ELTE | FACULTY OF INFORMATICS

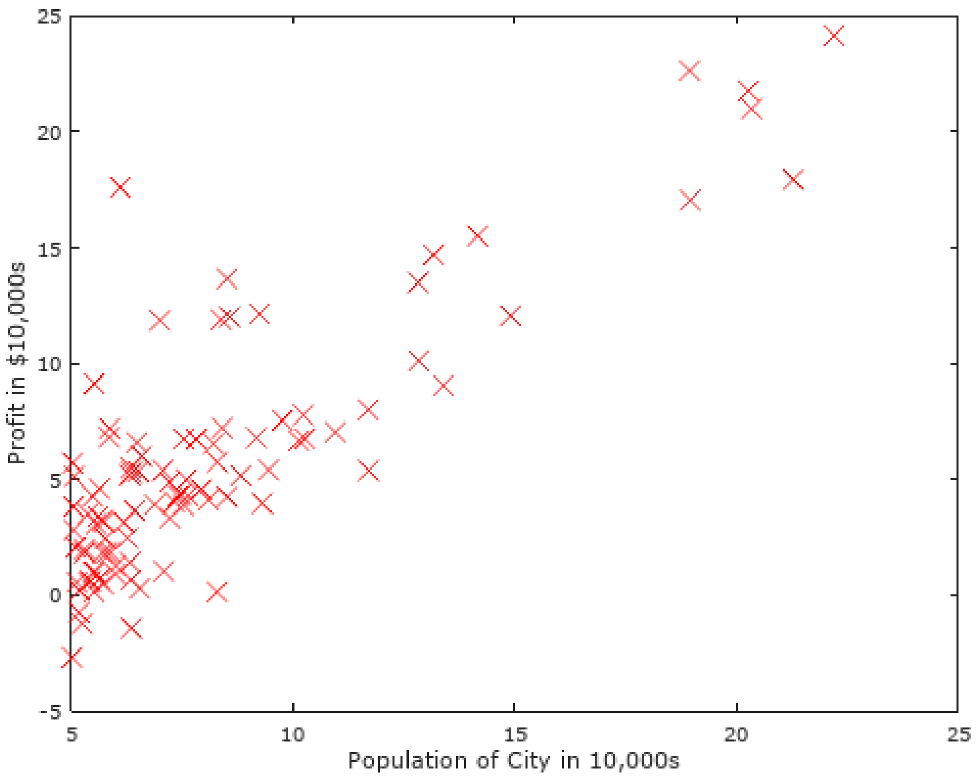
LINEAR REGRESSION

Machine Learning Course
Balázs Nagy, PhD

ELTE | IK
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Linear regression with one variable

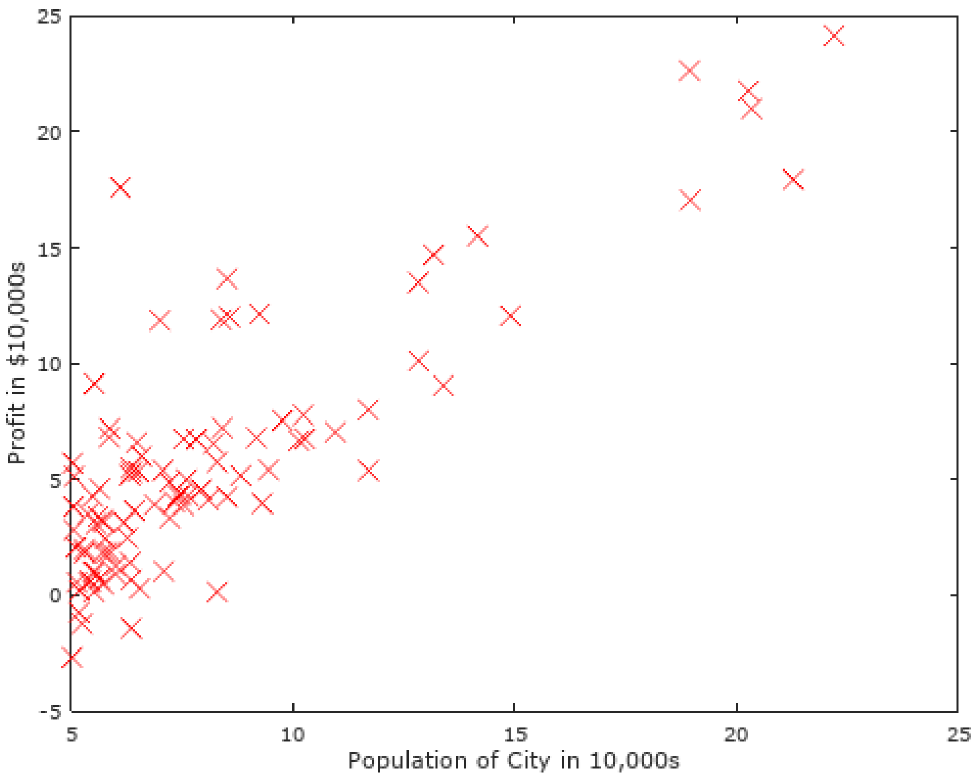
You have data for profits and populations from different cities. You would like to use this data to help you select which city to expand your food truck company.



One variable linear regression

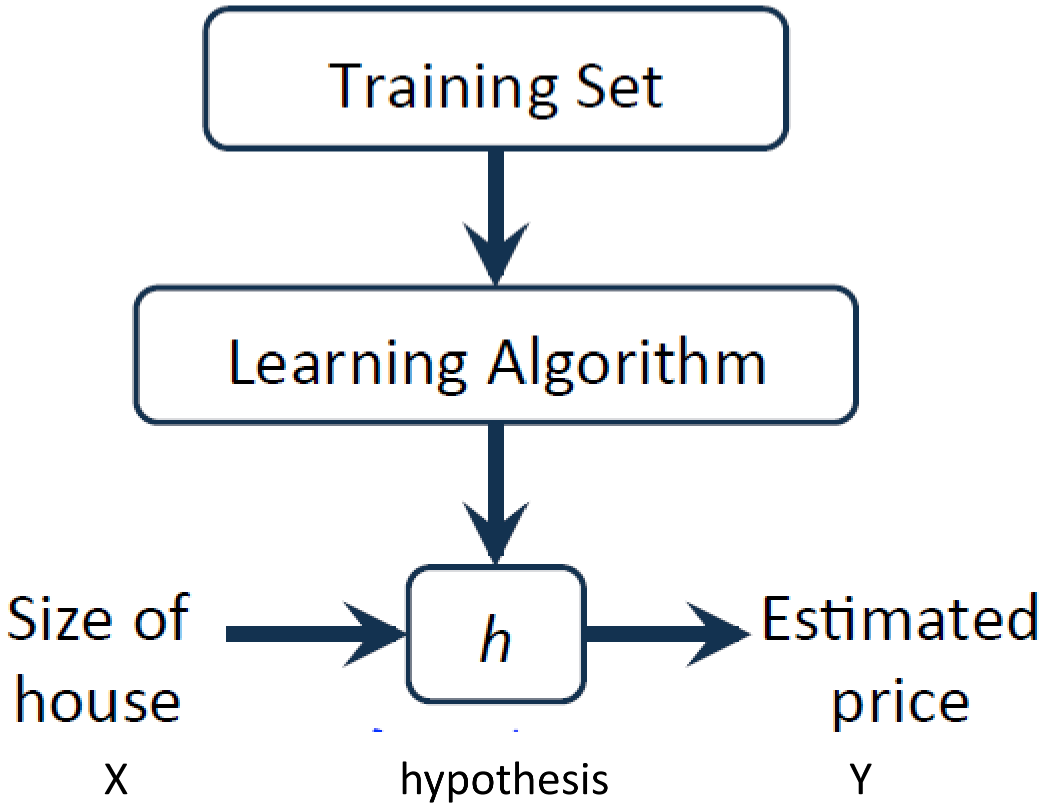
You have data for profits and populations from different cities. You would like to use this data to help you select which city to expand your food truck company.

It can be a Supervised Learning Task

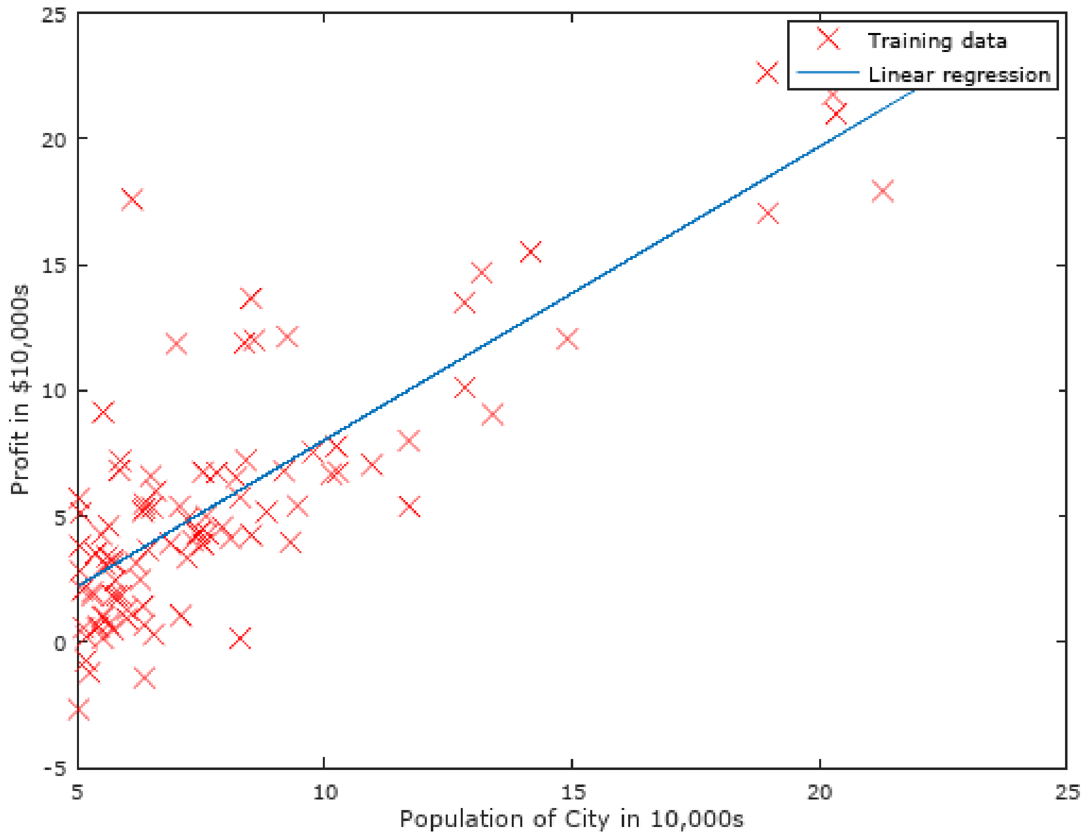


Model

- m – number of training examples
- x – input variable
- y – output variable
- (x, y) – one training example
- $(x^{(i)}, y^{(i)})$ – i^{th} training example



