Politecnico di Bari

Supervised Learning

Politecnico di Bari

Linear regression with one variable

Model Representation

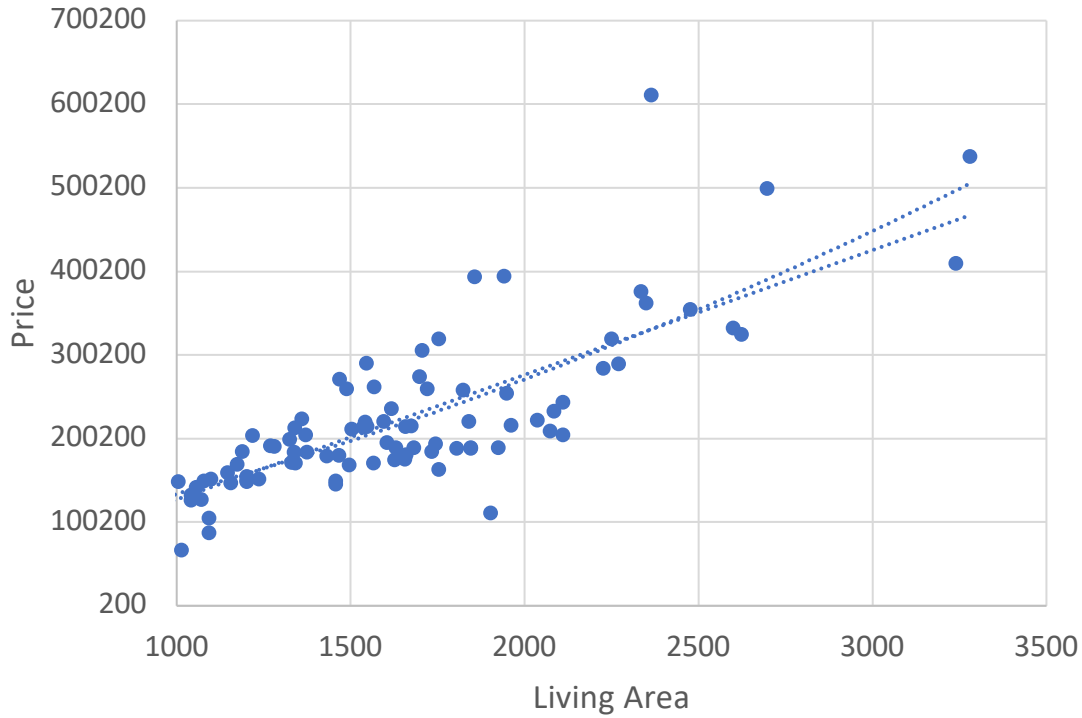
Housing prediction problem in Ames

Which kind of data we have?

- Actual answers from real data examples

What we want to predict?

- A real valued output



Housing prediction problem in Ames

Training set of housing prices

Living area (feet2)	Price (1000$s)
1656	215
896	105
1329	172
2110	244
…	…

Notation:

m = Number of training examples

x = "input" variable / feature

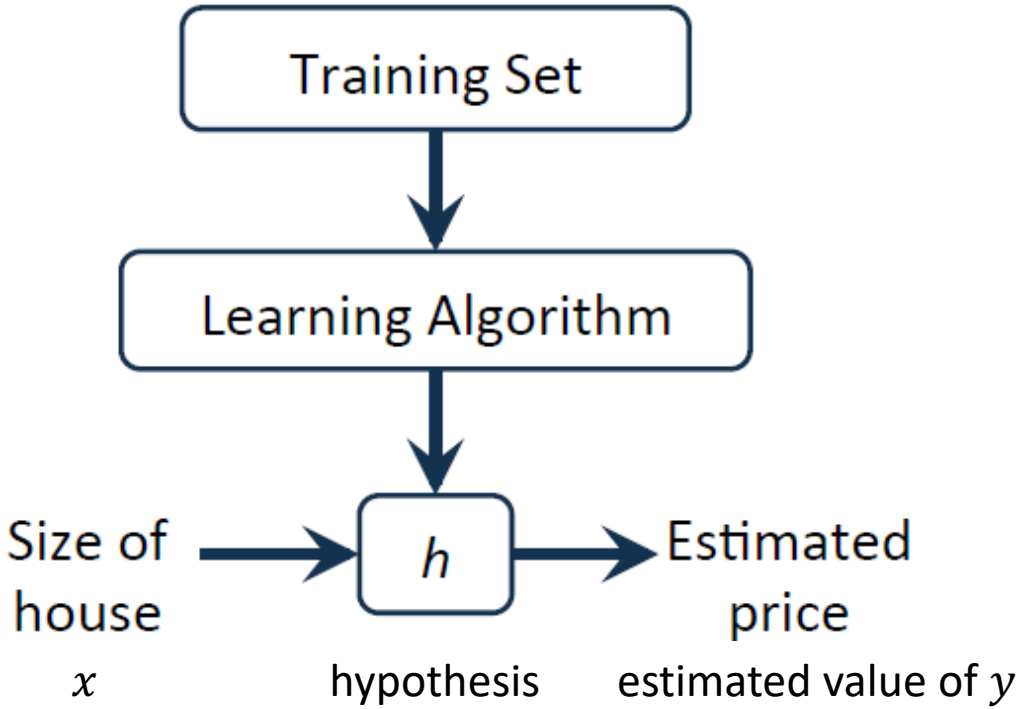
y = "output" variable / "target" variable

