ELTE | FACULTY OF INFORMATICS

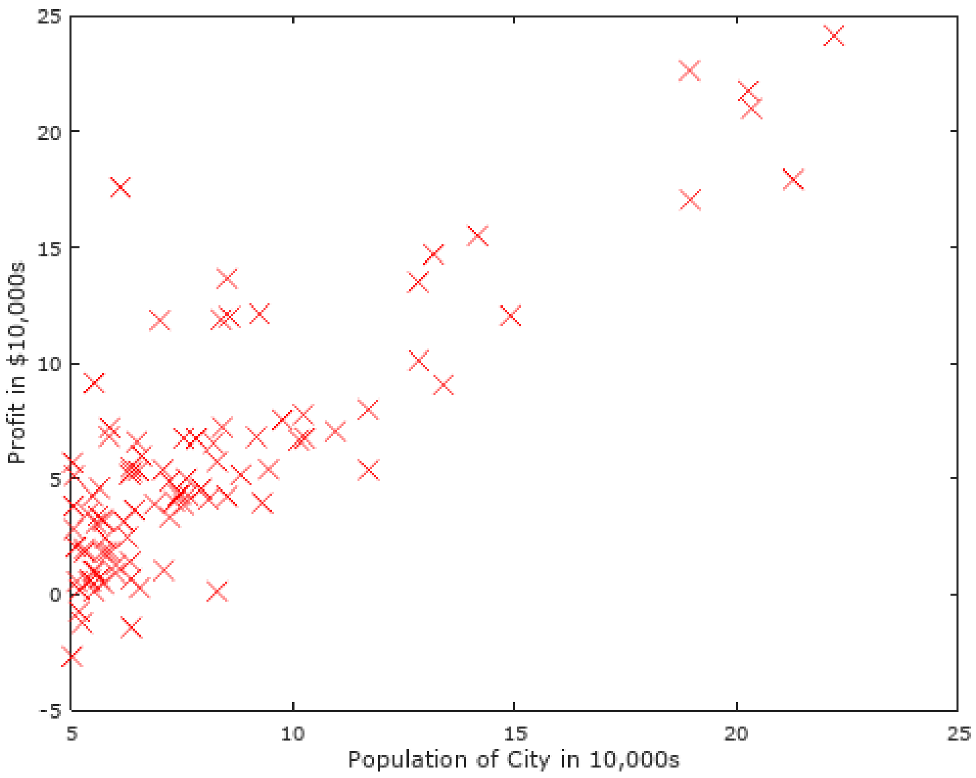
LINEAR REGRESSION

Machine Learning Course
Balázs Nagy, PhD

ELTE | IK
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Linear regression with one variable

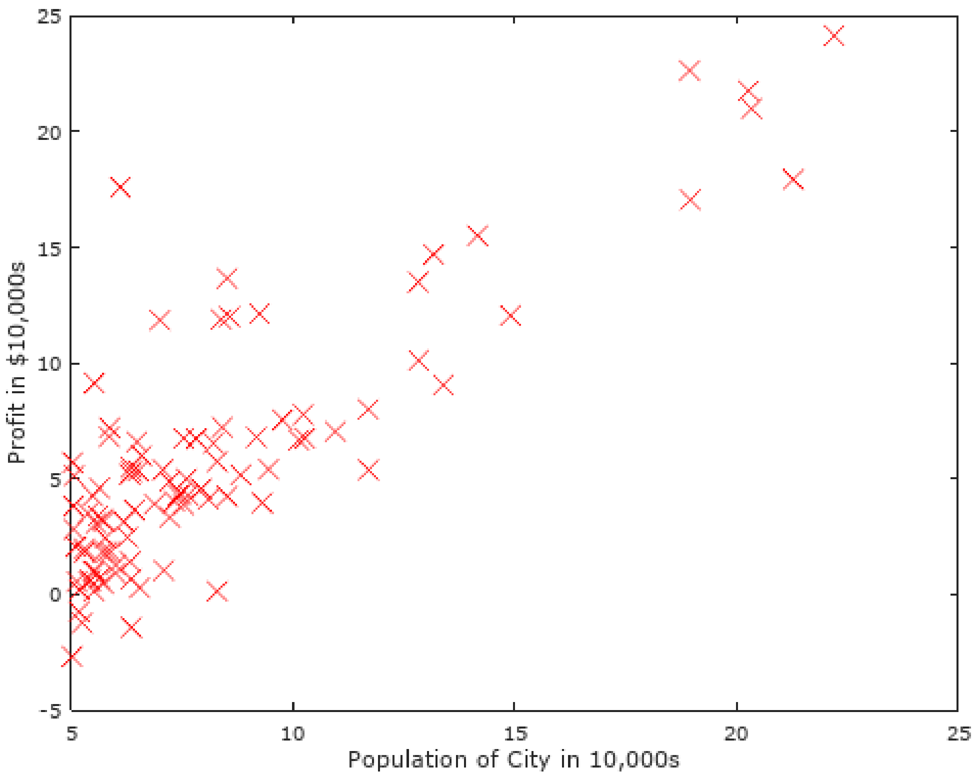
You have data for profits and populations from different cities. You would like to use this data to help you select which city to expand your food truck company.



One variable linear regression

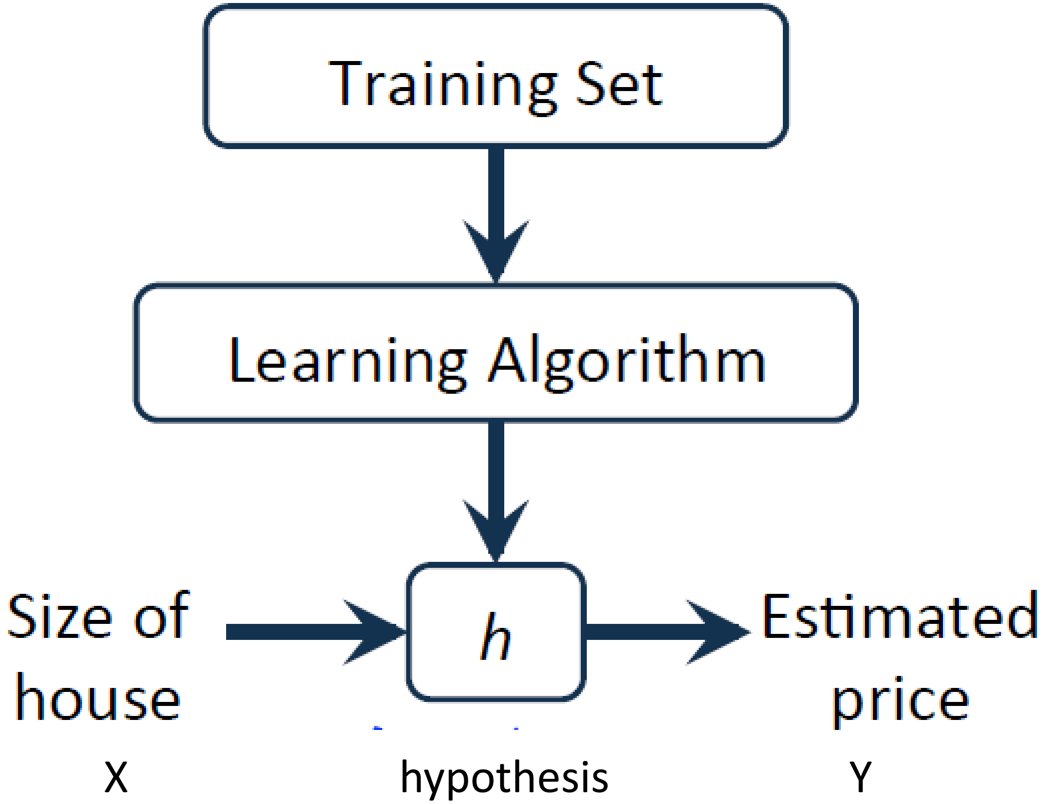
You have data for profits and populations from different cities. You would like to use this data to help you select which city to expand your food truck company.