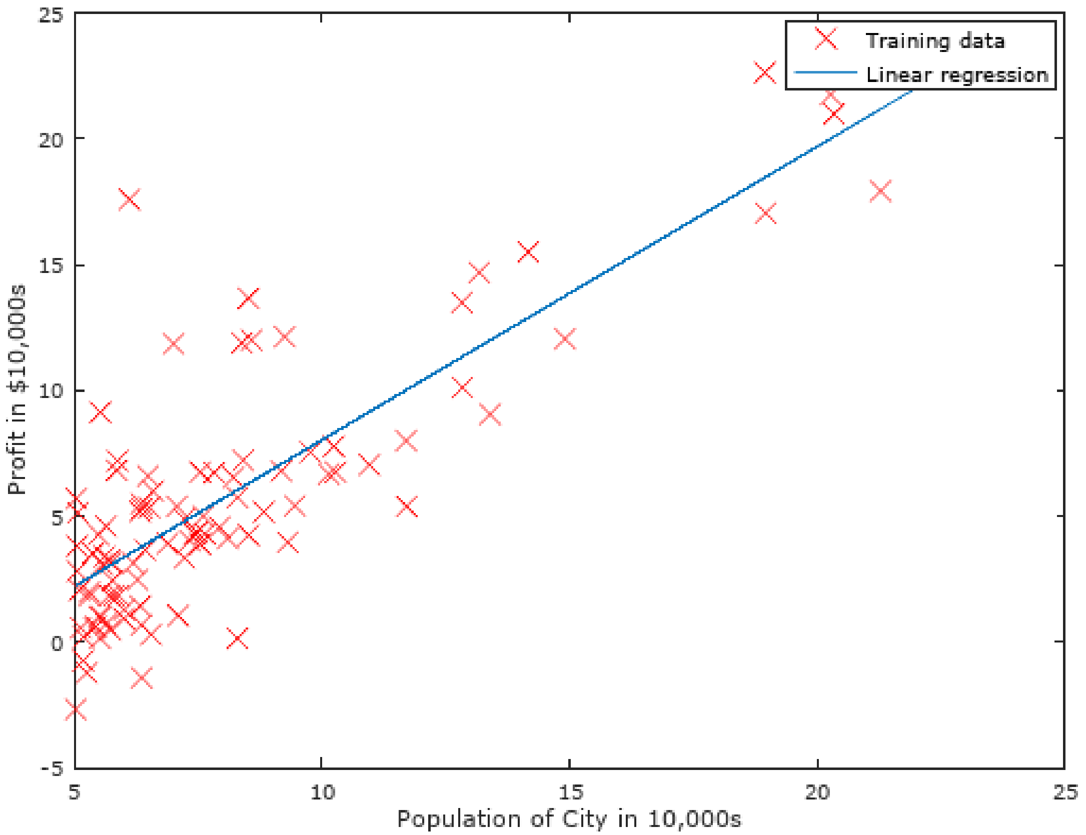
Hypothesis & Cost Function (linear Case)



Hypothesis & Cost Function (linear Case)

Hypothesis:

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



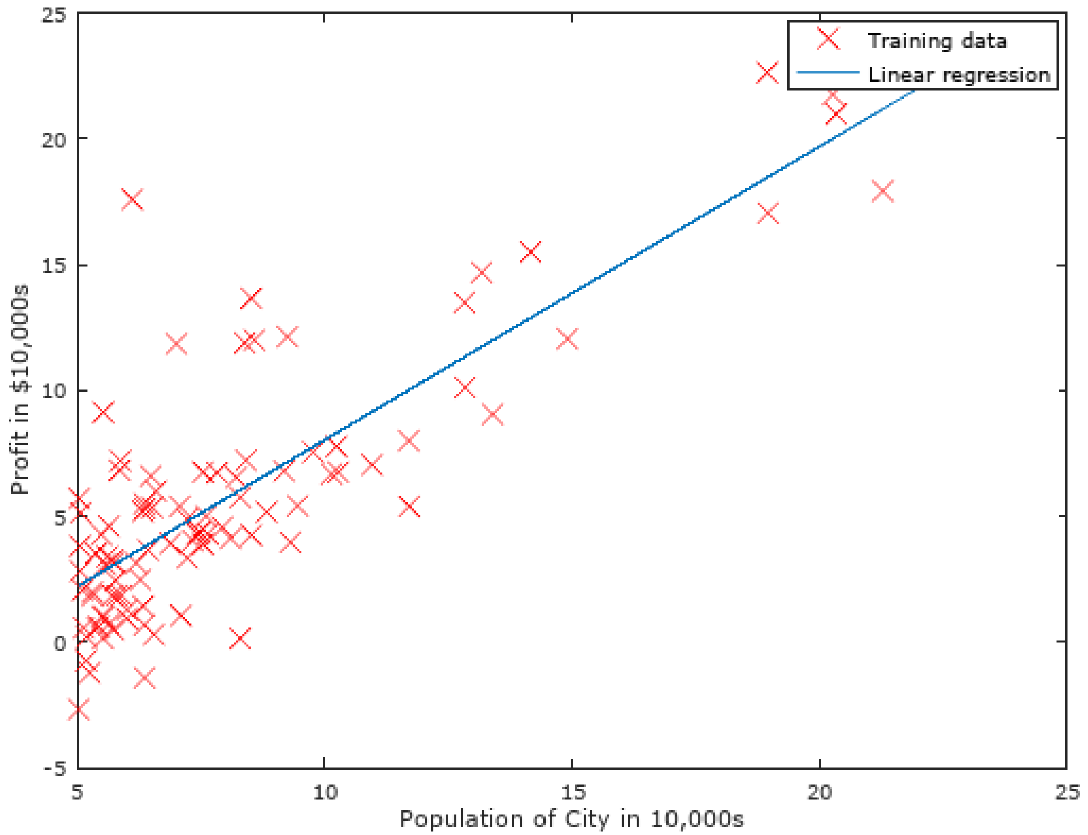
Hypothesis & Cost Function (linear Case)

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$



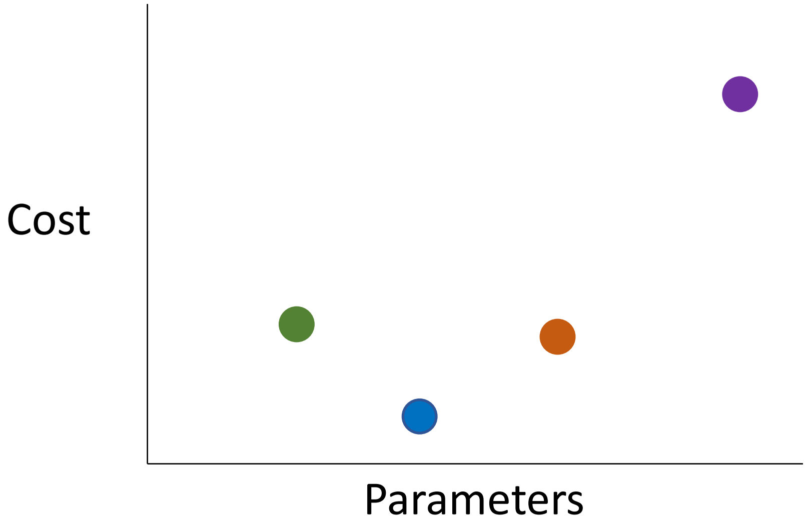
Hypothesis & Cost Function (linear Case)

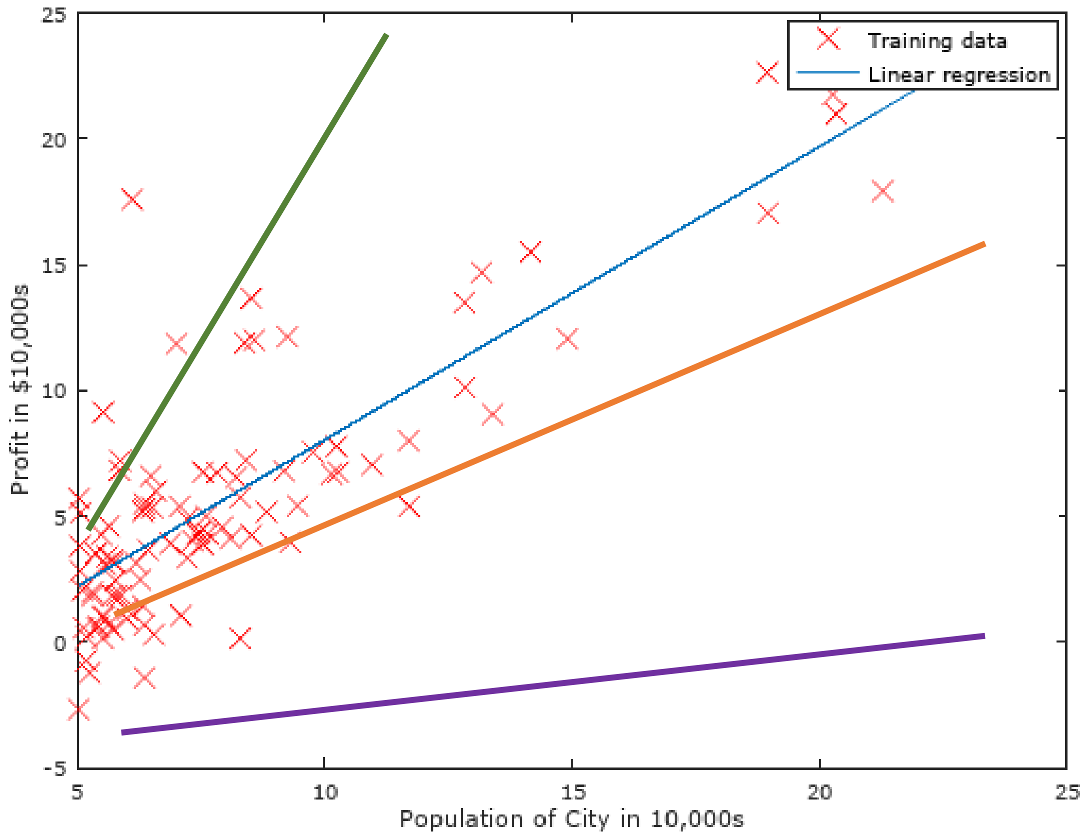
Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$





