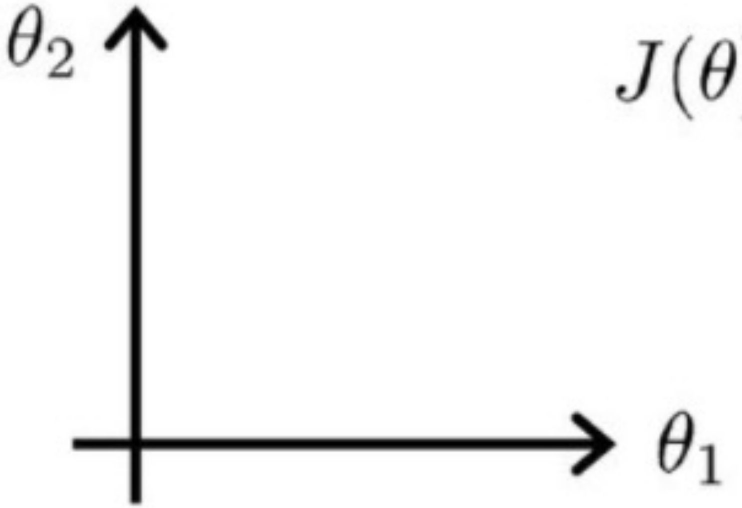
E.g. x_1 = size (0-2000 feet²)

x_2 = number of bedrooms (1-5)



$$x_1 = \frac{\text{size (feet}^2)}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$



Gradient Descent in practice : Feature scaling

Get every feature into approximately a $-1 \leq x_i \leq 1$ range

Gradient Descent in practice : Feature scaling

Artificial Inteligence & Computer Vision Laboratory

Mean normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean

(Do not apply to $x_0 = 1$)

E.g.

$$x_1 = \frac{size - 1000}{2000}$$

$$x_2 = \frac{\#bedrooms - 2}{5}$$

$$-0.5 \le x_1 \le 0.5, -0.5 \le x_2 \le 0.5$$

Gradient Descent in practice : Learning rate

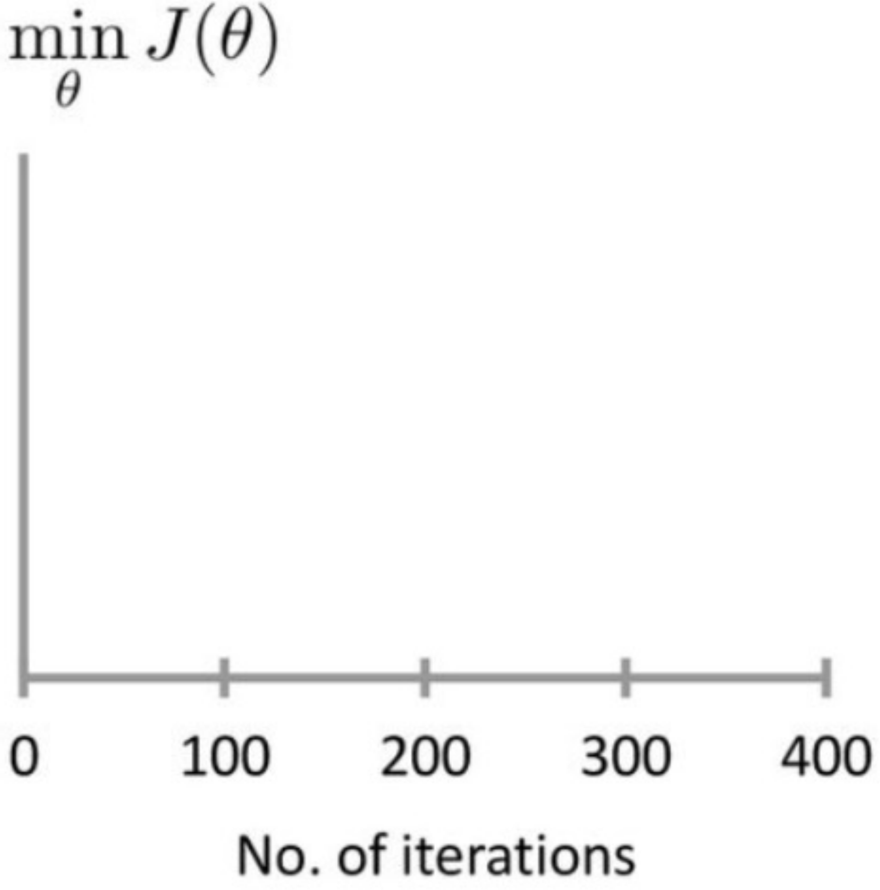
Gradient decent

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

- “Debugging”: How to make sure gradient descent is working correctly
- How to choose learning rate α