It can be a Supervised Learning Task

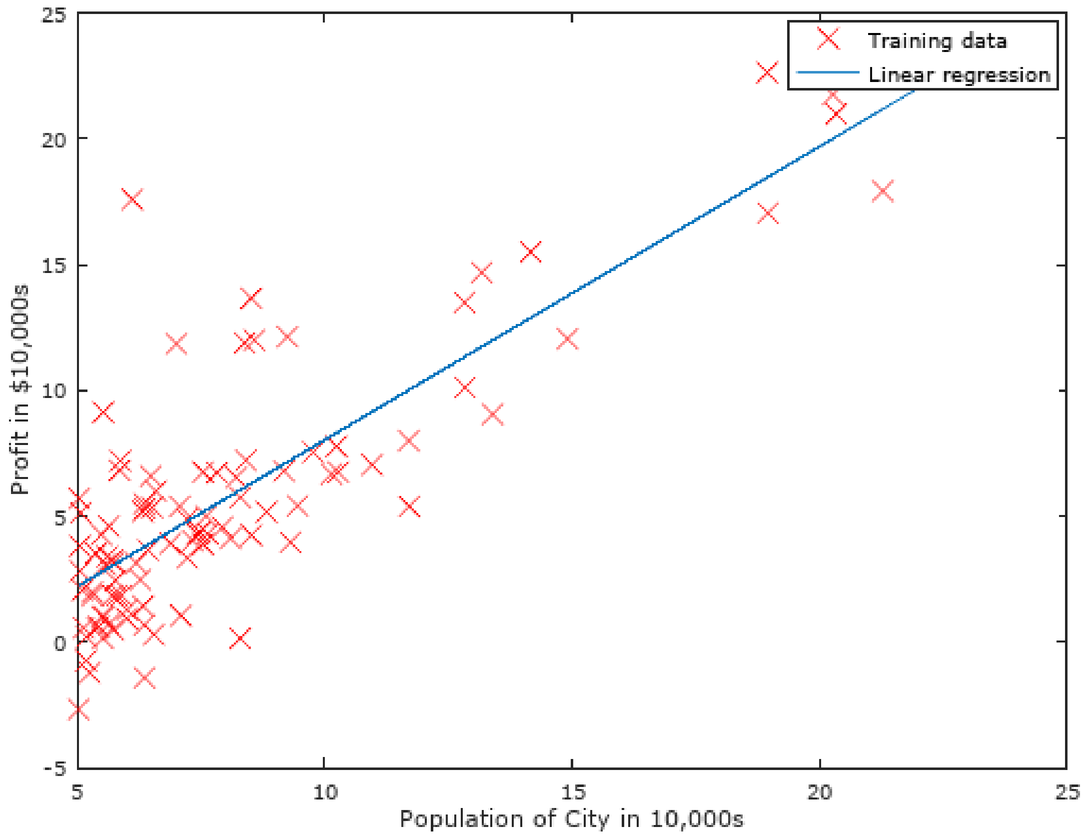


Model

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



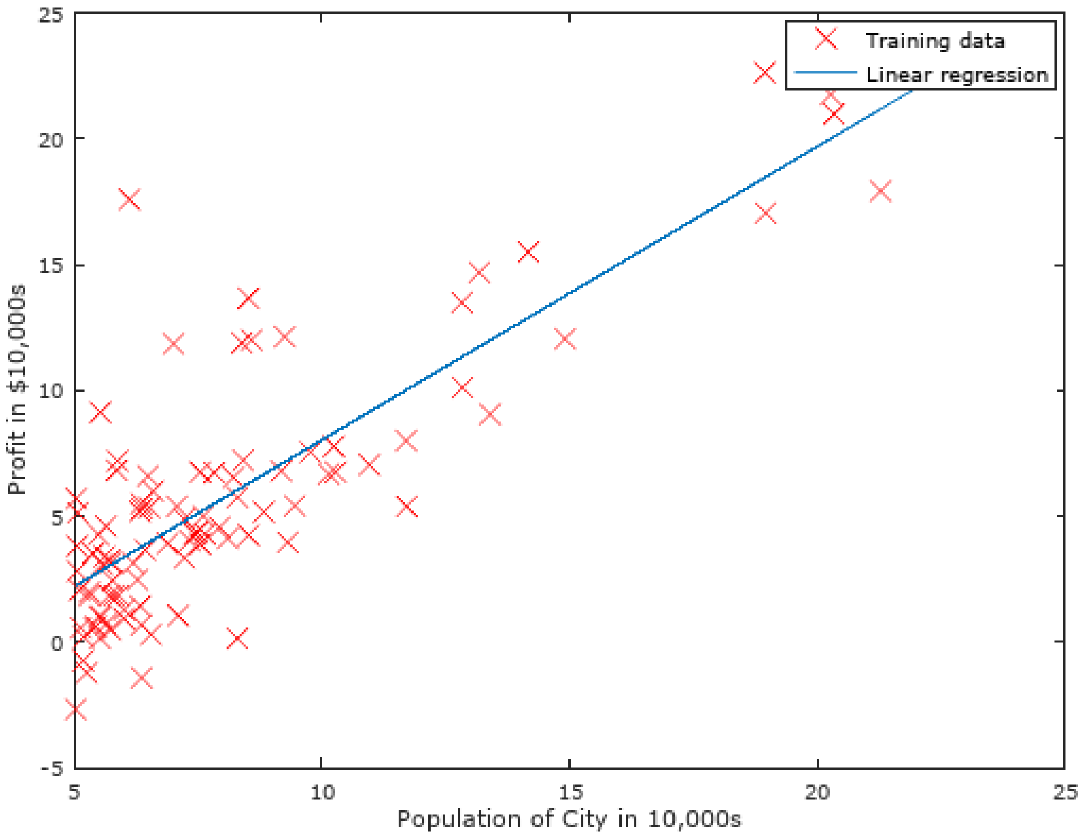
Hypothesis & Cost Function (linear Case)



Hypothesis & Cost Function (linear Case)

Hypothesis:

$$h_w(x) = w_0 + xw_1 = \hat{y}$$



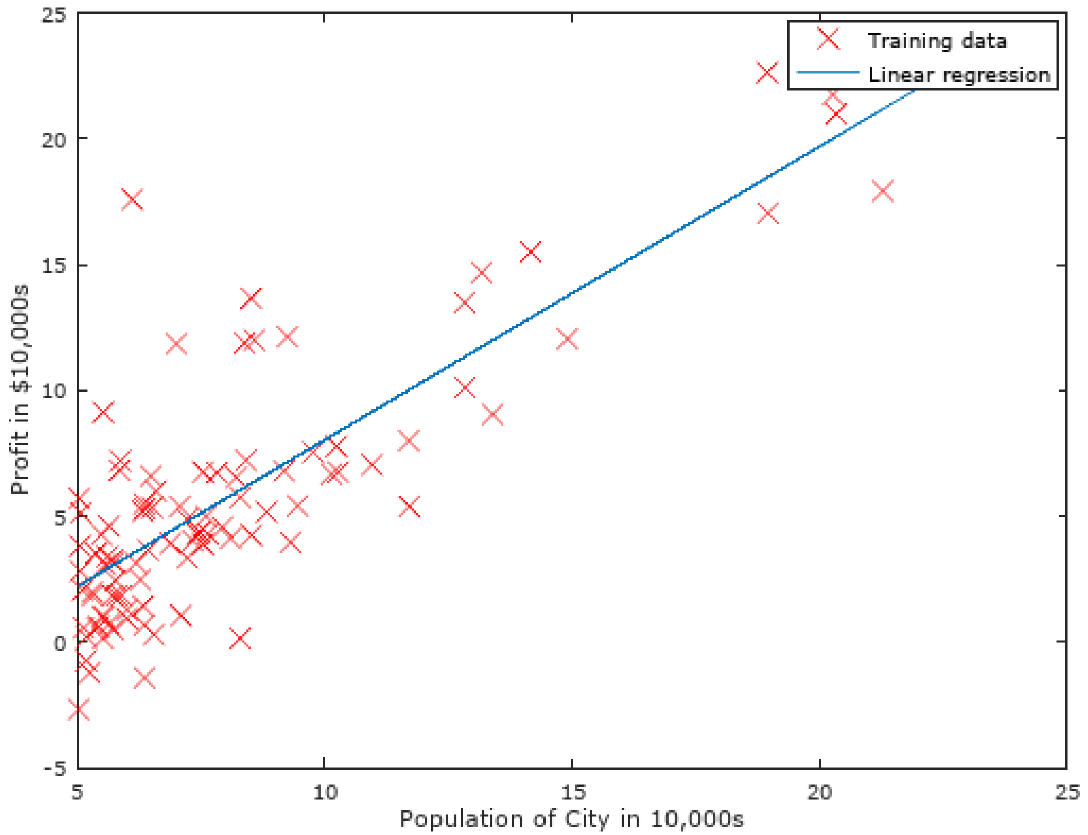
Hypothesis & Cost Function (linear Case)

Hypothesis:

$$h_w(x) = w_0 + x w_1 = \hat{y}$$

Parameters:

$$w_0, w_1$$



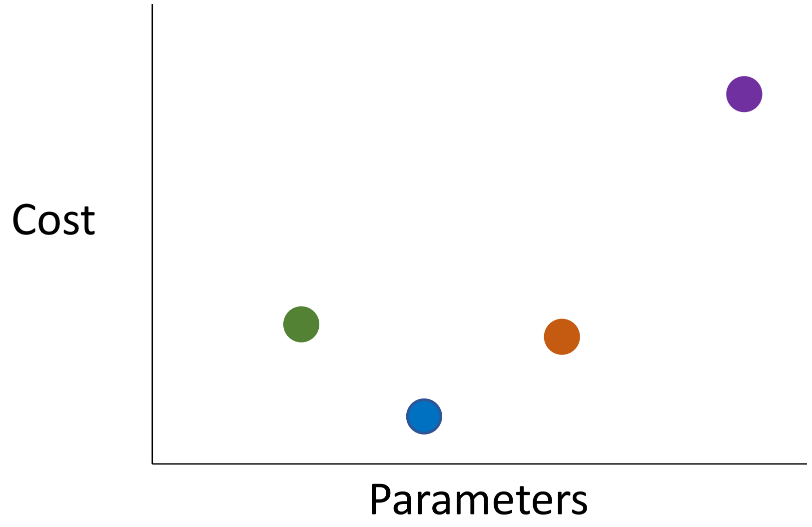
Hypothesis & Cost Function (linear Case)

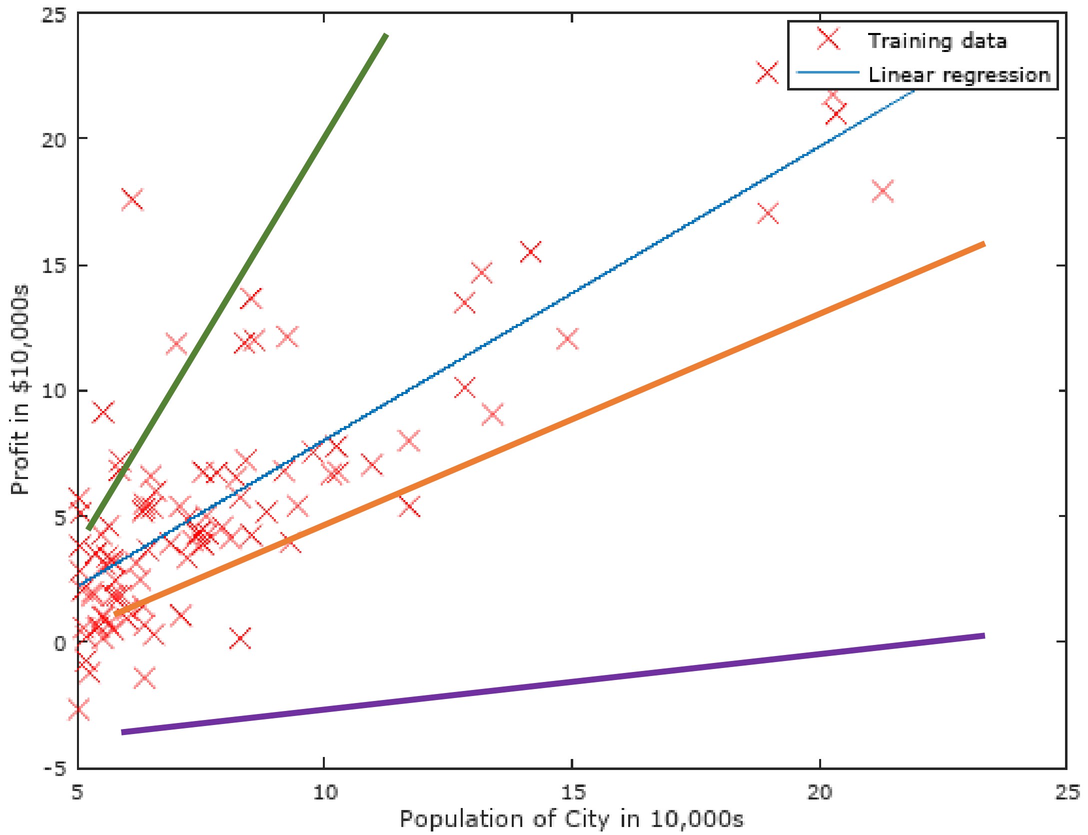
Hypothesis:

$$h_w(x) = w_0 + x w_1 = \hat{y}$$

Parameters:

$$w_0, w_1$$





Hypothesis & Cost Function (linear Case)