Hypothesis & Cost Function (linear Case)

Hypothesis:

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

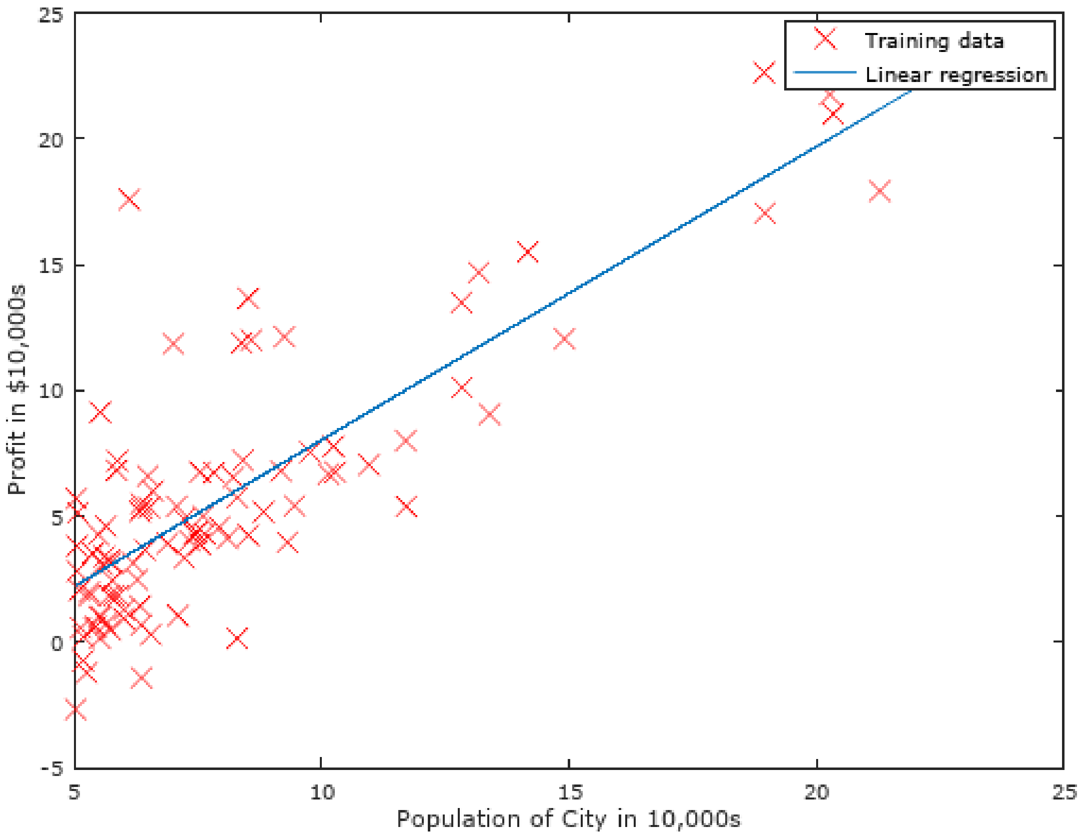
Parameters:

$$\theta_0, \theta_1$$

Mean Squared Error (MSE)

Cost Function:

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



Hypothesis & Cost Function (linear Case)

Hypothesis:

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

Parameters:

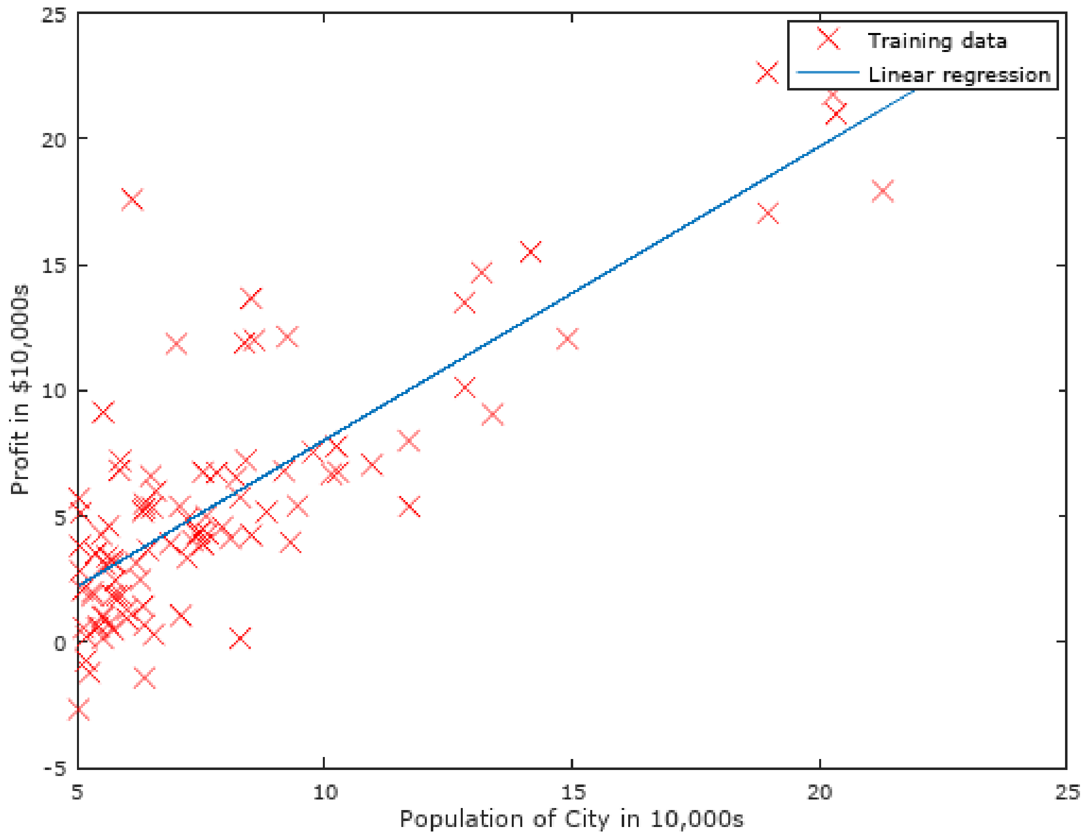
$$\theta_0, \theta_1$$

Mean Squared Error (MSE)

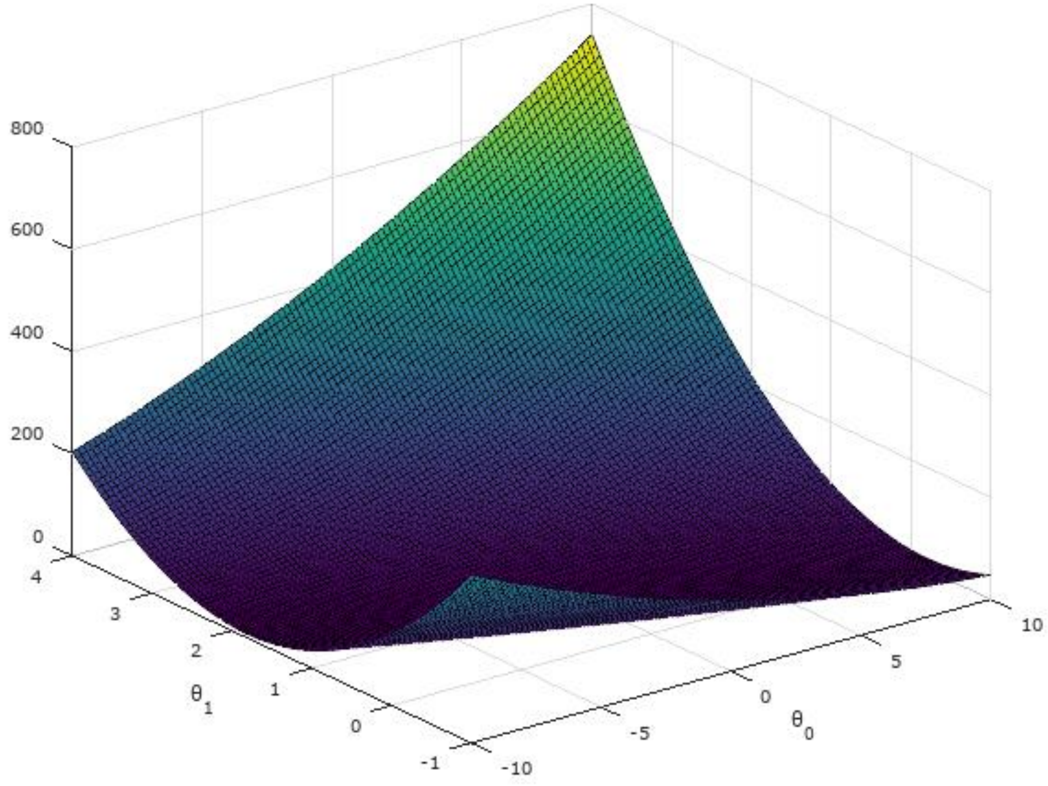
Cost Function:

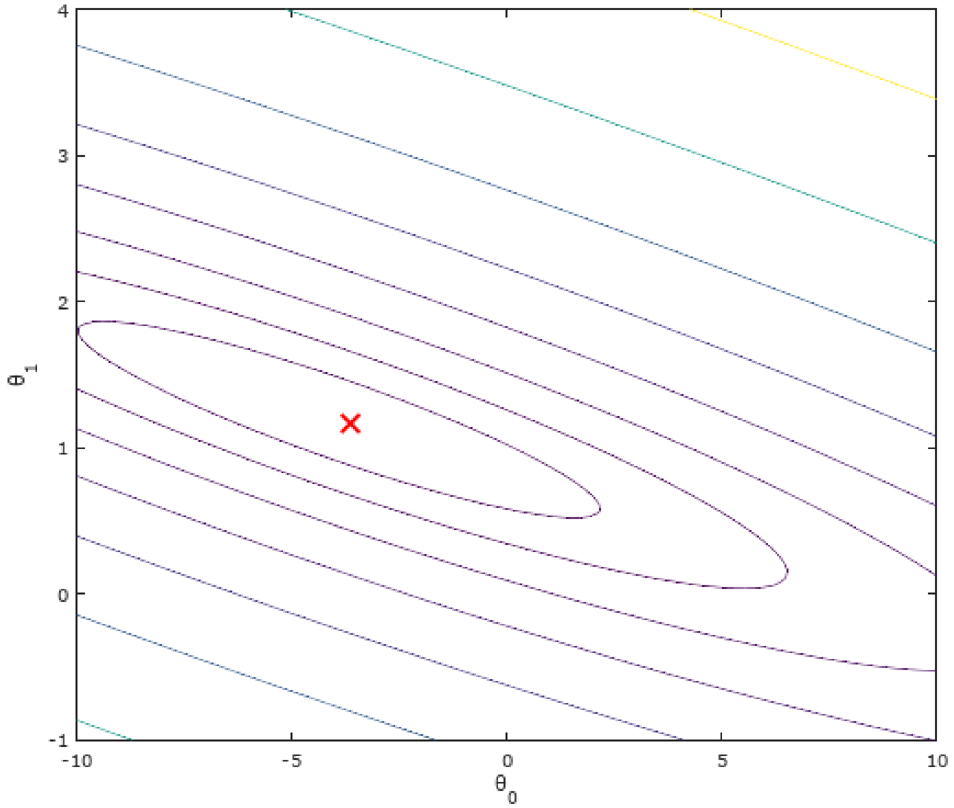
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

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



Cost function

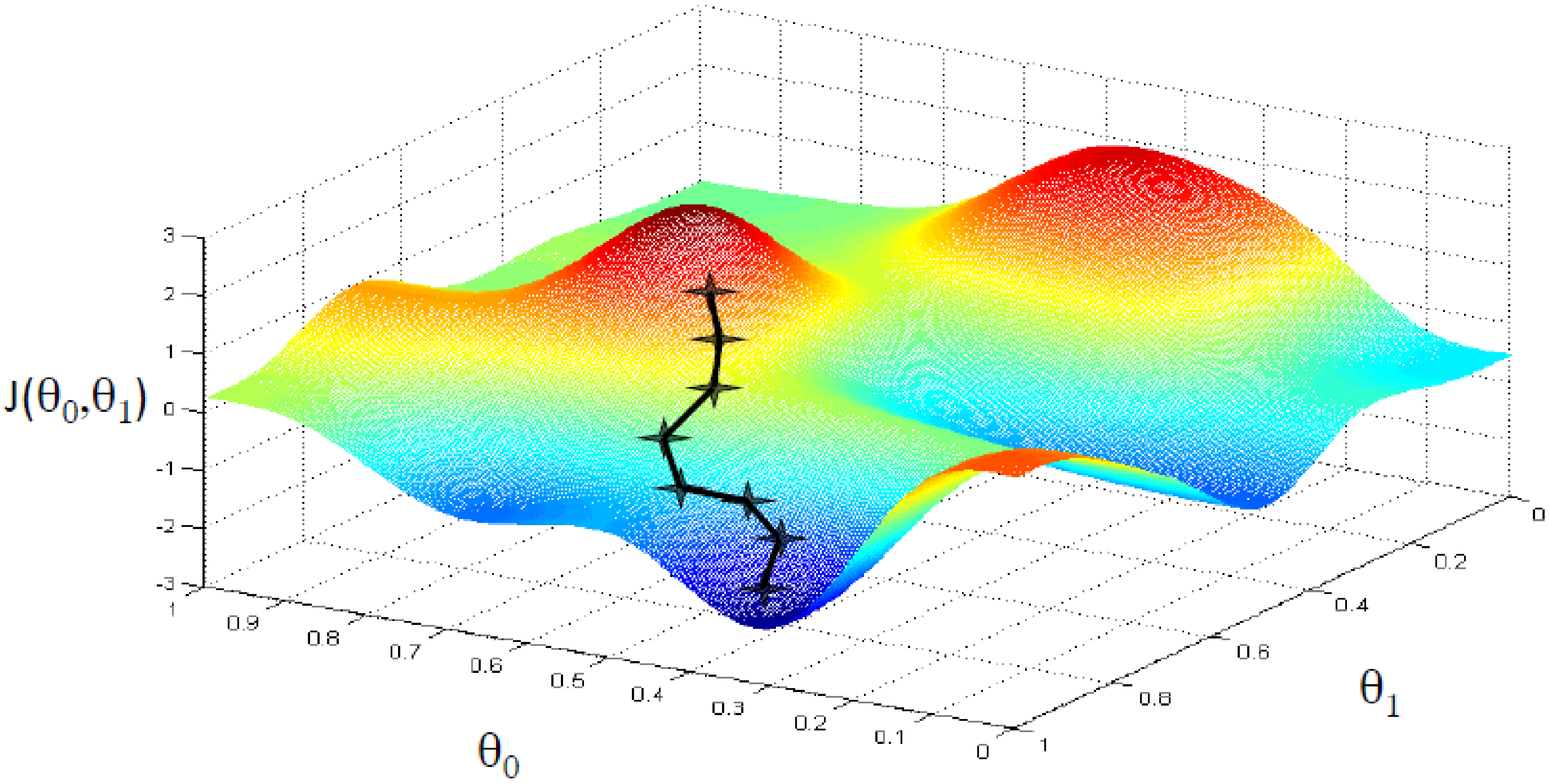




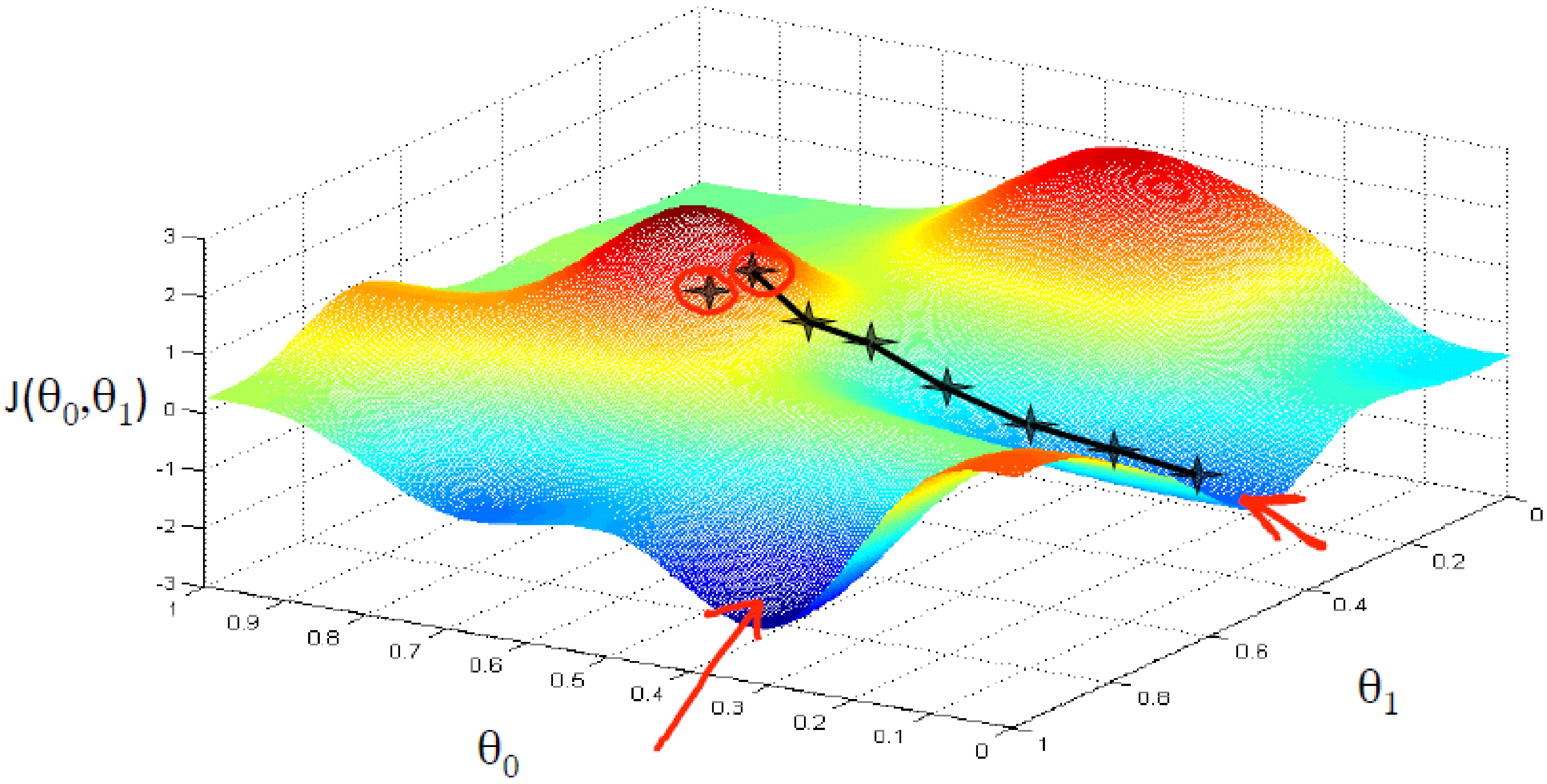
Gradient Descent method

- Have some cost function: $J(\Theta_0, \Theta_1)$
- Want to minimize cost function
- Outline:
  - Start with some Θ_0, Θ_1 ($\Theta_0=0$, $\Theta_1=0$)
  - Keep changing Θ_0, Θ_1 to reduce $J(\Theta_0, \Theta_1)$ until we hopefully end up at a minimum.