(x, y) = tuple representing one training example

$(x^{(i)}, y^{(i)})$ = i^{th} training example

$x^{(1)}$	1656
$x^{(2)}$	896
$y^{(1)}$	215

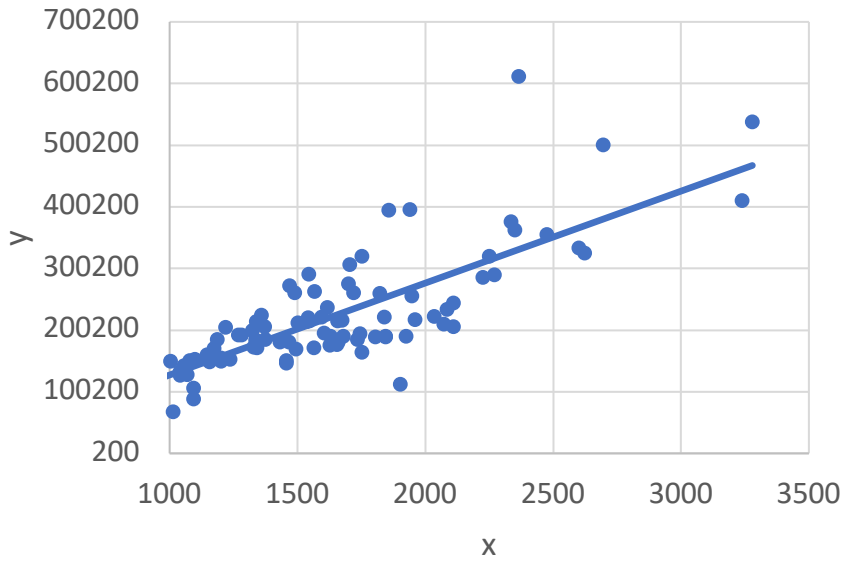
What we are going to realize



We want to realize a function that, given a value of x, it will return the estimated value of y.

How *h* could be?

$h_\theta(x) = \theta_0 + \theta_1 x$ Usually abbreviated as $h(x)$



Linear regression with one variable
(Univariate linear regression)

Politecnico di Bari

Linear regression with one variable

Cost function

Linear hypothesis

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

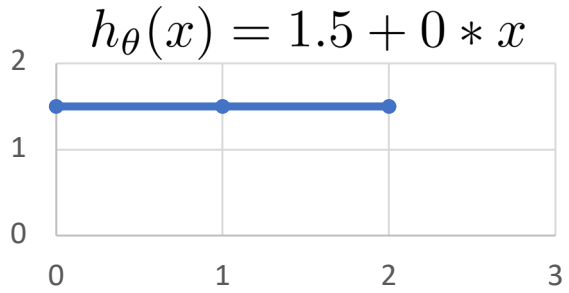
θ_i : parameters/weights

How θ_i can be chosen?

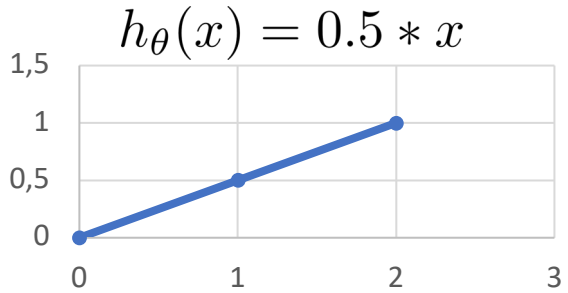
Living area ($feet^2$)	Price (1000$)
1656	215
896	105
1329	172
2110	244
…	…

Univariate linear regression

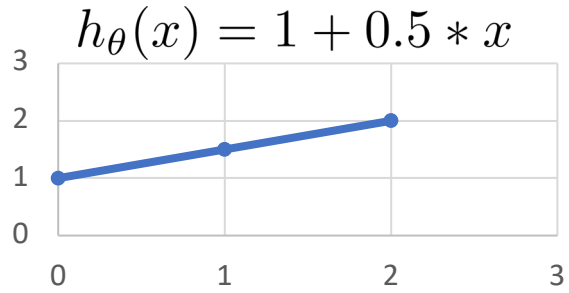
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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