Hypothesis:

$$h_w(x) = w_0 + xw_1 = \hat{y}$$

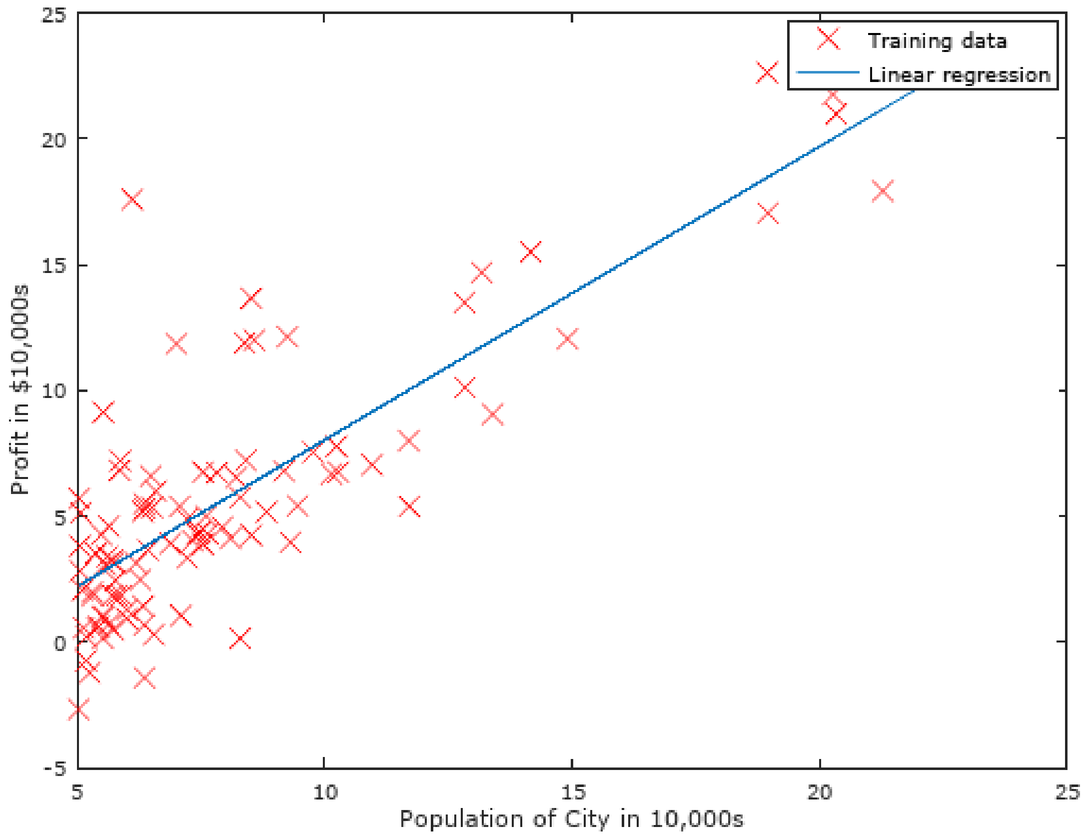
Parameters:

$$w_0, w_1$$

Cost function:

Mean Squared Error (MSE) →

$$C = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^i) - y^i)^2$$



Hypothesis & Cost Function (linear Case)

Hypothesis:

$$h_w(x) = w_0 + x w_1 = \hat{y}$$

Parameters:

$$w_0, w_1$$

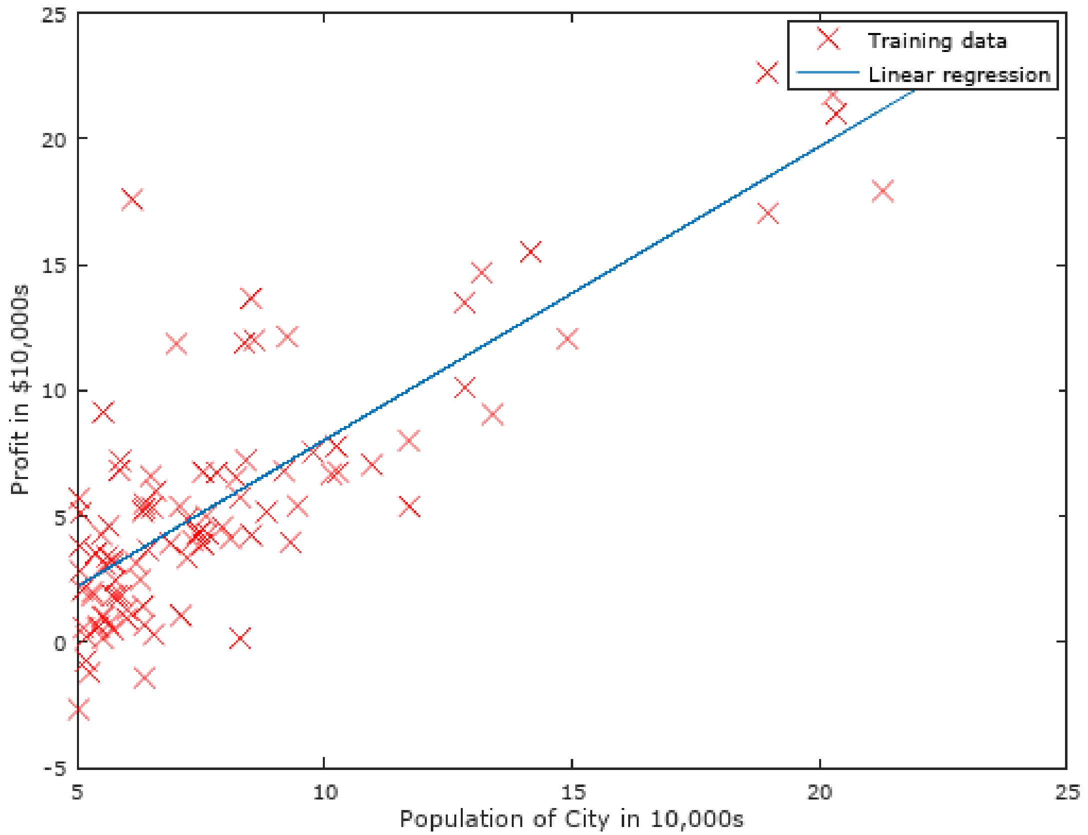
Cost function:

Mean Squared Error (MSE) $\rightarrow$

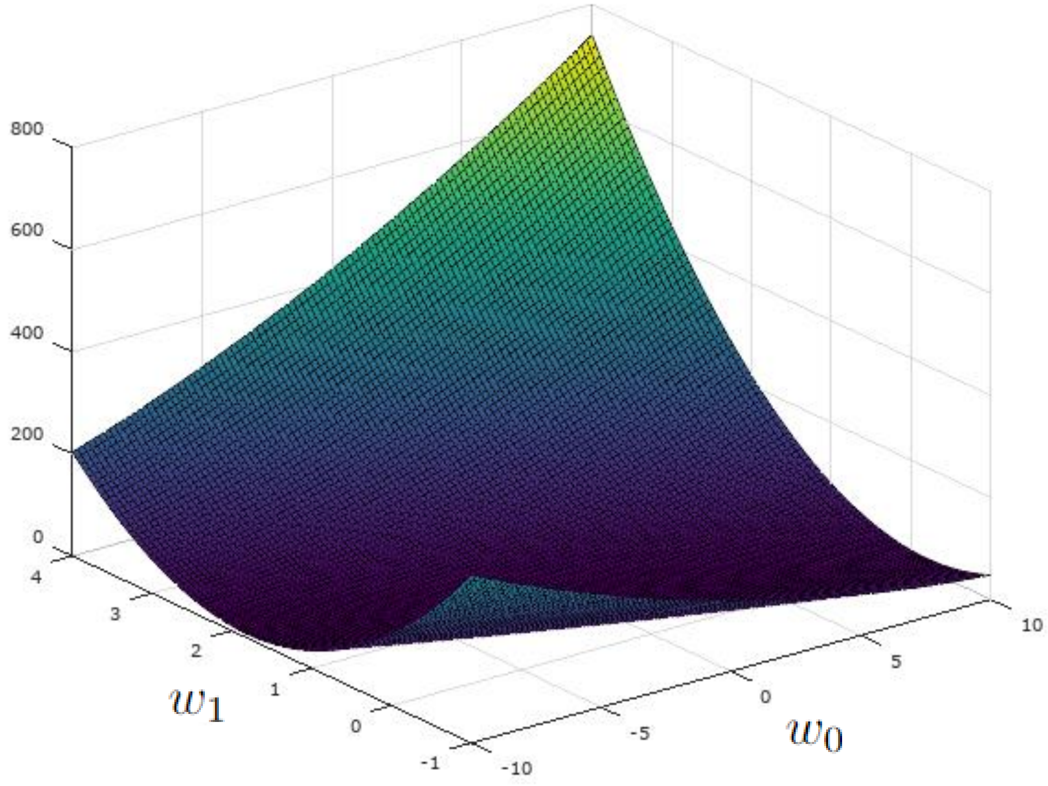
$$C = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^i) - y^i)^2$$