Gradient Descent visualization



Gradient Descent visualization



Gradient Descent algorithm

repeat until convergence {

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

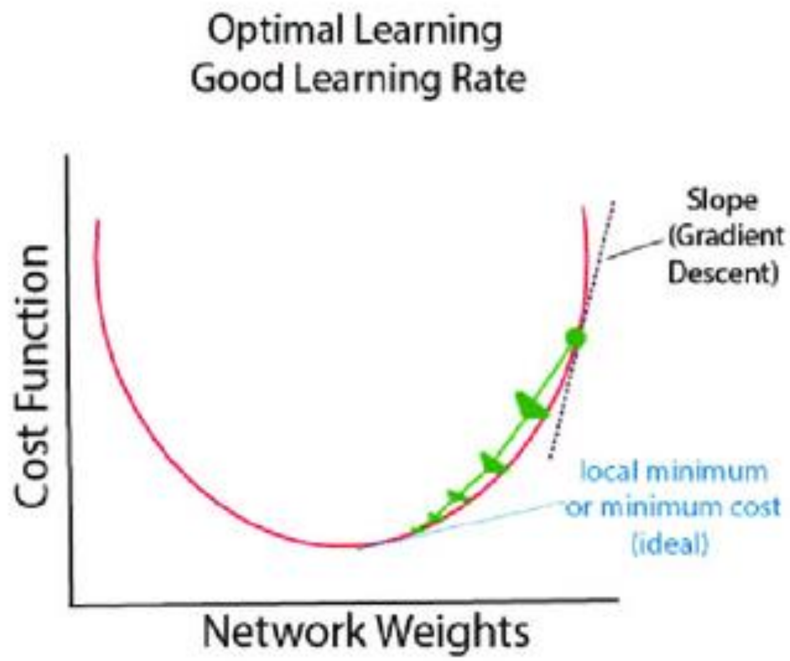
}

Correct: Simultaneous update

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

Learning rate

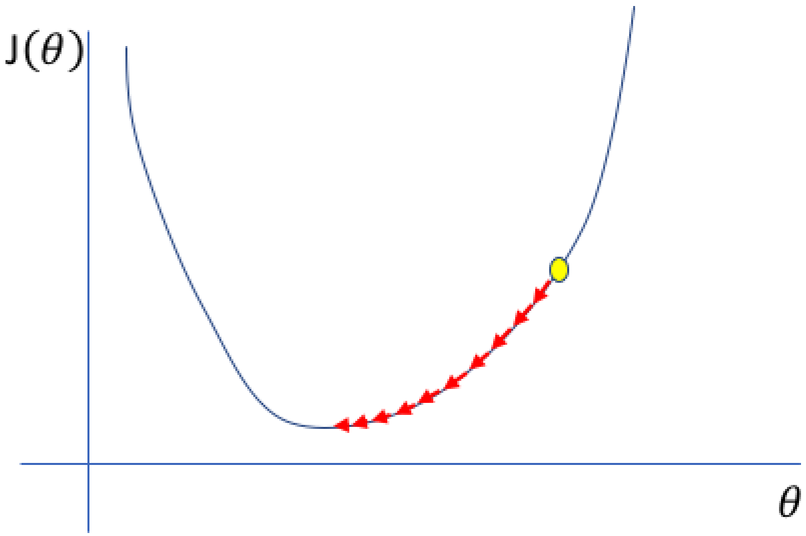
$$\theta_1 := \theta_1 - \boxed{\alpha} \frac{d}{d\theta_1} J(\theta_1)$$



- 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.

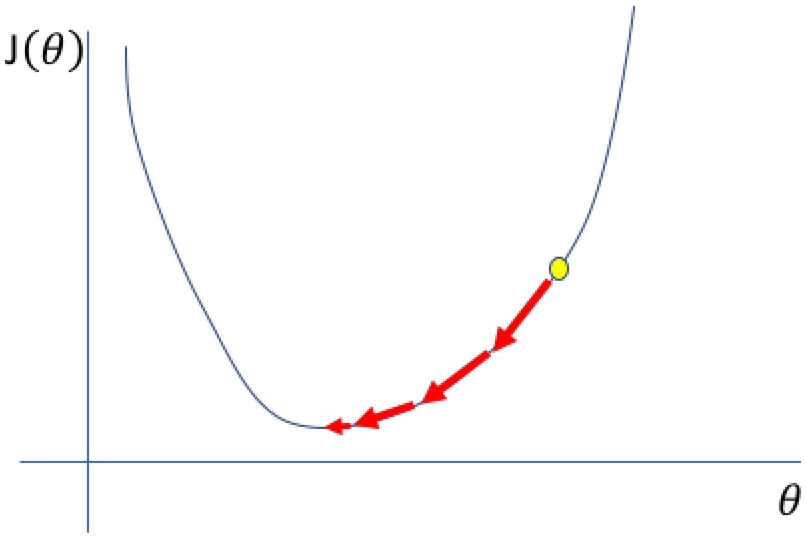
Learning rate

Too low



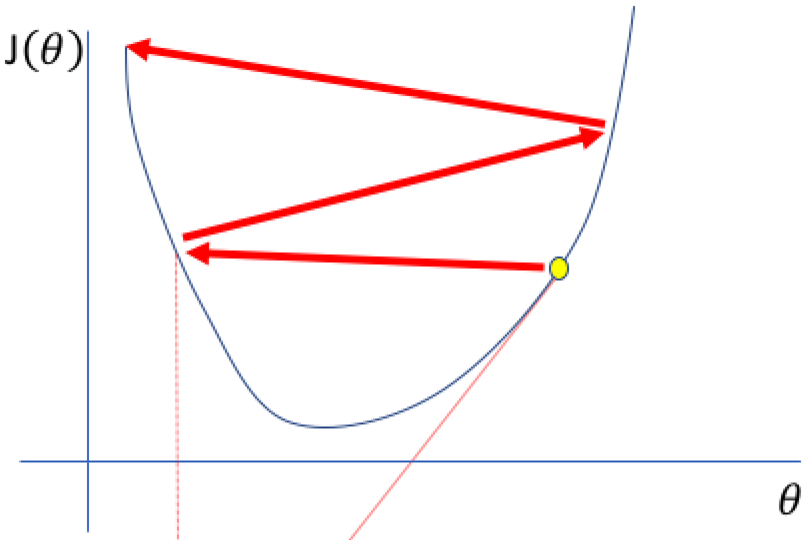
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

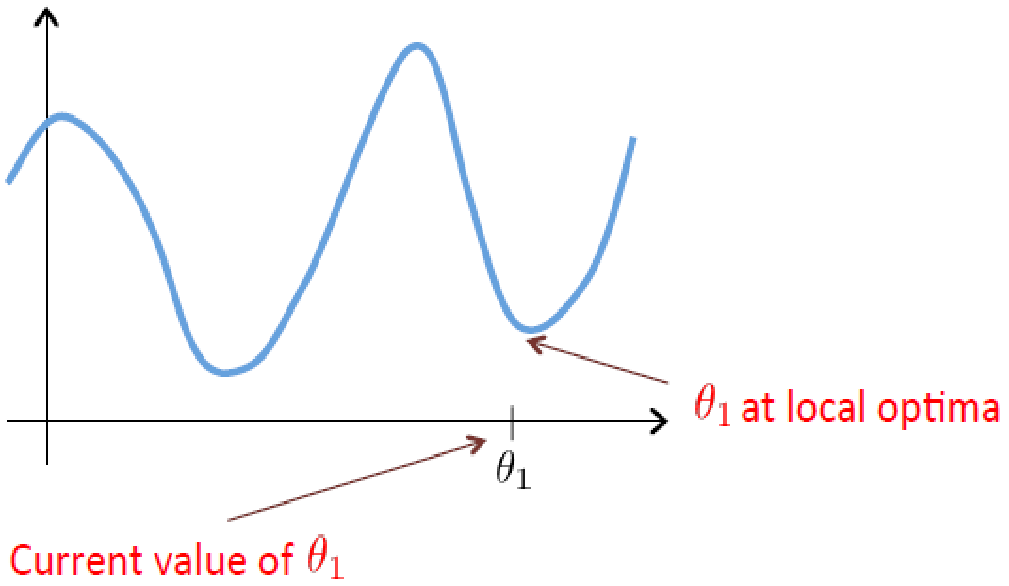
Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviors

Learning rate

- Gradient descent can converge to a local minimum, even with a fixed learning rate.
- As we approach a local minimum, gradient descent will automatically take smaller steps. No need to decrease α over time.