Gradient Descent in practice : Learning rate

Make sure gradient descent is working correctly

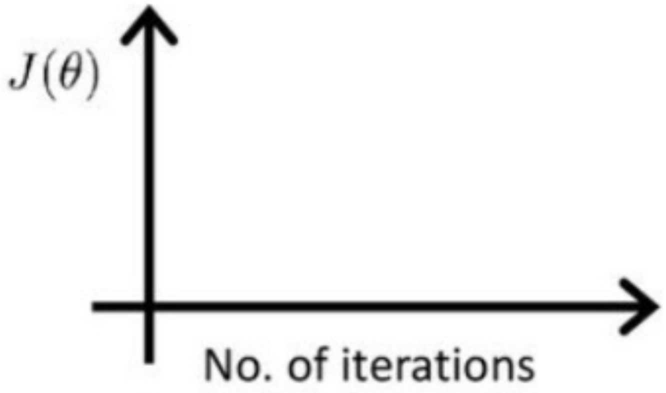


Example automatic convergence test:

Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

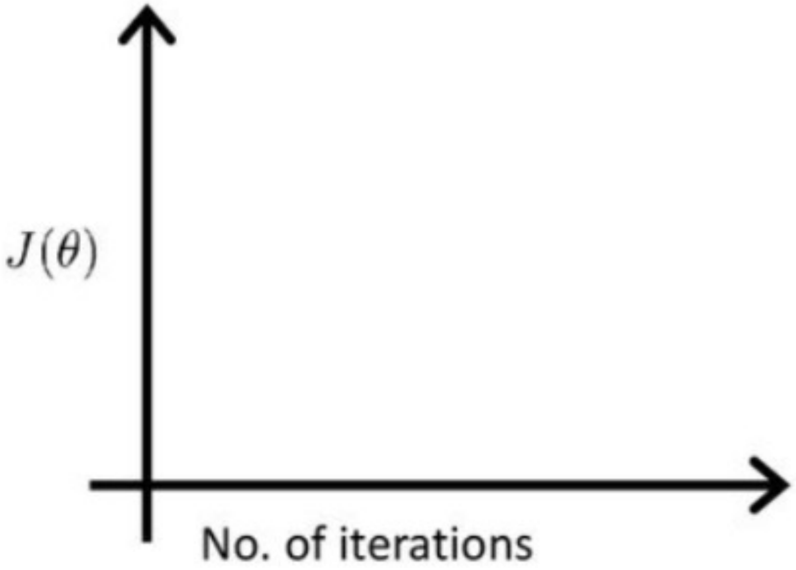
Gradient Descent in practice : Learning rate

Make sure gradient descent is working correctly



Gradient descent not working.

Use smaller α.



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Gradient Descent in practice : Learning rate

Artificial Inteligence & Computer Vision Laboratory

Summary

- If α is too small: slow convergence
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge

To choose learning rate α, try

…, 0.001, , 0.01, 0.1, 1, …….