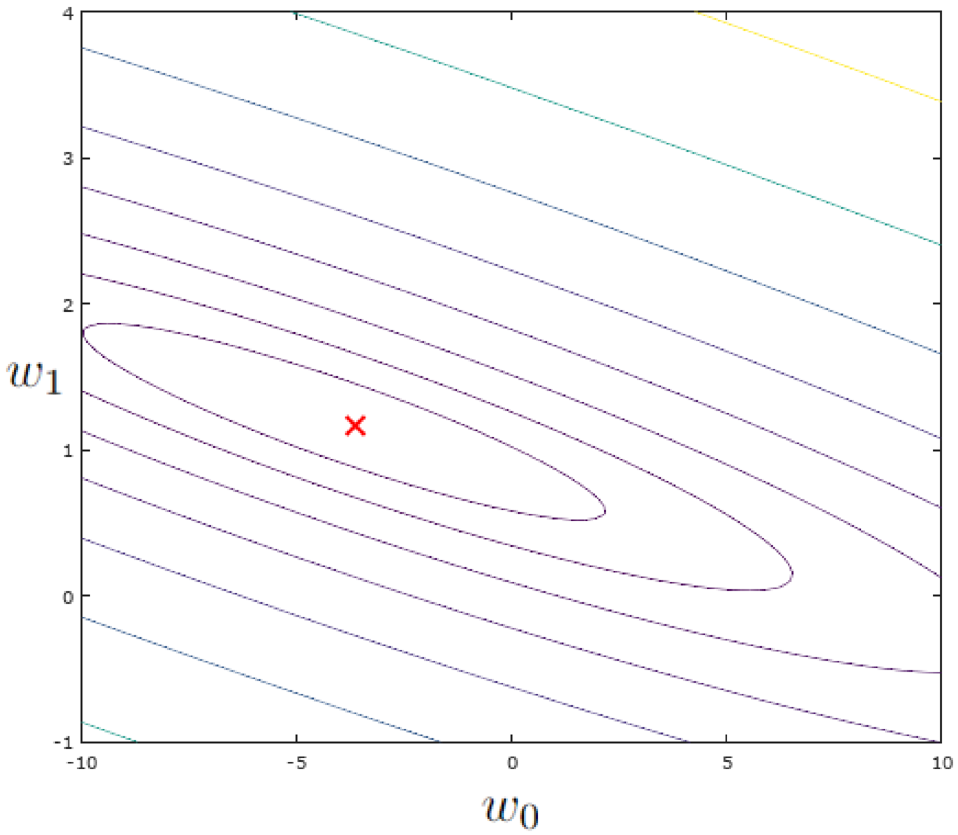
Goal: minimize

$$C(w_0, w_1)$$



Cost function

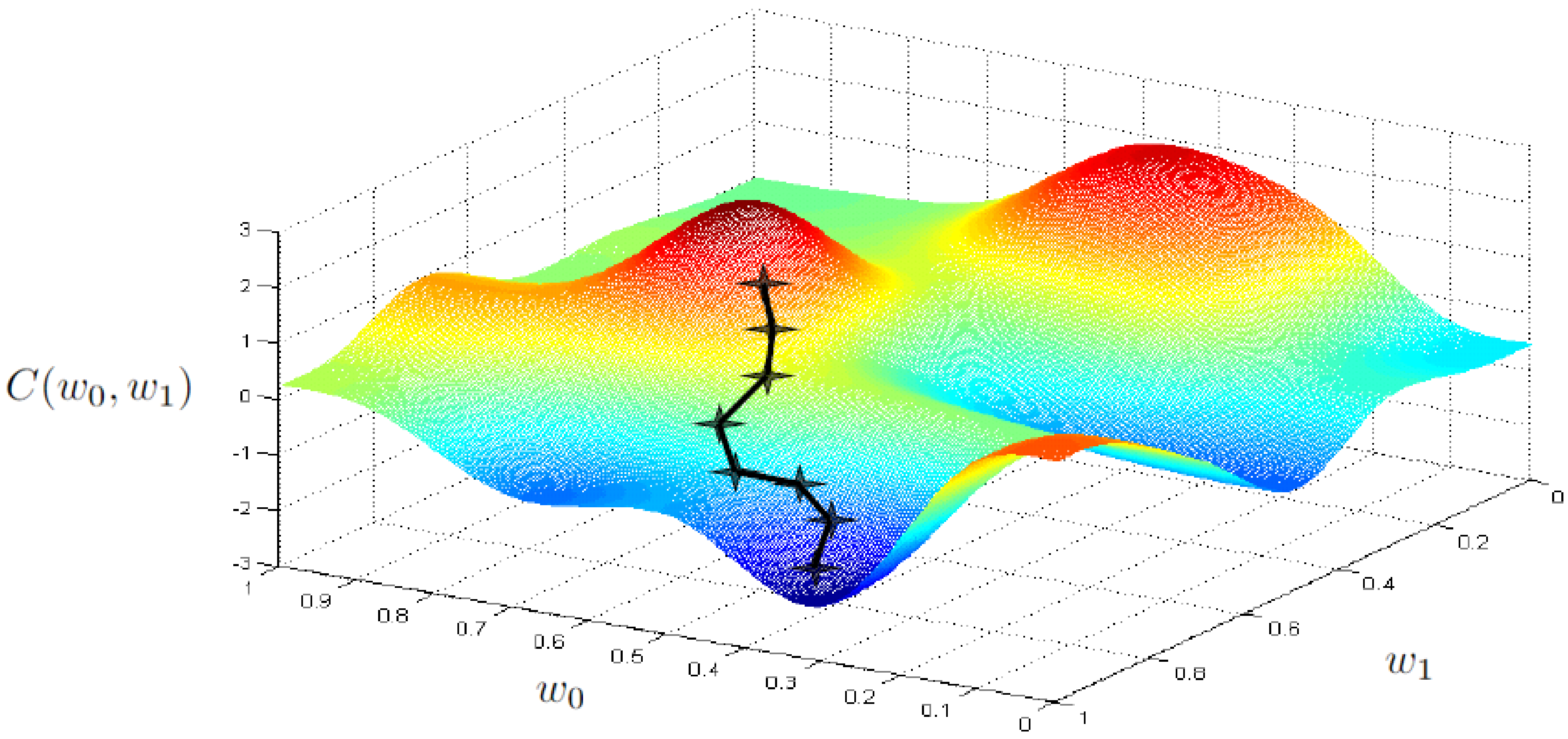




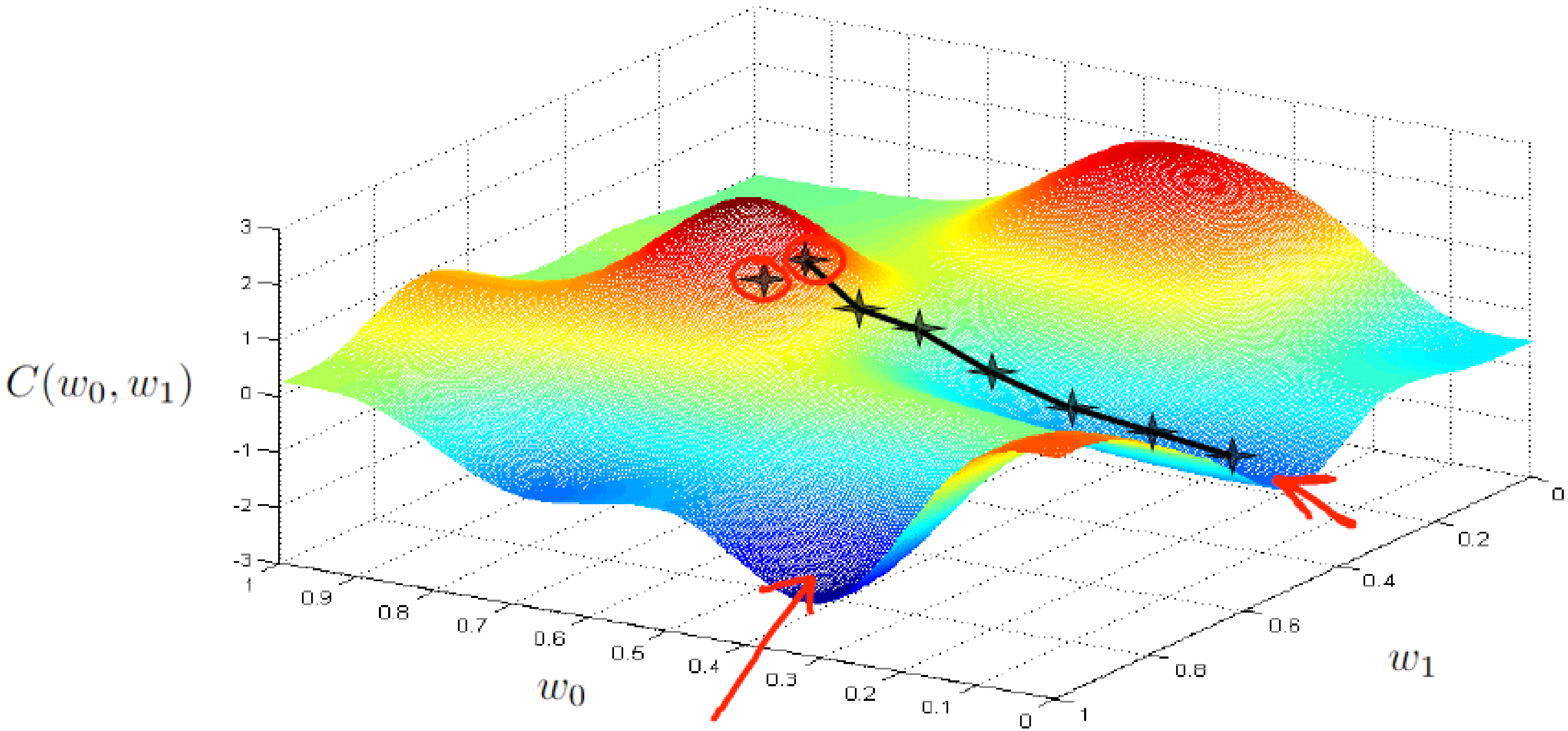
Gradient Descent method

- Have some cost function: C(w_0, w_1)
- Want to minimize cost function
- Outline:
  - Start with some w_0, w_1 (w_0=0, w_1=0)
  - Keep changing w_0, w_1 to reduce C(w_0, w_1) until we hopefully end up at a minimum

Gradient Descent visualization



Gradient Descent visualization



Gradient Descent algorithm

- Weight update of the Gradient Descent :
  (repeat until convergence)

$$w_j := w_j - \mu \frac{\partial}{\partial w_j} C(w_0, w_1)$$

- Simultaneous update!

Gradient Descent algorithm

- Weight update of the Gradient Descent :
  (repeat until convergence)

$$w_j := w_j - \mu \frac{\partial}{\partial w_j} C(w_0, w_1)$$

- Simultaneous update!

$$C = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^i) - y^i)^2$$

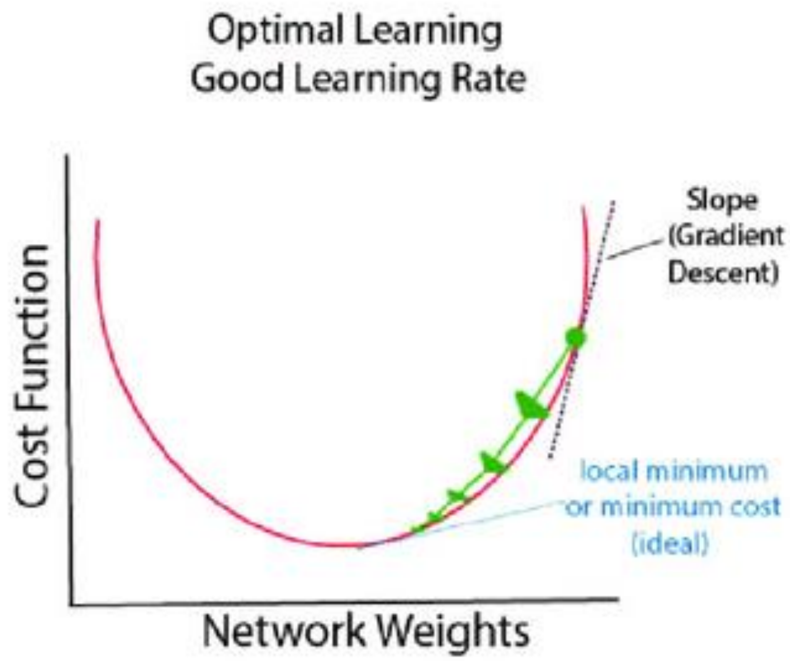
$$(j = 0) \quad \frac{\partial}{\partial w_j} C(w_0, w_1) = \frac{1}{m} \sum_{i=1}^{m} (w_0 + w_1 x^i - y^i) \cdot 1 = \frac{1}{m} \sum_{i=1}^{m} ((h_w(x^i) - y^i) \cdot x_0^i)$$

$$(j = 1) \quad \frac{\partial}{\partial w_j} C(w_0, w_1) = \frac{1}{m} \sum_{i=1}^{m} (w_0 + w_1 x^i - y^i) \cdot x_1^i = \frac{1}{m} \sum_{i=1}^{m} ((h_w(x^i) - y^i) \cdot x_1^i)$$

$$w_0 = w_0 - \frac{\mu}{m} \sum_{i=1}^{m} ((h_w(x^i) - y^i) \cdot x_0^i)$$
$$w_1 = w_1 - \frac{\mu}{m} \sum_{i=1}^{m} ((h_w(x^i) - y^i) \cdot x_1^i)$$

Learning rate

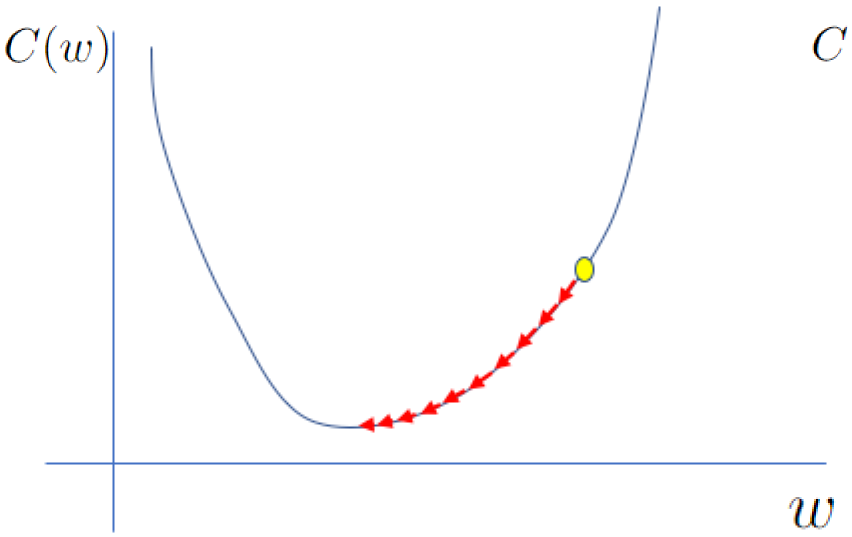
$$w_j := w_j - \boxed{\mu}\frac{\partial}{\partial w_j}C(w_0, w_1)$$



- 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.