Modell integration

Gradient descent algorithm

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

Linear Regression Model

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

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

Modell integration

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

where the hypothesis $h_\theta(x)$ is given by the linear model

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

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

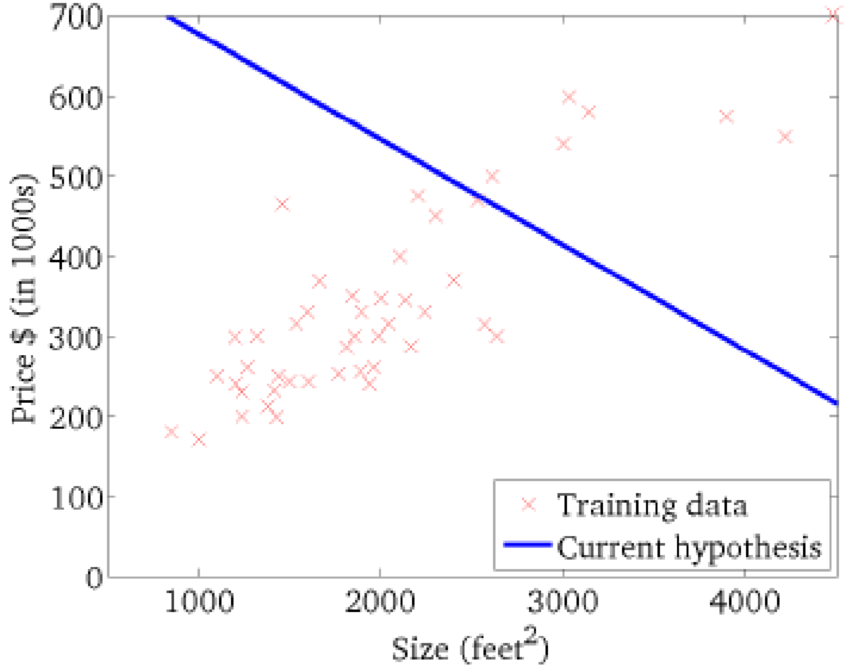
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$$

- „Batch” Gradient Descent: Each step of gradient descent uses all the training examples.

Example

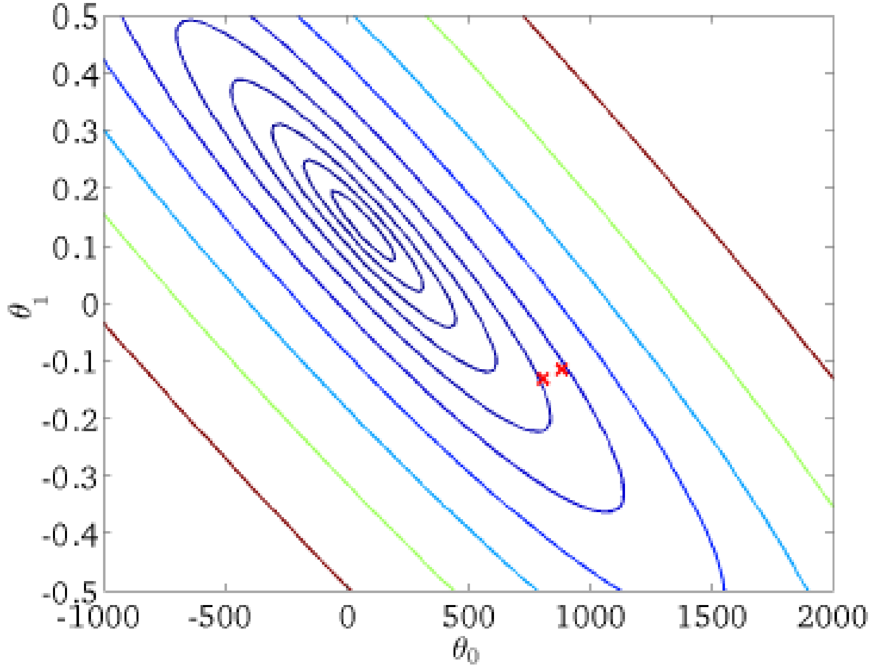
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

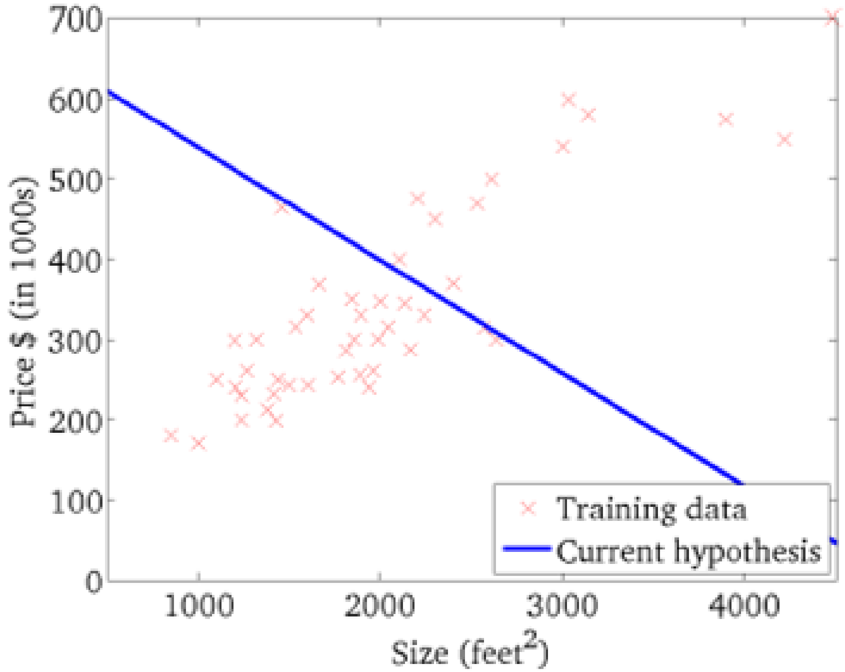
(function of the parameters θ_0, θ_1)



Example

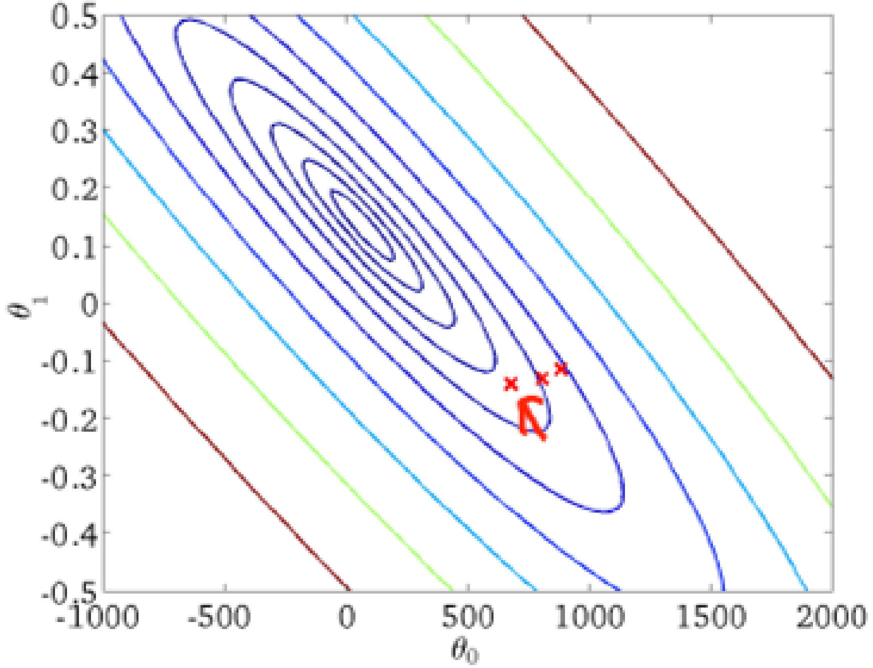
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

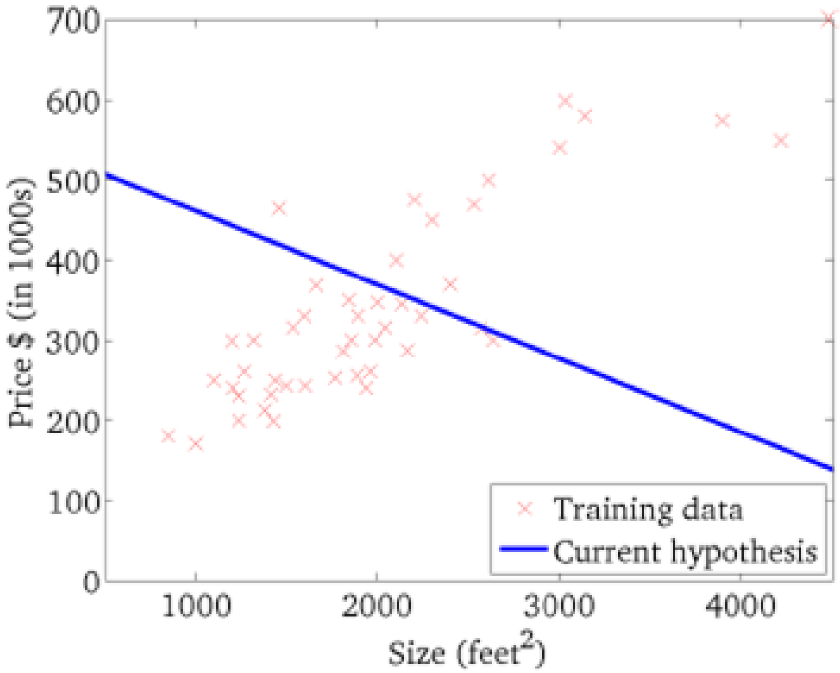
(function of the parameters θ_0, θ_1)