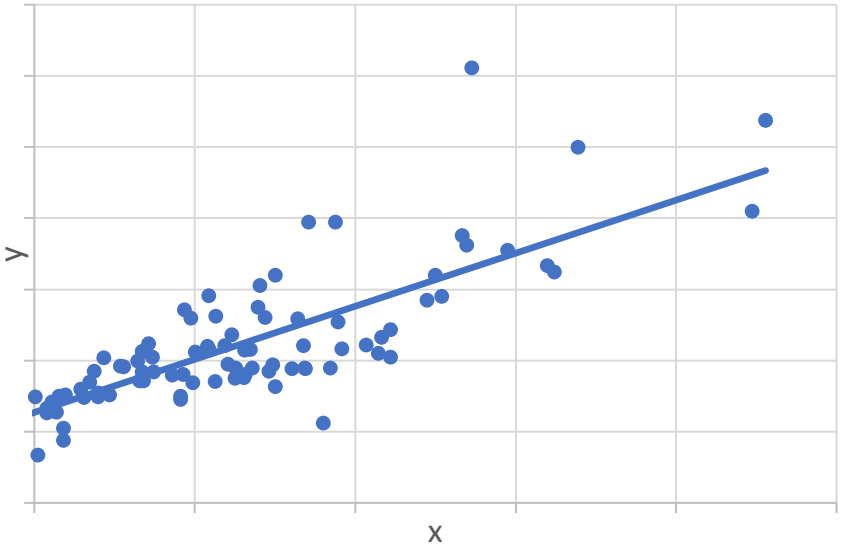


$$\theta_0 = 0$$
$$\theta_1 = 0.5$$



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

Cost Function: Intuition



We can choose the **parameters**

so the **hypothesis is** as **close** as possible

to y for our **training examples**

#training examples

Squared error function

$$\underset{\theta_0,\theta_1}{argmin}\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(\mathbf{x^{(i)}})-y^{(i)})^2$$

$$h_\theta(\mathbf{x^{(i)}})=\theta_0+\theta_1 x^{(i)}$$

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

$$\theta_{min}=argmin_\theta(J(\theta_0,\theta_1))=argmin_\theta(J(\theta)).$$

Politecnico di Bari

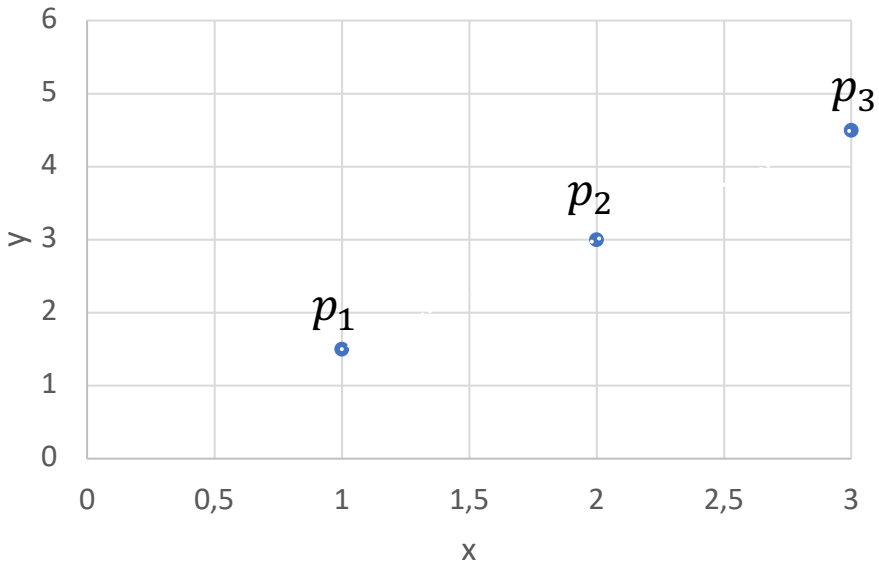
Linear regression with one variable

Cost function Intuition I

Cost Function Example

$p_1 = (1, 1.5)$
$p_2 = (2, 3)$
$p_3 = (3, 4.5)$

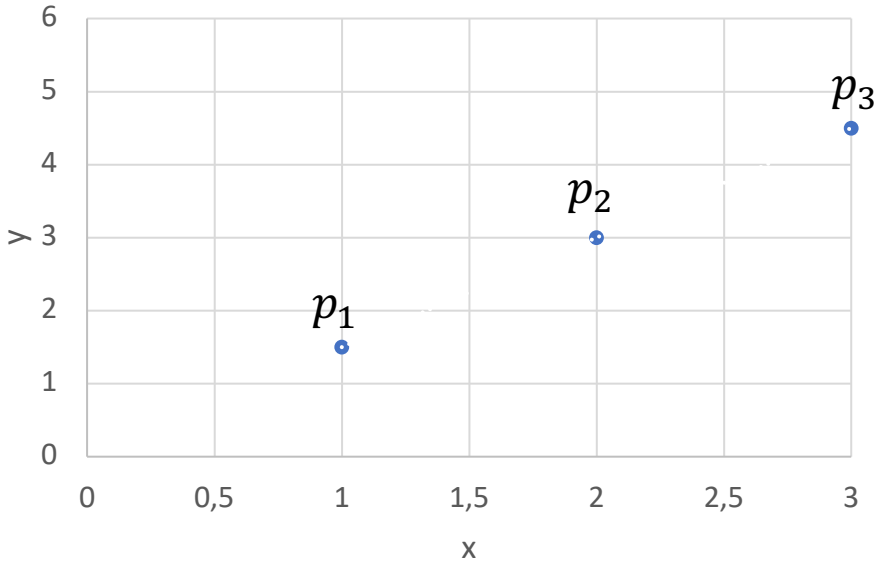
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$



$$J(\theta) = \frac{1}{2 \cdot 3} ((\theta_0 + 1 \cdot \theta_1 - 1.5)^2 + (\theta_0 + 2 \cdot \theta_1 - 3)^2 + (\theta_0 + 3 \cdot \theta_1 - 4.5)^2)$$