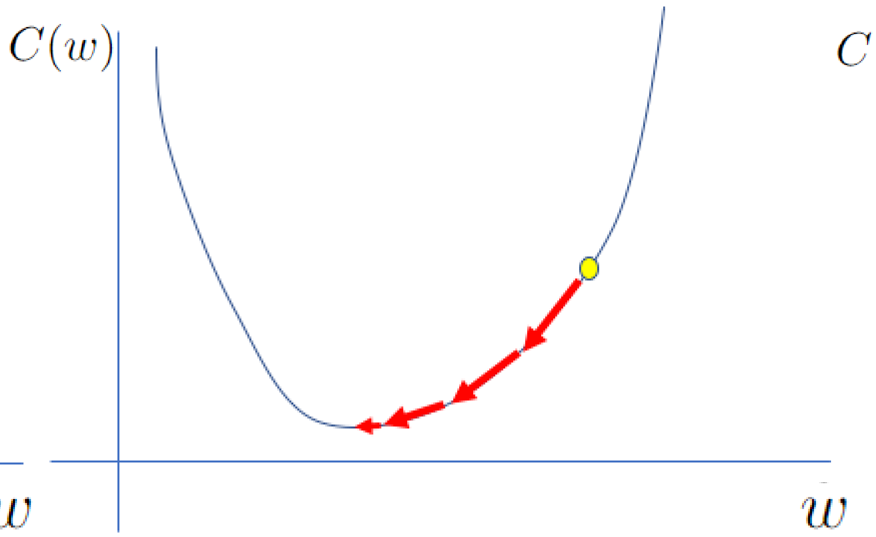
Learning rate

Too low



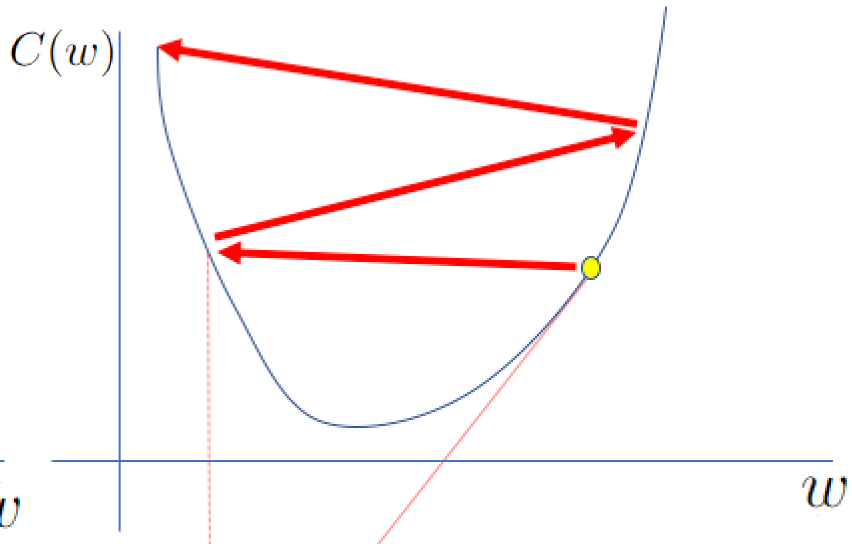
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

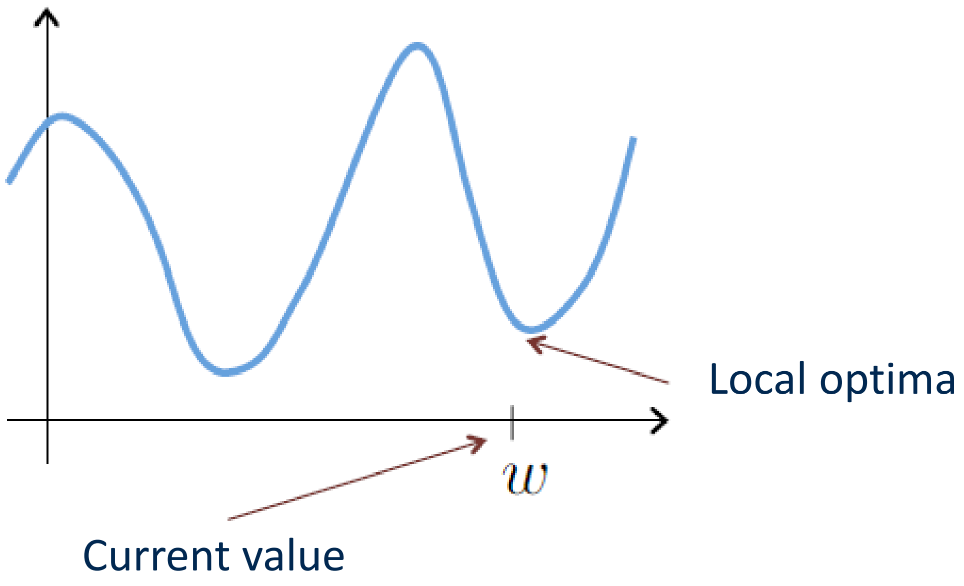
Too high



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 μ over time



Modell integration

Gradient descent algorithm

$$\text{repeat until convergence } \{$$
$$w_j := w_j - \mu \frac{\partial}{\partial w_j} C(w_0, w_1)$$
$$(\text{for } j = 1 \text{ and } j = 0)$$
$$\}$$

Linear Regression Model

$$h_w(x) = w_0 + x w_1$$

$$C = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^i) - y^i)^2$$

Modell integration

$$C(w) = \frac{1}{2m}\sum_{i=1}^{m}(h_w(x^i) - y^i)^2$$

The $h_w(x)$ is given by the linear model

$$h_w(x) = w^T x = w_0 + x w_1$$

$$w_0 := w_0 - \frac{\mu}{m}\sum_{i=1}^{m}((h_w(x^i) - y^i)$$

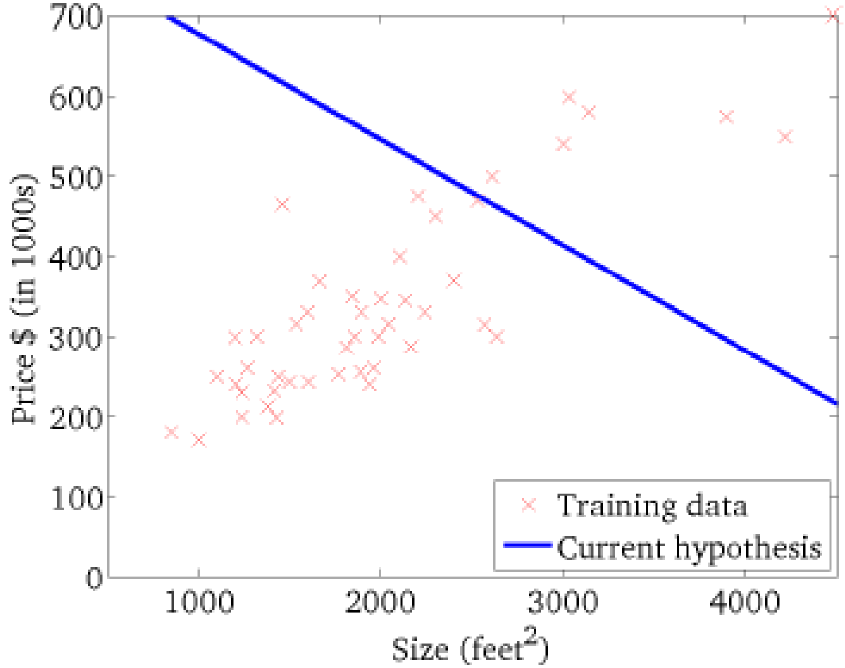
$$w_1 := w_1 - \frac{\mu}{m}\sum_{i=1}^{m}((h_w(x^i) - y^i) \cdot x_1^i)$$