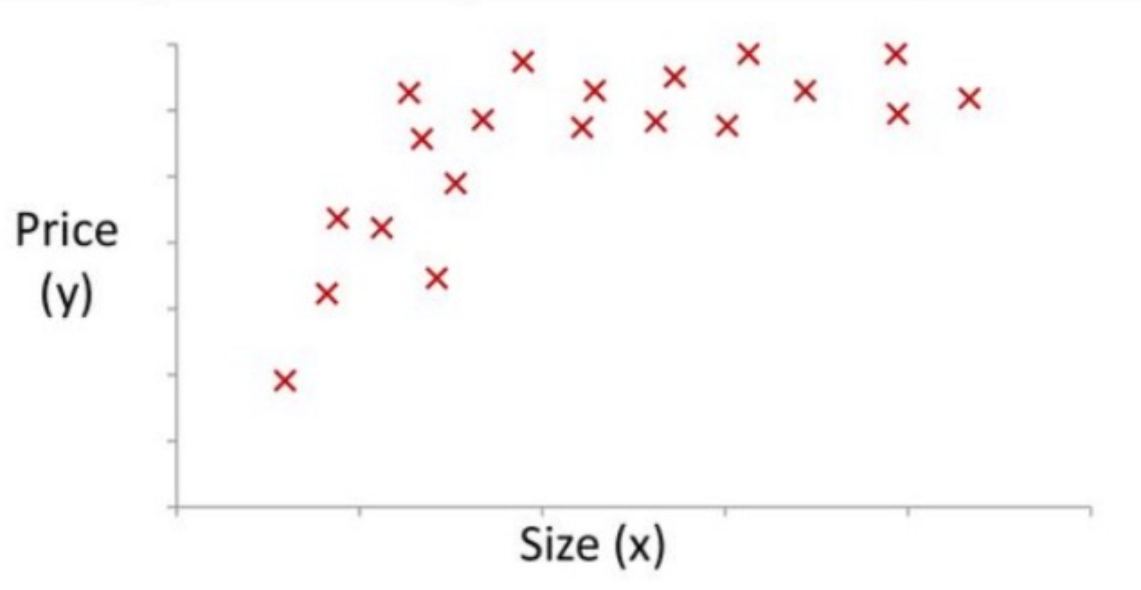
Features and Polynomial Regression

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$$

Polynomial Regression

Artificial Inteligence & Computer Vision Laboratory



$$\theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

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

$$= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$$

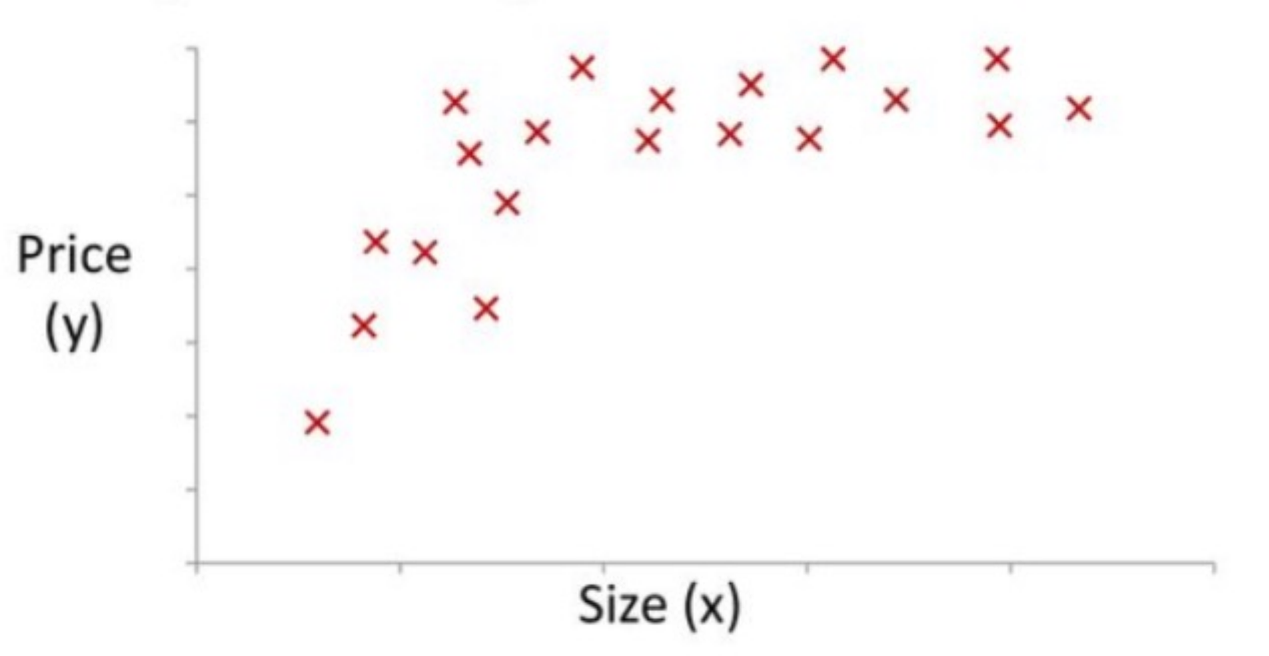
$$x_1 = (size)$$

$$x_2 = (size)^2$$

$$x_3 = (size)^3$$

Polynomial Regression

Choice of features



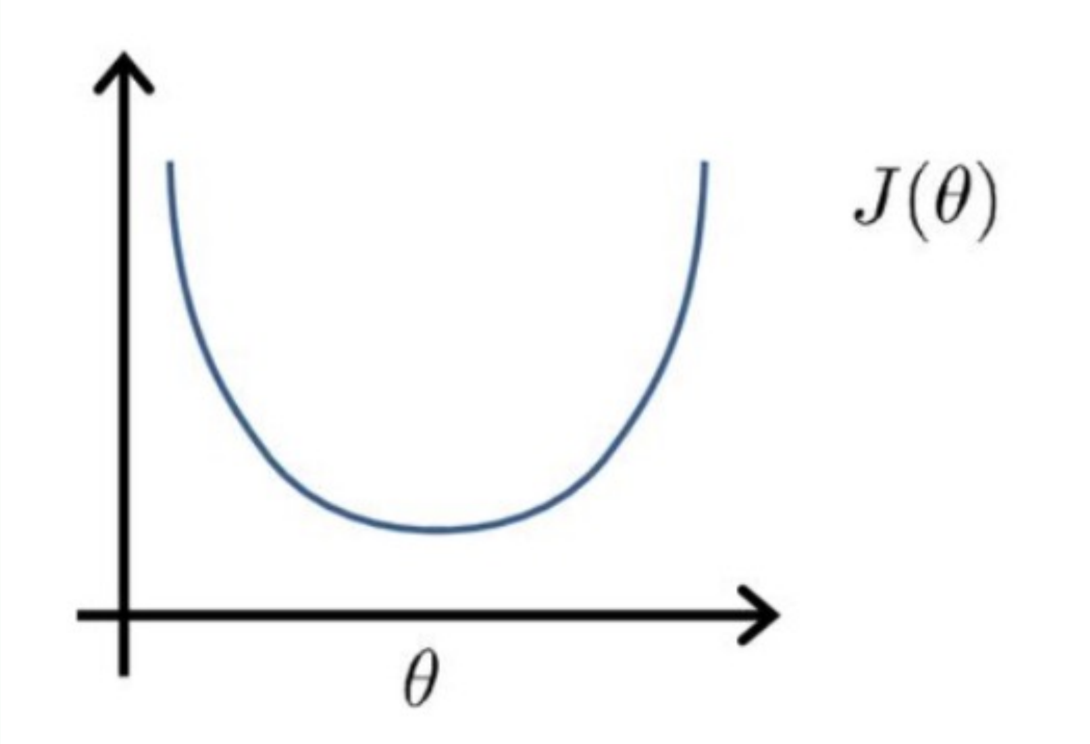
$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$$

Normal Equation

Normal Equation

Gradient Descent



Normal equation:

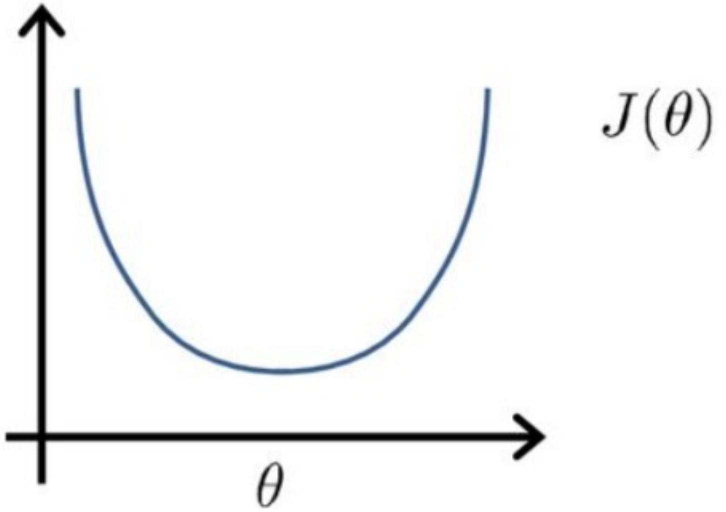
Method to solve for θ analytically

Normal Equation

Artificial Inteligence
& Computer Vision
Laboratory

Intuition: If 1D ($\theta \in \mathbb{R}$)

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



$\theta \in \mathbb{R}^{n+1}$ $\qquad J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots = 0 \qquad \text{(for every } j\text{)}$$

Solve for $\theta_0, \theta_1, \ldots, \theta_n$

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

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

$$\theta = \left(X^T X\right)^{-1} X^T y$$