Example

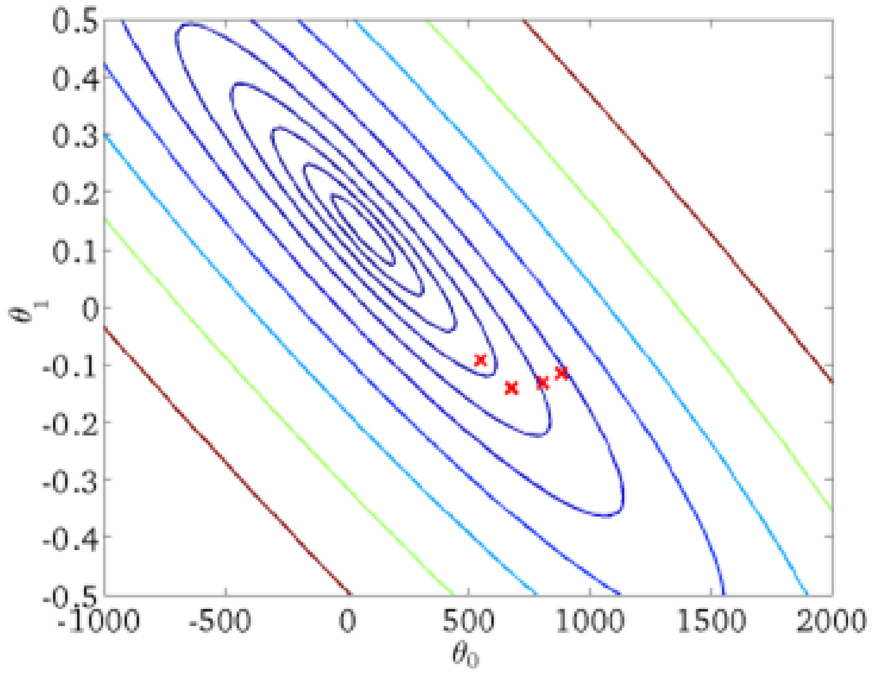
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

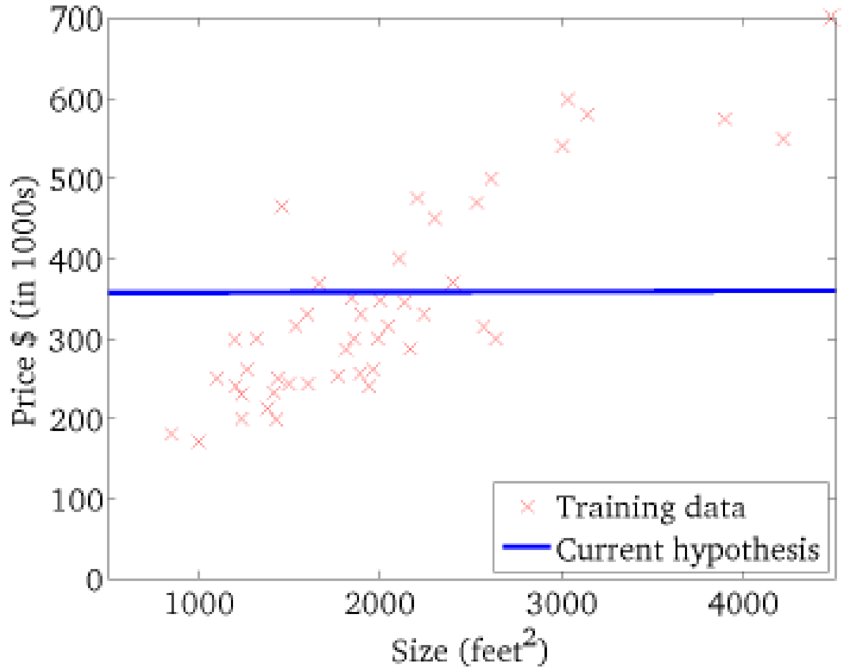
(function of the parameters θ_0, θ_1)



Example

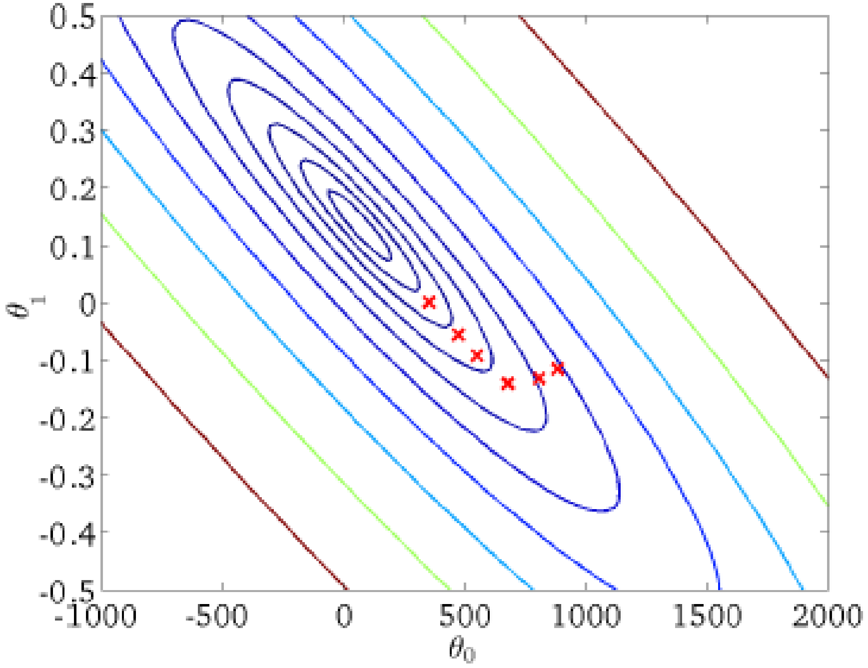
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

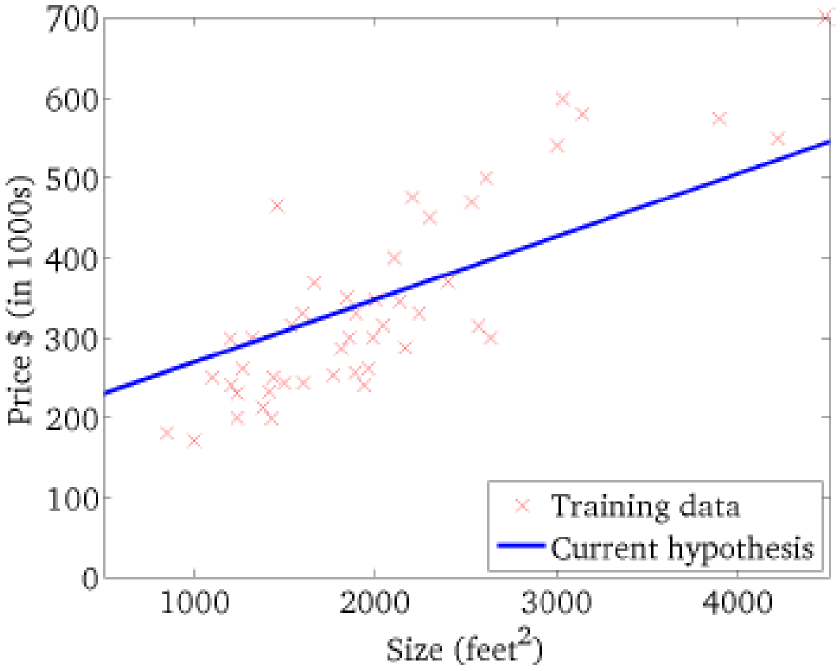
(function of the parameters θ_0, θ_1)



Example

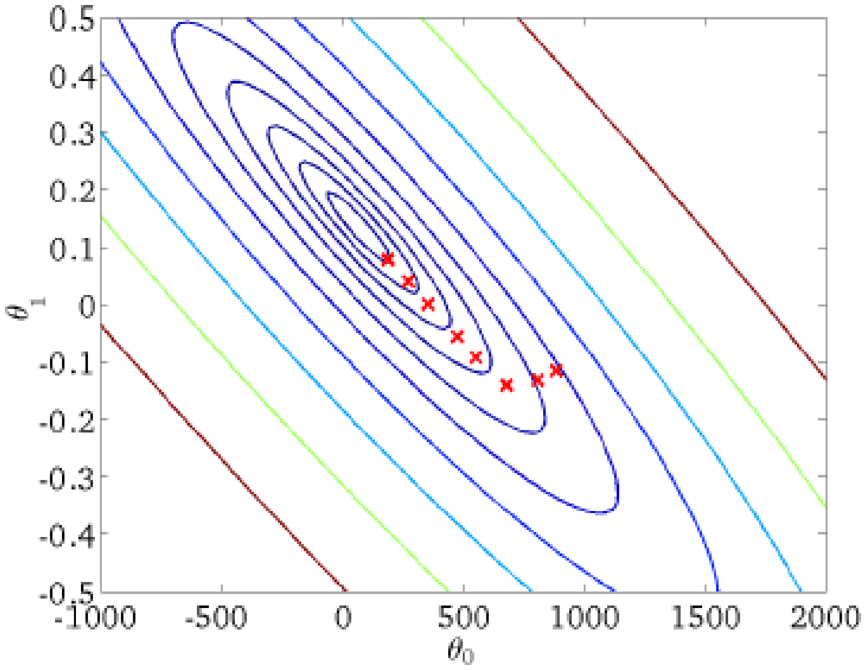
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

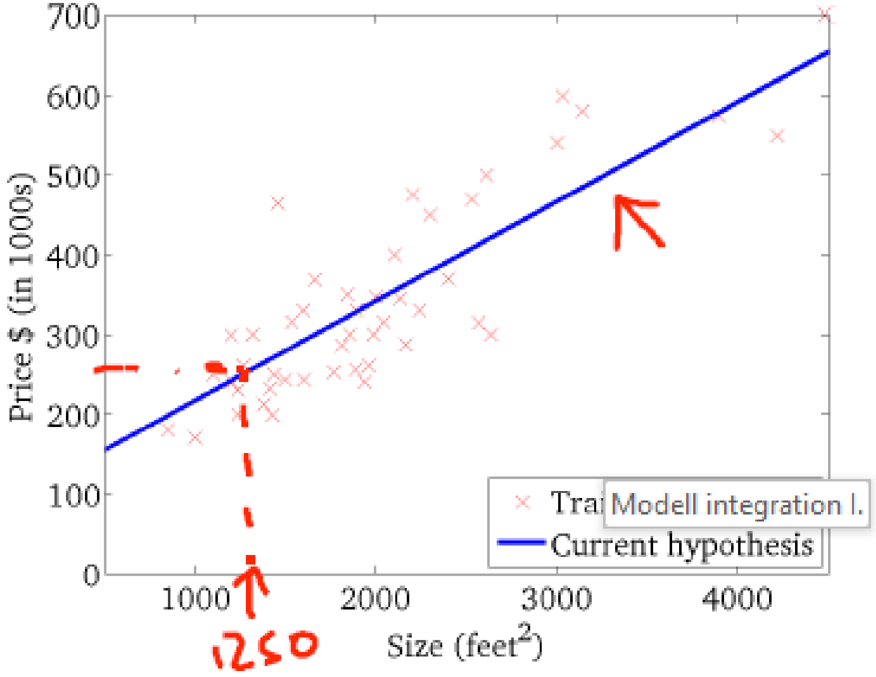
(function of the parameters θ_0, θ_1)