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

Example

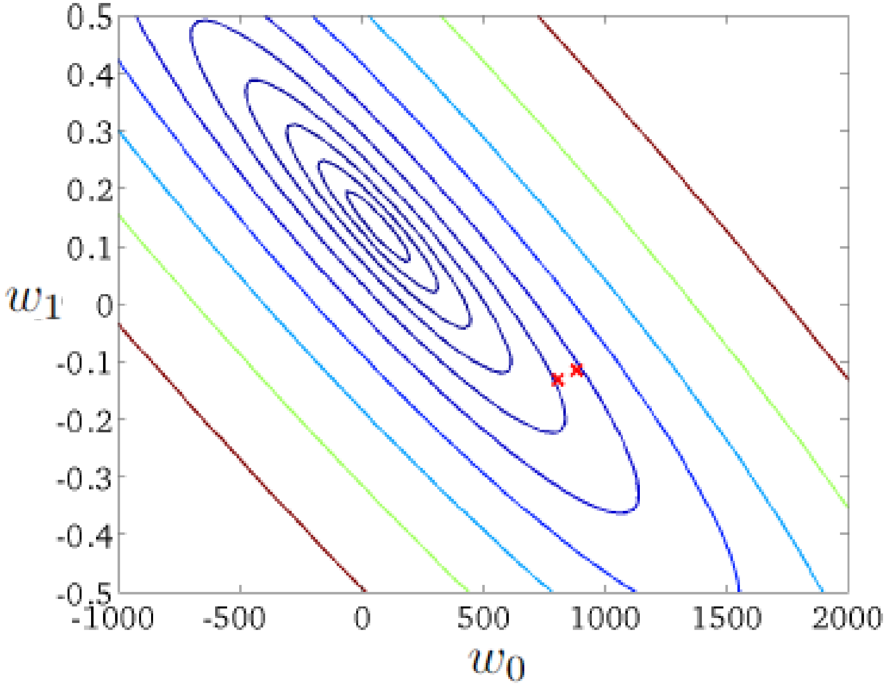
$h_w(x)$

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



$C(w_0, w_1)$

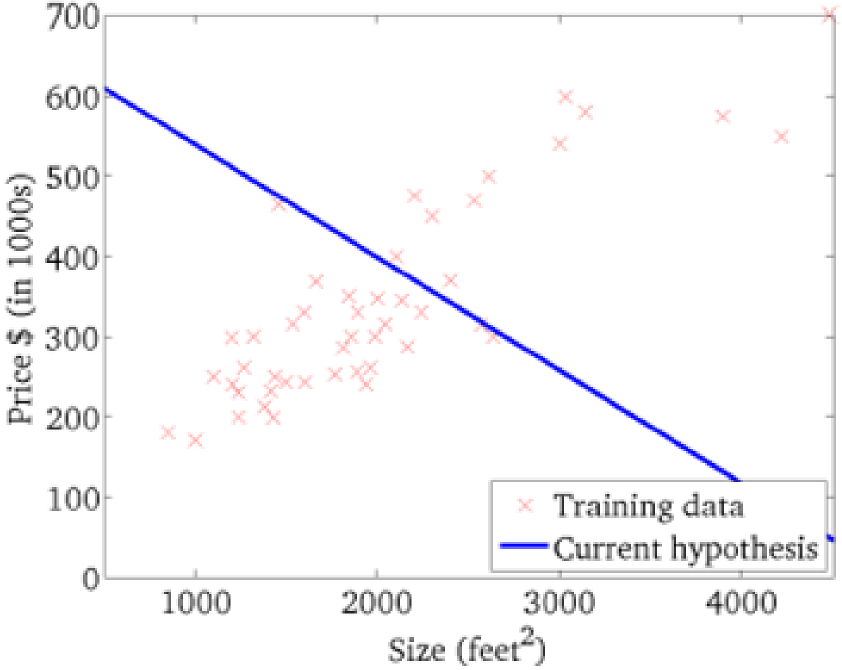
(function of the parameters θ_0, θ_1)



Example

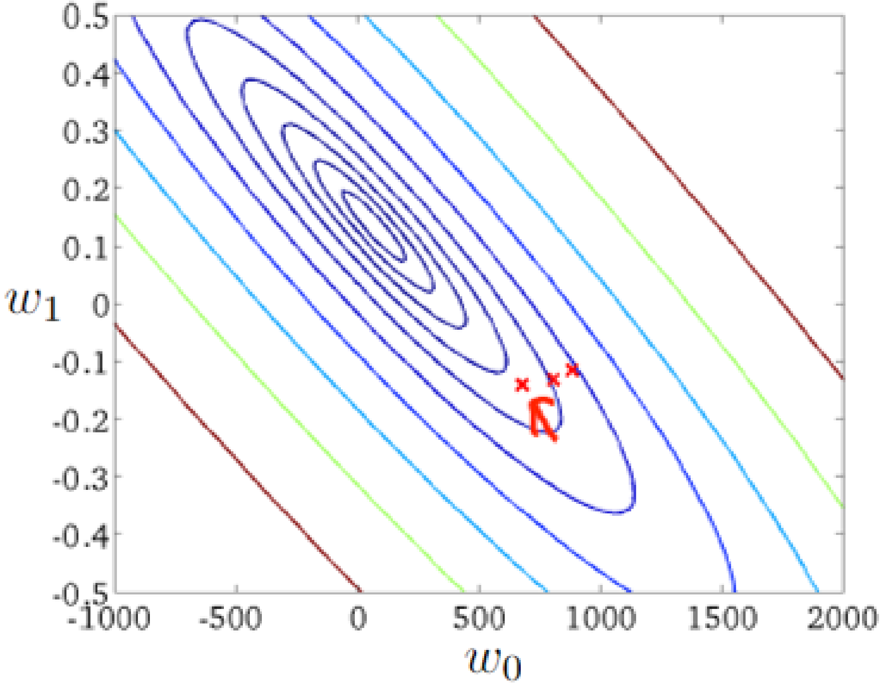
$h_w(x)$

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



$C(w_0, w_1)$

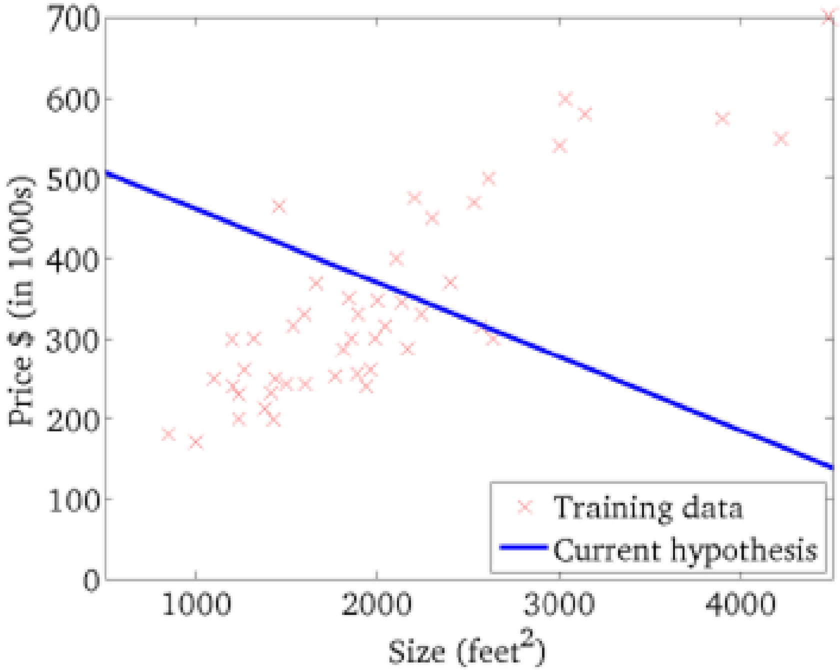
(function of the parameters θ_0, θ_1)



Example

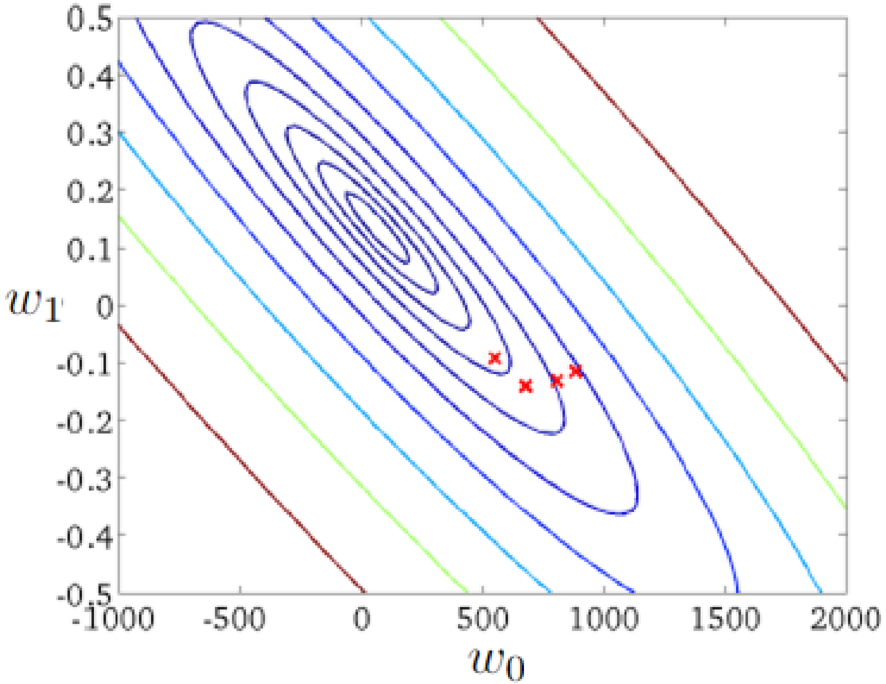
$h_w(x)$

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



$C(w_0, w_1)$

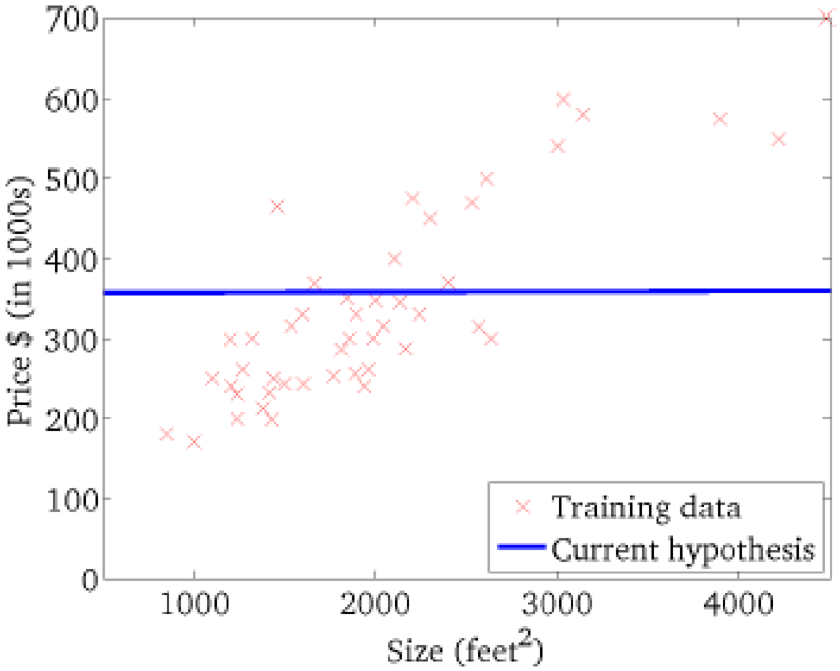
(function of the parameters θ_0, θ_1)