Cost Function Example

$p_1 = (1, 1.5)$
$p_2 = (2, 3)$
$p_3 = (3, 4.5)$



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

$$J(\theta) = \frac{1}{2 \cdot 3} \overbrace{((\theta_0 + 1 \cdot \theta_1 - 1.5)^2}^{h_\theta(x^{(1)})} + (\theta_0 + 2 \cdot \theta_1 - 3)^2 + (\theta_0 + 3 \cdot \theta_1 - 4.5)^2)$$

m $\quad$ $x^{(1)}$ $\quad$ $y^{(1)}$

Univariate Linear regression

Hypothesis:

$$h_\theta(\mathbf{x^{(i)}}) = \theta_0 + \theta_1 x^{(i)}$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

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

Goal:

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

Simplified toy example

Hypothesis:

$$h_\theta(\mathbf{x^{(i)}}) = \theta_1 x^{(i)} \qquad (\theta_0 = 0)$$

Parameters:

$$\theta_1$$

Cost Function:

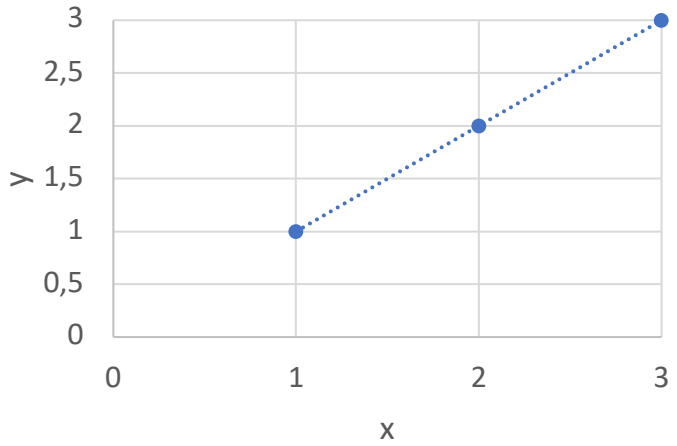
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)})^2$$

Goal:

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

$h_\theta(x)$

(function of x, with θ_1 fixed)

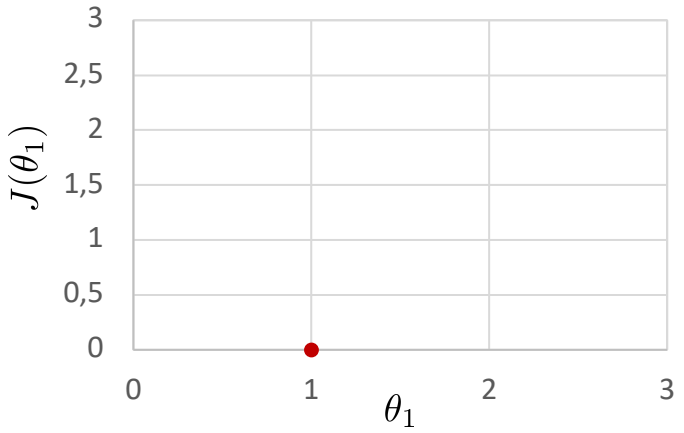


$$\theta_1 = 1$$

$$h_\theta(x^{(i)}) = y^{(i)}$$

$J(\theta_1)$

(function of θ_1, an aggregate function over all x's)



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

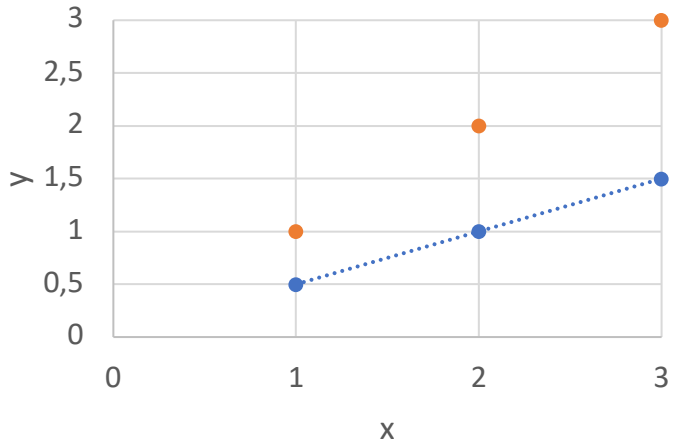
$$= \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2*3}\left[0^2 + 0^2 + 0^2\right]$$

$$J(1) = 0$$

$h_\theta(x)$

(function of x, with θ_1 fixed)