Example

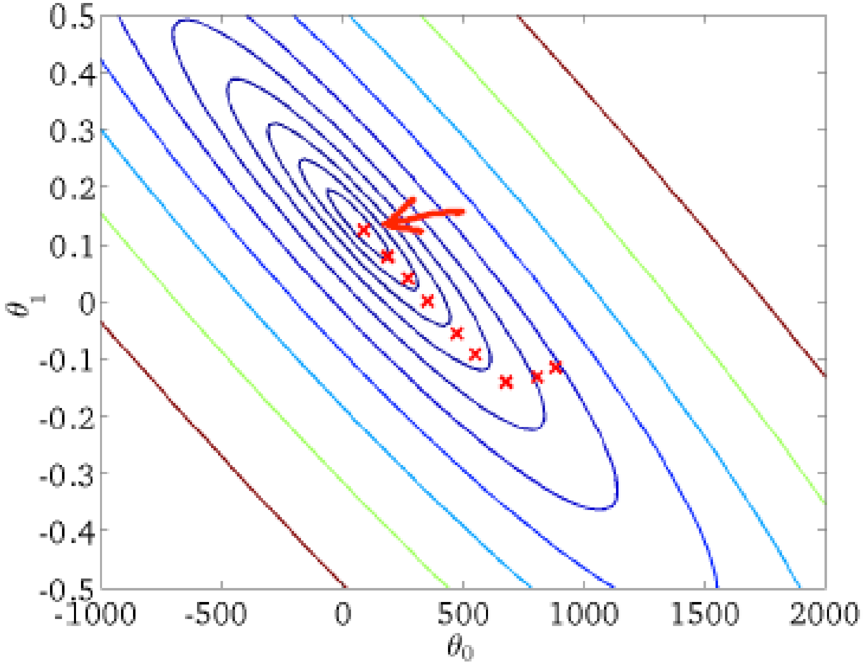
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Linear regression with multiple features

One variable

Size (feet²) x	Price ($1000) y
2104	460
1416	232
1534	315
852	178
…	…

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

Linear regression with multiple features

One variable

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

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

Multiple variables

Size (feet²)	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
...	...	...	...	...

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

For convenience of notation, define $x_0 = 1$.

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.

Linear regression with multiple features

$$x_j^{(i)} = \text{value of feature } j \text{ in the } i^{th} \text{ training example}$$
$$x^{(i)} = \text{the input (features) of the } i^{th} \text{ training example}$$
$$m = \text{the number of training examples}$$
$$n = \text{the number of features}$$

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

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

$$X = \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ & \vdots & \\ - & (x^{(m)})^T & - \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}.$$

$$x_0^{(i)} = 1 \text{ for } (i \in 1, \ldots, m).$$

$$J(\theta) = \frac{1}{2m} (X\theta - \vec{y})^T (X\theta - \vec{y})$$

- This allows us to do matrix operations.

Modifying Gradient Descent algorithm

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 $j = 0, \ldots, n$)

$\}$

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

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

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

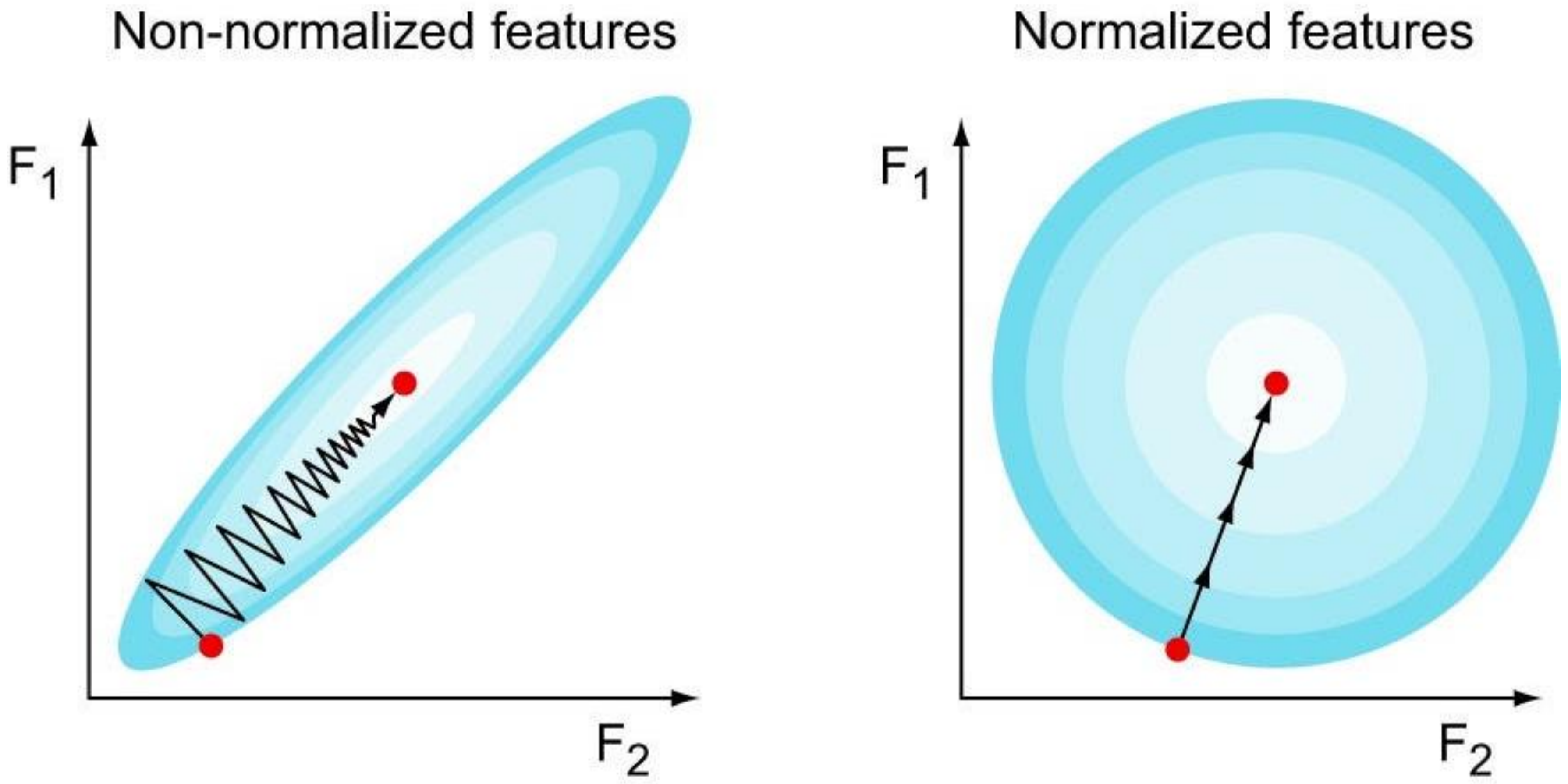
Feature Scaling

- If the variables have different ranges, it can slow down the convergence
  - For example: x_1 = size (0-2000 m^2)
    x_2 = number of bedrooms (1-5)
- Get every feature into approximately a -1… +1 range.
  - **Feature scaling**
  - **Mean Normalization**

$$x_i := \frac{x_i - \mu_i}{s_i}$$

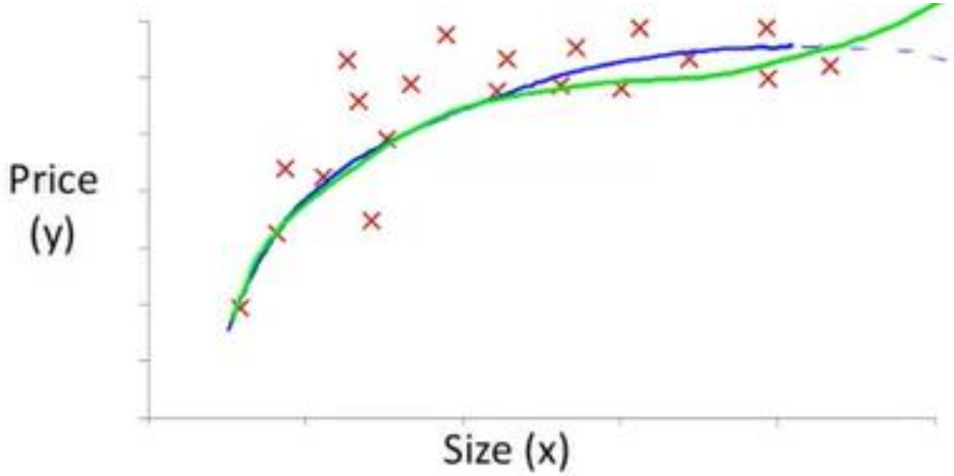
Where μ_i is the **average** of all the values for feature (i) and s_i is the range of values (max - min), or s_i is the standard deviation.

Feature Scaling



Polynomial regression

- Sometimes the given features are not enough or sufficient
- Need more parameters



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\boxed{\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3}$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$$

$$x_1 = (size)$$
$$x_2 = (size)^2$$
$$x_3 = (size)^3$$

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

$$X = \underbrace{\begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}}_{m \times (n+1)} \qquad y = \underbrace{\begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}}_{m\text{-dimensional vector}}$$

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

Overview

Gradient Descent

- Need to chosse α
- Needs many iterations
- Works well even when n is large

Normal Equation

- No need to chose α
- Don't need to iterate
- Need to compute $(X^TX)^{-1}$
- Slow if n is very large

ELTE | FACULTY OF INFORMATICS

Thank you for your attention!