Example

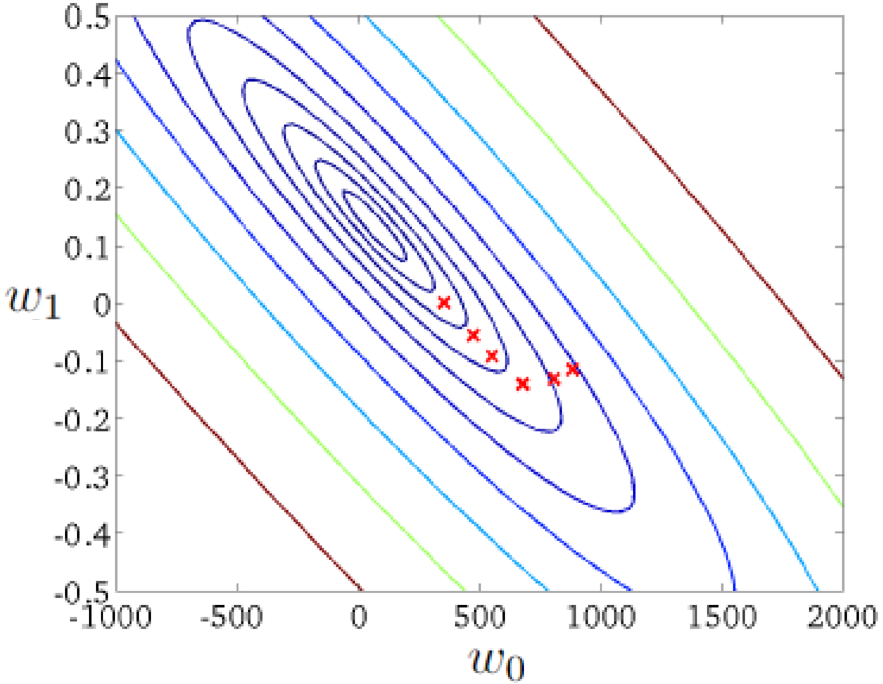
$h_w(x)$

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



$C(w_0, w_1)$

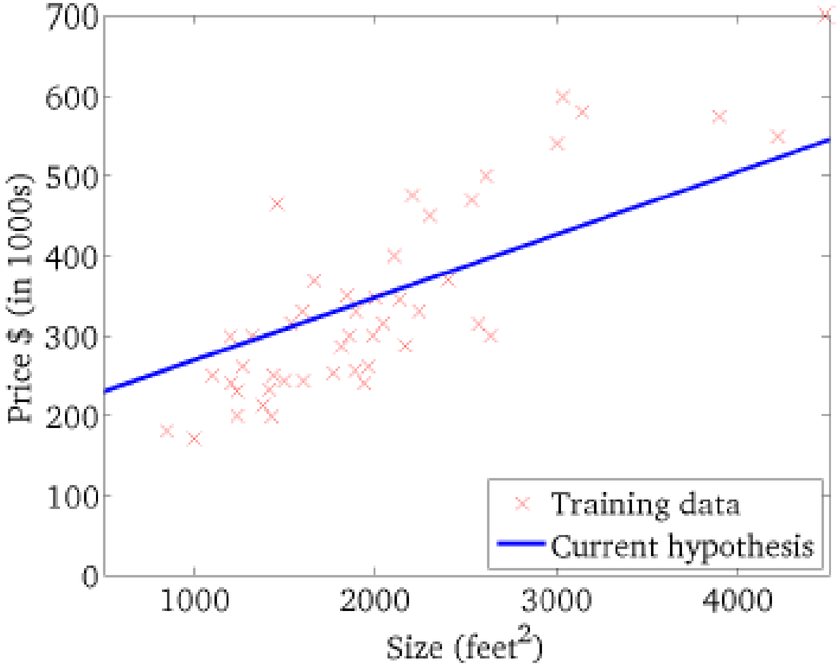
(function of the parameters θ_0, θ_1)



Example

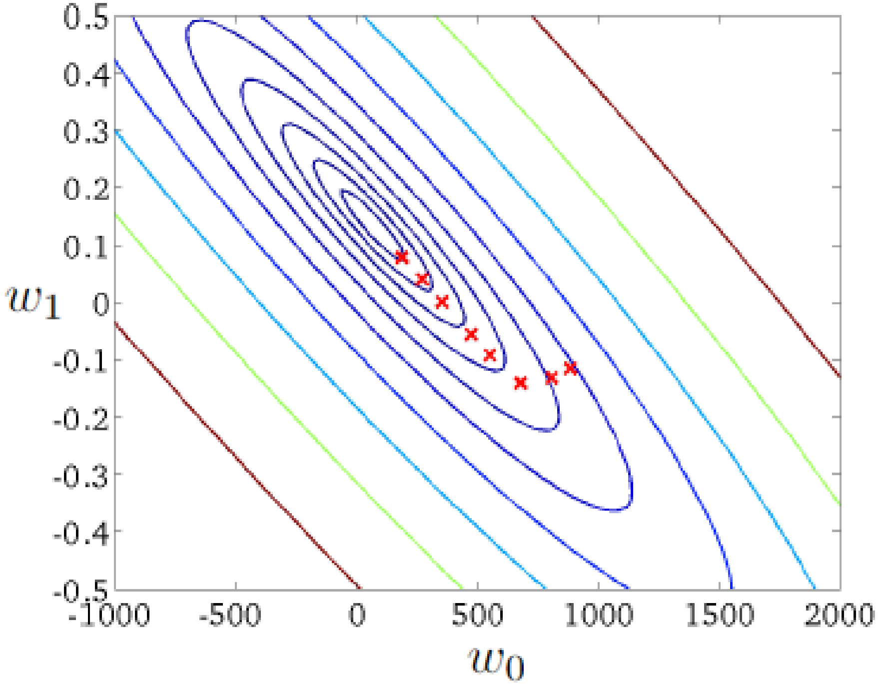
$$h_w(x)$$

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



$$C(w_0, w_1)$$

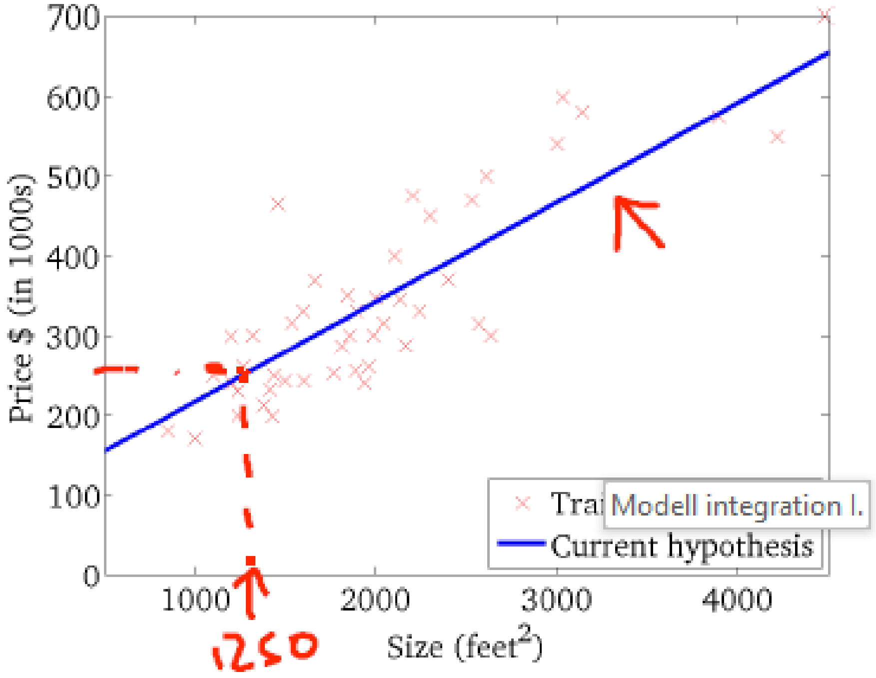
(function of the parameters θ_0, θ_1)



Example

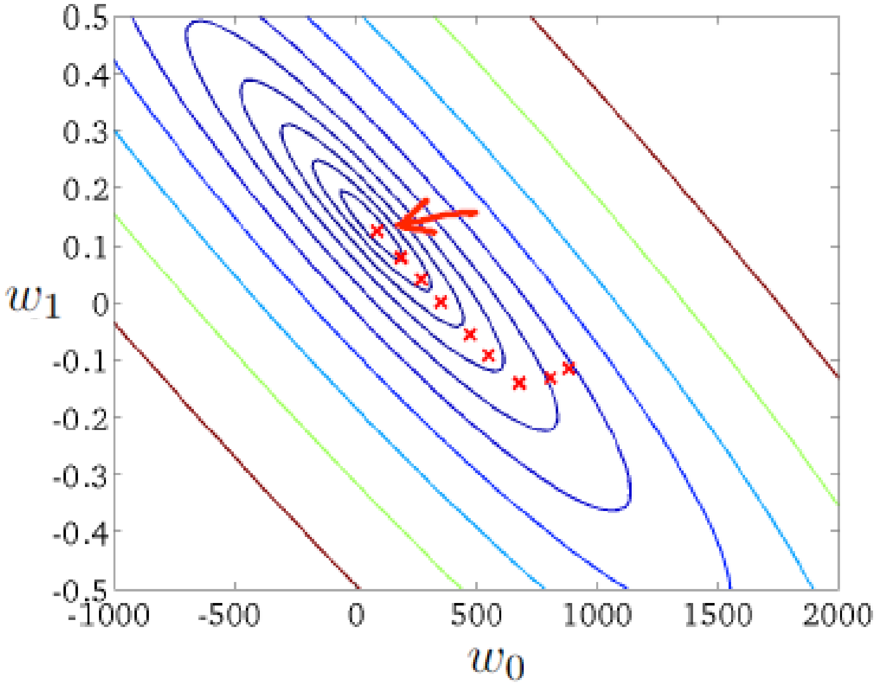
$$h_w(x)$$

(for fixed w_0, w_1, this is a function of x)



$$C(w_0, w_1)$$

(function of the parameters w_0, w_1)



Linear regression with multiple features

One variable

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

$$h_w(x) = w_0 + xw_1$$

Linear regression with multiple features

One variable

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

$$h_w(x) = w_0 + xw_1$$

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
…	…	…	…	…

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

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

$$h_w(x) = w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

$$\underset{m\times(n+1)}{X} = \begin{bmatrix} x_0^1 & x_1^1 & x_2^1 & \ldots & x_n^1 \\ x_0^2 & x_1^2 & x_2^2 & \ldots & x_n^2 \\ x_0^3 & x_1^3 & x_2^3 & \ldots & x_n^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^m & x_1^m & x_2^m & \ldots & x_n^m \end{bmatrix}, \underset{(n+1)\times 1}{W} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}, \underset{m\times 1}{Y} = \begin{bmatrix} y^1 \\ y^2 \\ y^3 \\ \vdots \\ y^m \end{bmatrix}$$

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

- This allows us to do matrix operations

$$h_w(x) = W^T x$$

$$C(w) = \frac{1}{2m}(XW - Y)^T(XW - Y)$$

Matrix operations hints

$$\underset{(n+1)\times 1}{W} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

$$\underset{m\times(n+1)}{X} = \begin{bmatrix} x_0^1 & x_1^1 & x_2^1 & \dots & x_n^1 \\ x_0^2 & x_1^2 & x_2^2 & \dots & x_n^2 \\ x_0^3 & x_1^3 & x_2^3 & \dots & x_n^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^m & x_1^m & x_2^m & \dots & x_n^m \end{bmatrix} \quad \begin{bmatrix} w_0x_0^1 + w_1x_1^1 + w_2x_2^1 + \dots + w_nx_n^1 \\ w_0x_0^2 + w_1x_1^2 + w_2x_2^2 + \dots + w_nx_n^2 \\ w_0x_0^3 + w_1x_1^3 + w_2x_2^3 + \dots + w_nx_n^3 \\ \vdots \\ w_0x_0^m + w_1x_1^m + w_2x_2^m + \dots + w_nx_n^m \end{bmatrix} \quad - \quad \underset{m\times 1}{Y} = \begin{bmatrix} y^1 \\ y^2 \\ y^3 \\ \vdots \\ y^m \end{bmatrix}$$

$$v = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

$$v' = \begin{bmatrix} v_0 & v_1 & v_2 & \dots & v_n \end{bmatrix} \quad \left[(v_0)^2 + (v_1)^2 + (v_2)^2 + \dots + (v_n)^2\right] \quad \rightarrow v'v = sum(v.\hat{}2)$$