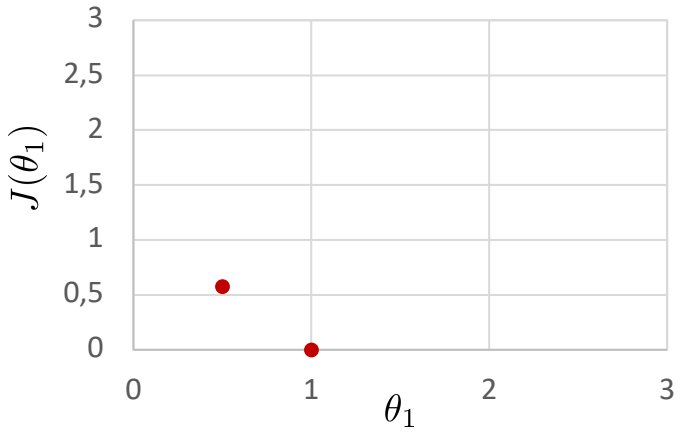


$$\theta_1 = 0.5$$

$$h_\theta(x^{(i)}) = y^{(i)}$$

$J(\theta_1)$

(function of θ_1, an aggregate function over all x's)



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

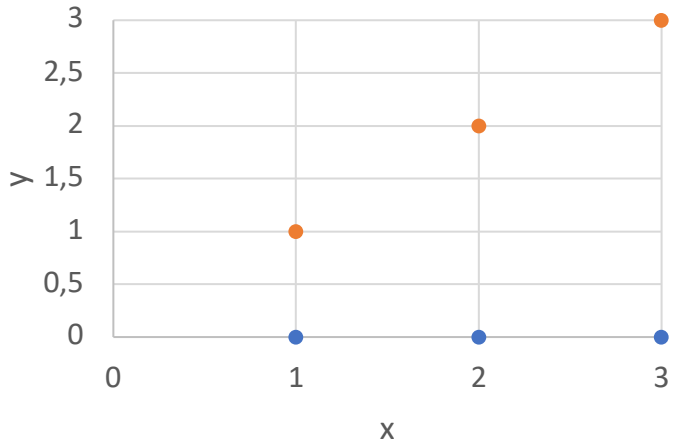
$$= \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2*3}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

$$J(0.5) = \frac{3.5}{6} = 0.58$$

$h_\theta(x)$

(function of x, with θ_1 fixed)

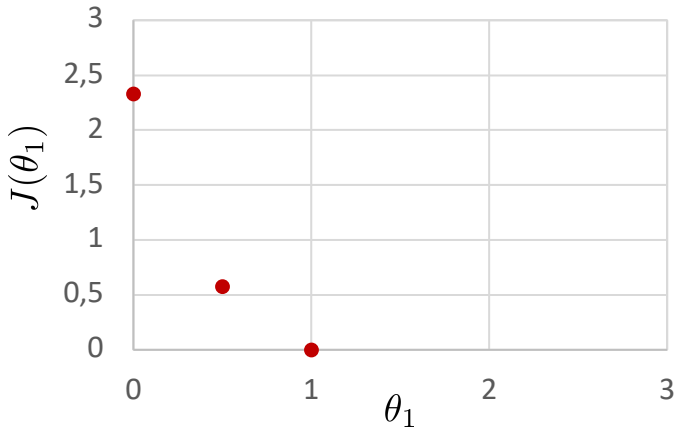


$$\theta_1 = 0$$

$$h_\theta(x^{(i)}) = y^{(i)}$$

$J(\theta_1)$

(function of θ_1, an aggregate function over all x's)



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

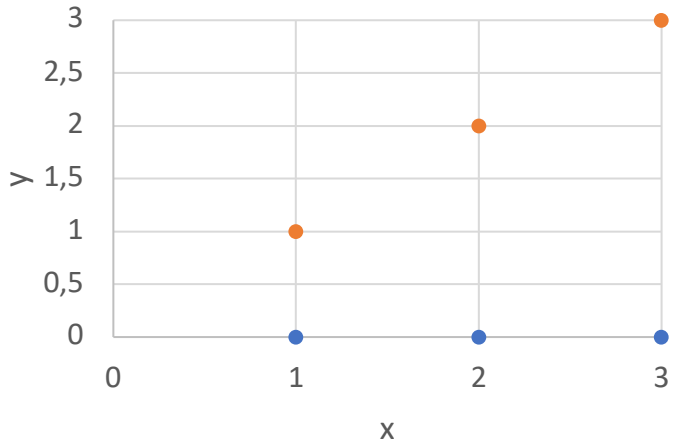
$$= \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2*3}\left[(0-1)^2 + (0-2)^2 + (0-3)^2\right]$$

$$J(0) = \frac{14}{6} = 2.33$$

$h_\theta(x)$

(function of x, with θ_1 fixed)