Normal Equation

m=4 examples $(\mathbf{x}^{(1)},\mathbf{y}^{(1)}),\ldots,(\mathbf{x}^{(m)},\mathbf{y}^{(m)})$: **n features**

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

E.g. If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

Normal Equation

$$\theta = \left(X^T X\right)^{-1} X^T y$$

$\left(X^T X\right)^{-1}$ is inverse of matrix $X^T X$

Pinv( )

Normal Equation

Artificial Inteligence & Computer Vision Laboratory

m=4 examples $(\mathbf{x}^{(1)},\mathbf{y}^{(1)}),\ldots,(\mathbf{x}^{(m)},\mathbf{y}^{(m)})$: **n features**

Gradient Descent	Normal Equation
• Need to choose α.	• No need to choose α.
• Needs many iterations.	• Don't need to iterate.
• Works well even when n is large.	• Need to compute $(X^T X)^{-1}$
	• Slow if n is very large.

Normal Equation and non-invertibility

Artificial Inteligence & Computer Vision Laboratory

Norma equation

$$\theta = \left(X^T X\right)^{-1} X^T y$$

- What if $X^T X$ is non-invertible?

Normal Equation and non-invertibility

Artificial Inteligence & Computer Vision Laboratory

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent)

  E.g. $x_1 = \text{size in feet}^2$

  $x_2 = \text{size in } m^2$

- Too many features (e.g. $m \le n$)

  → Delete some features, or use regularization