Modifying Gradient Descent algorithm

New algorithm ($n \geq 1$)
Repeat {

$$w_j := w_j - \frac{\mu}{m}\sum_{i=1}^{m}((h_w(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)})$$

Simultaneously update w_j for $j=0,\ldots,n$

}

$$w_0 := w_0 - \mu\frac{1}{m}\sum_{i=1}^{m}((h_w(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)})$$

$$w_1 := w_1 - \mu\frac{1}{m}\sum_{i=1}^{m}((h_w(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)})$$

$$w_2 := w_2 - \mu\frac{1}{m}\sum_{i=1}^{m}((h_w(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)})$$

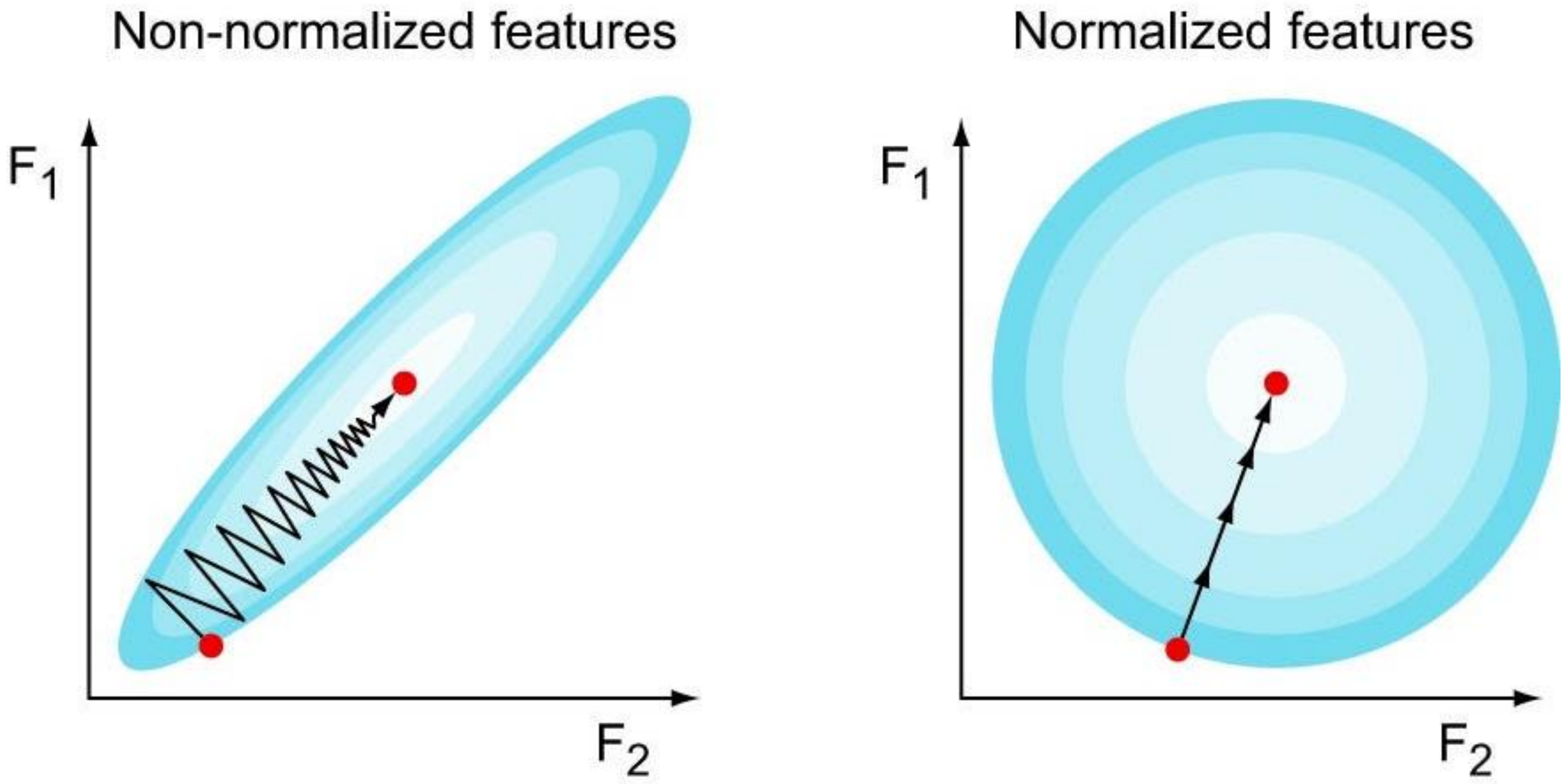
Feature Scaling

- 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**

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

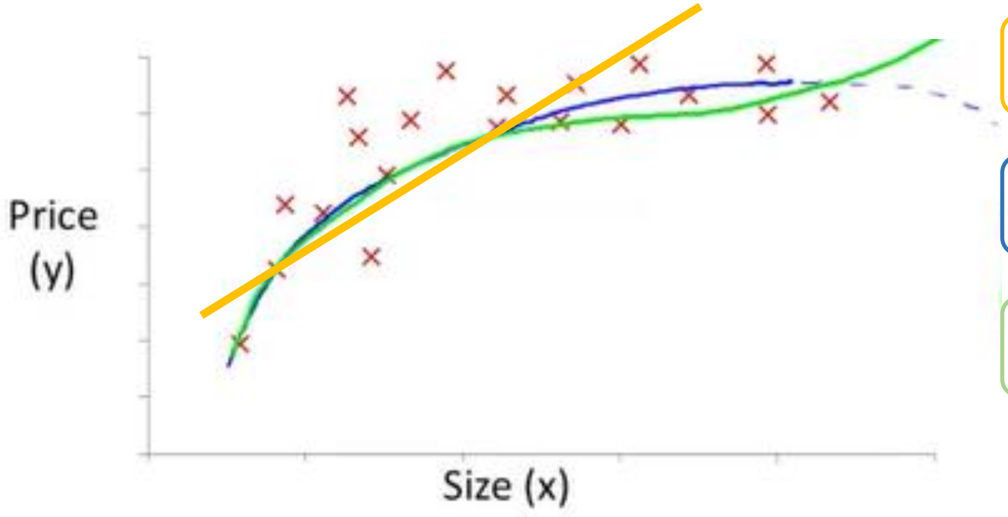
Where μ_i is the **average** of all the values for feature (i) and s_i is the range of values (max - min), or s_i is the standard deviation.

Feature Scaling



Polynomial regression

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



$$w_0 + w_1 x_1$$

$$w_0 + w_1 x_1 + w_2 x_2$$

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

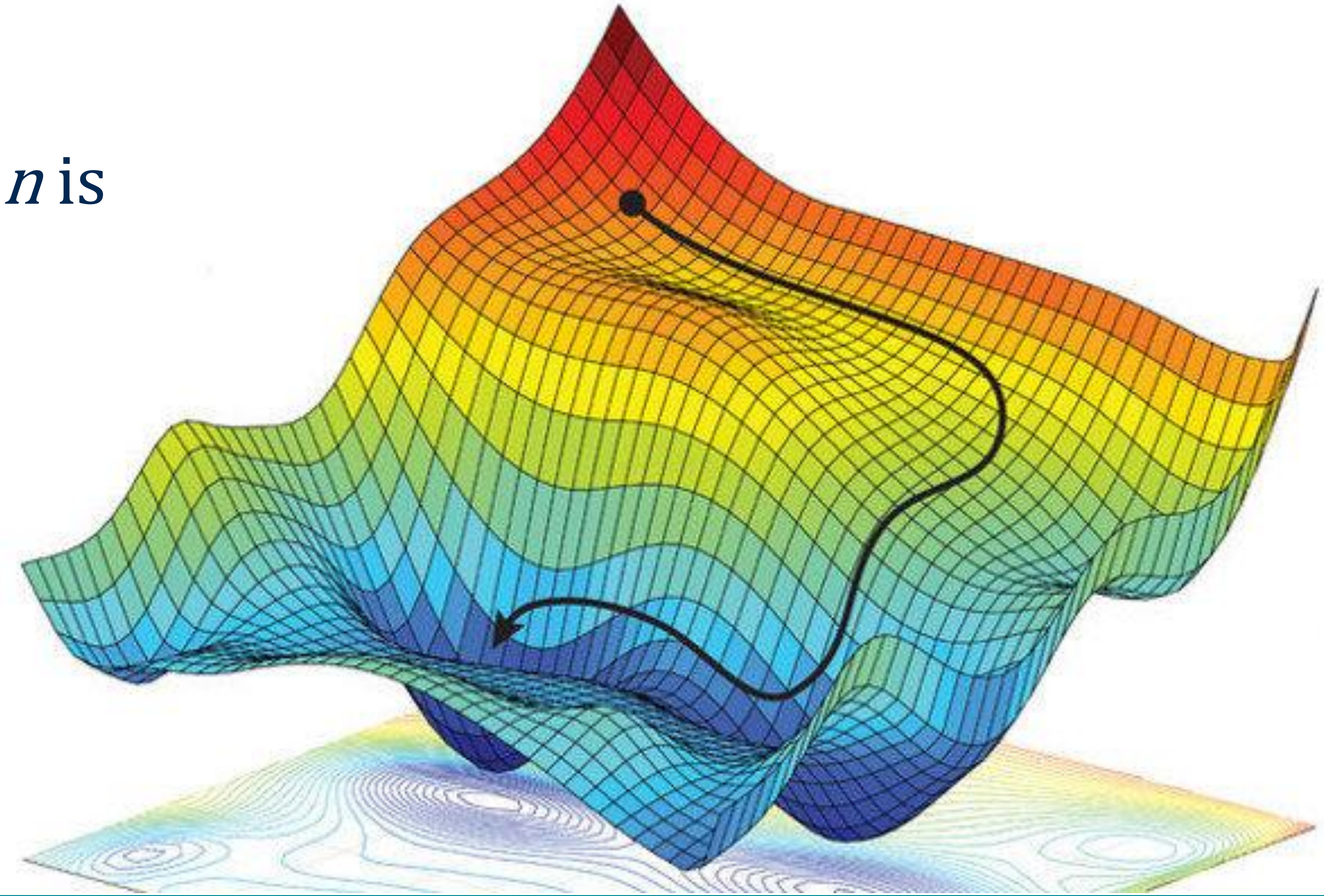
$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 =$$

$$w_0 + w_1(size) + w_2(size)^2 + w_3(size)^3$$

$$\boxed{\begin{aligned} x_1 &= (size) \\ x_2 &= (size)^2 \\ x_3 &= (size)^3 \end{aligned}}$$

Gradient Descent Overview

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



ELTE | FACULTY OF INFORMATICS

Thank you for your attention!