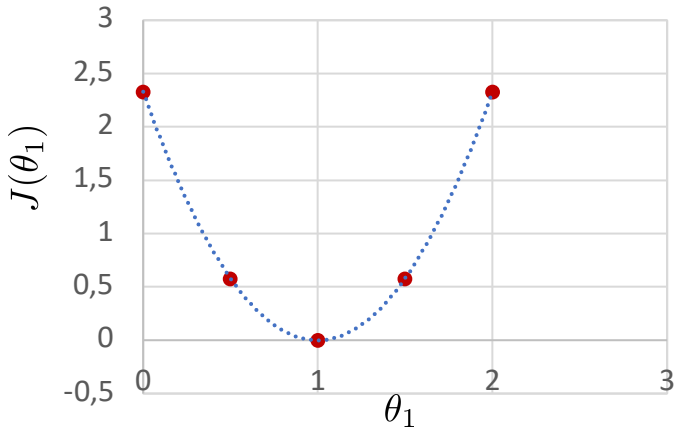


$$\theta_1 = 0$$

$$h_\theta(x^{(i)}) = y^{(i)}$$

$J(\theta_1)$

(function of θ_1, an aggregate function over all x's)



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

Politecnico di Bari

Linear regression with one variable

Cost function Intuition II

Recap

Univariate Linear regression

Hypothesis:

$$h_\theta(\mathbf{x^{(i)}}) = \theta_0 + \theta_1 x^{(i)}$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

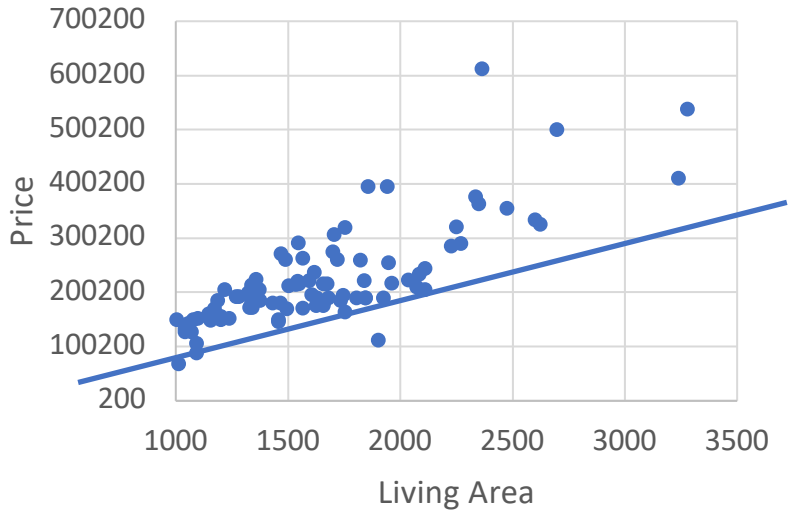
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)})^2$$

Goal:

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

$h_\theta(x)$

(function of x, with θ_1 fixed)



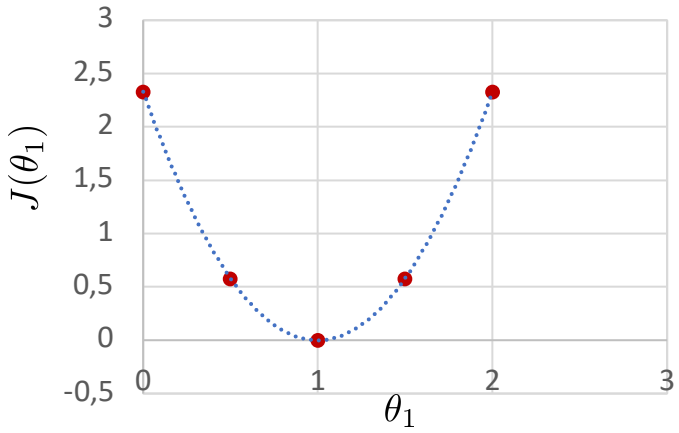
$$\theta_0 = 100000$$

$$\theta_1 = 66$$

$$h_\theta(x^{(i)}) = 100000 + 66x$$

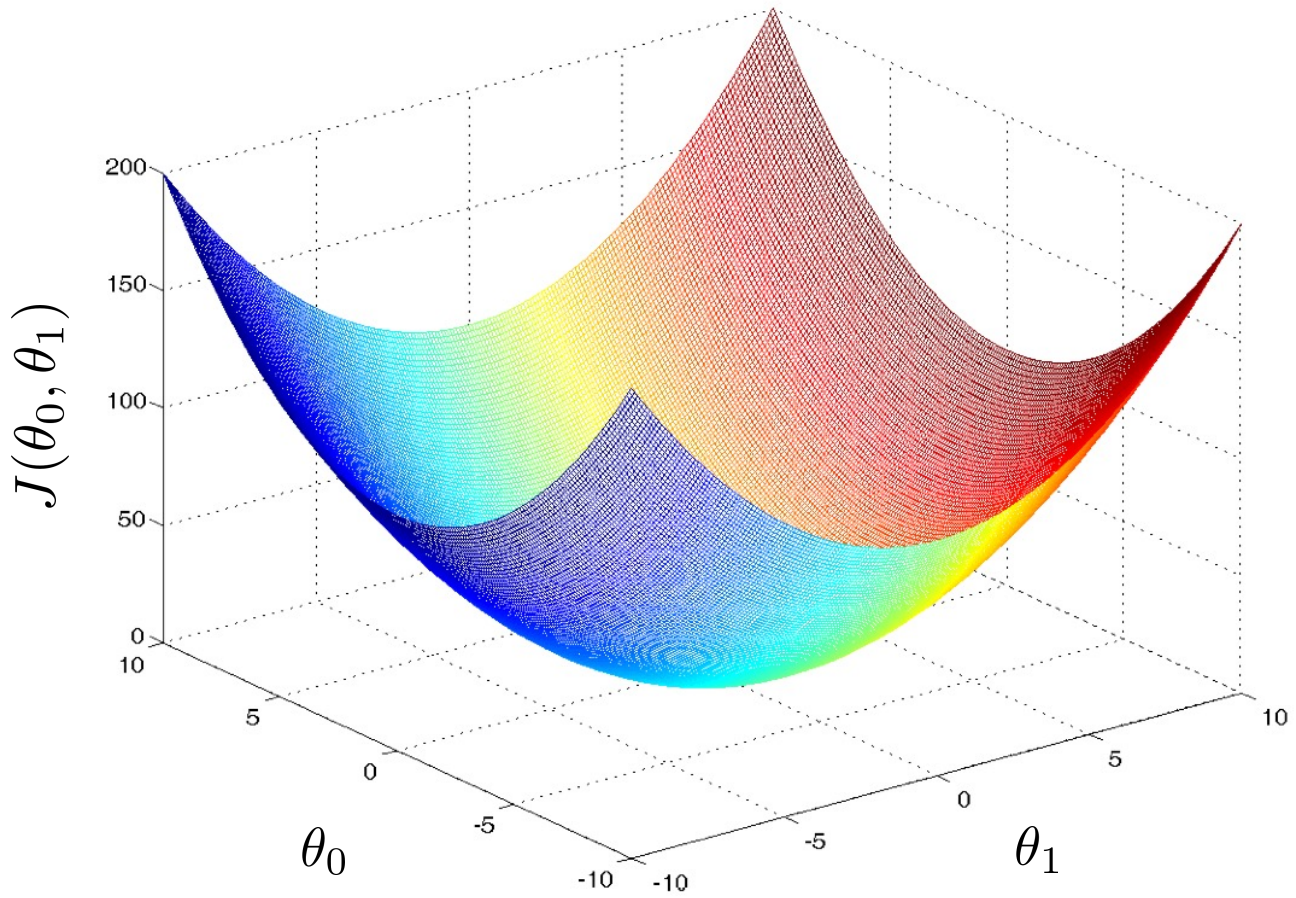
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



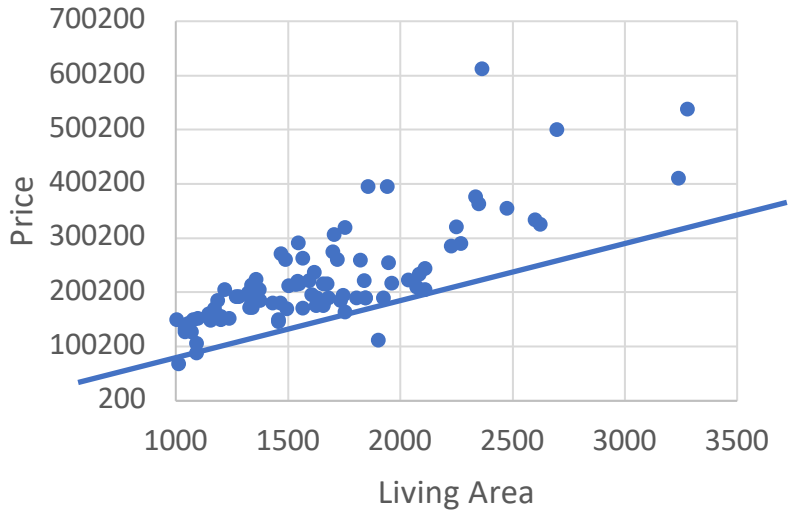
$$\underset{\theta_0,\theta_1}{minimize}(J(\theta_0, \theta_1))$$

$J(\theta_0, \theta_1)$: contour plot



$h_\theta(x)$

(function of x, with θ_1 fixed)



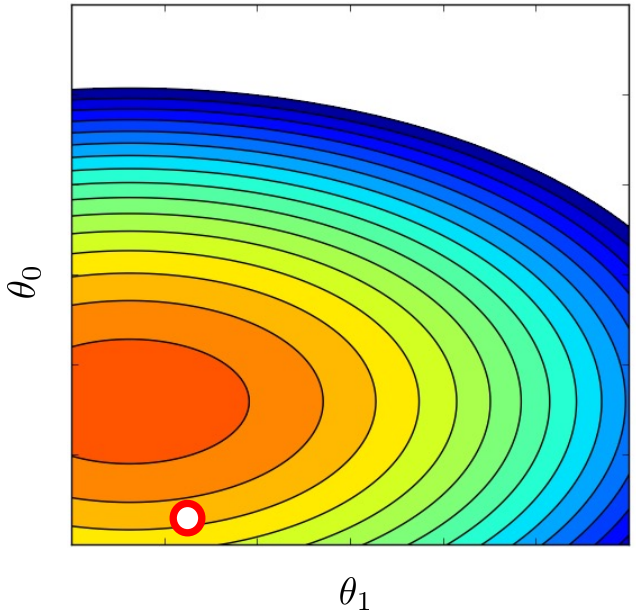
$$\theta_0 = 100000$$

$$\theta_1 = 66$$

$$h_\theta(x^{(i)}) = 100000 + 66x$$

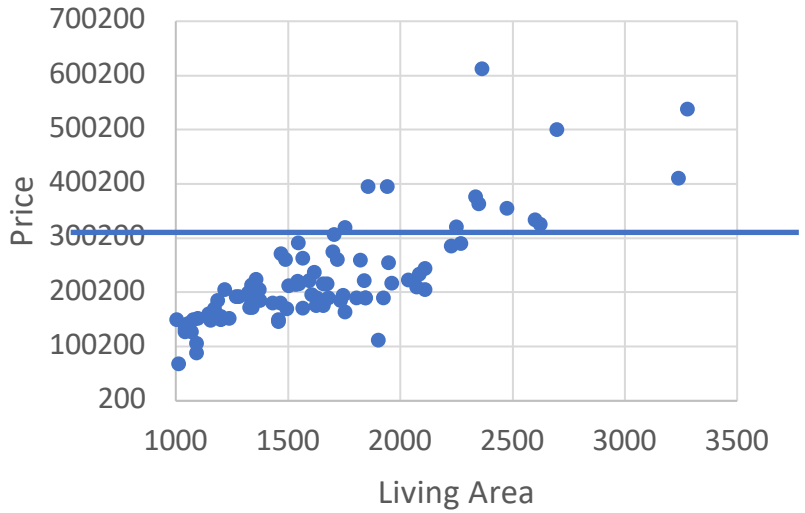
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



$h_\theta(x)$

(function of x, with θ_1 fixed)



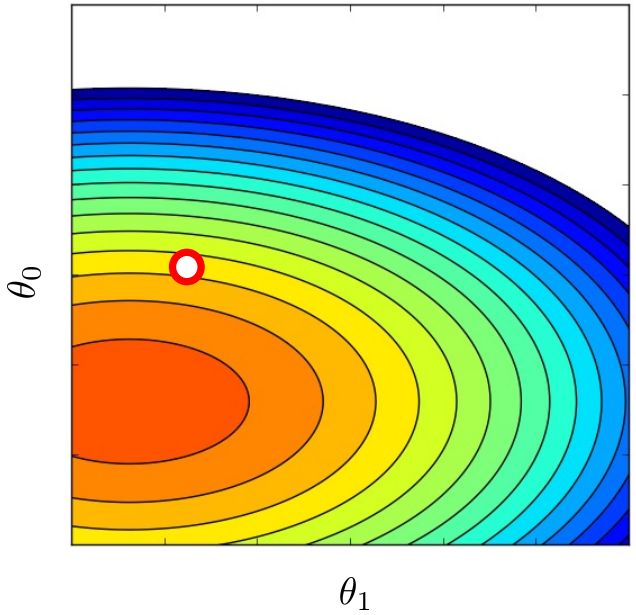
$$\theta_0 = 300200$$

$$\theta_1 = 0$$

$$h_\theta(x^{(i)}) = 300200 + 0x$$

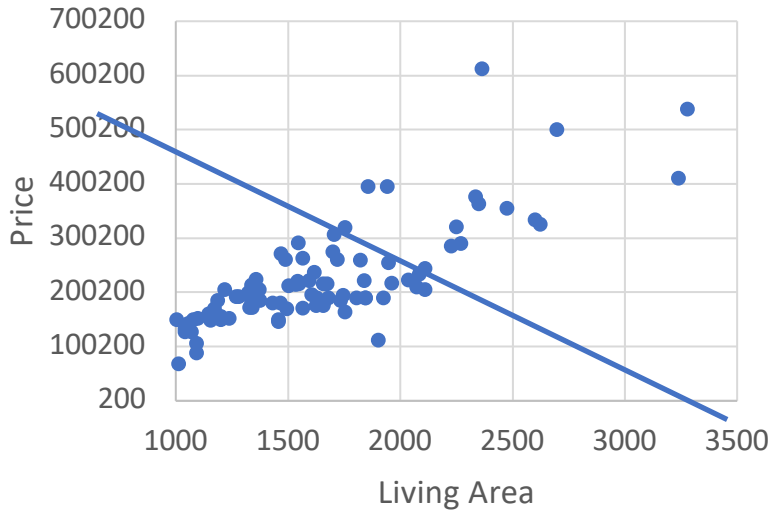
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



$h_\theta(x)$

(function of x, with θ_1 fixed)



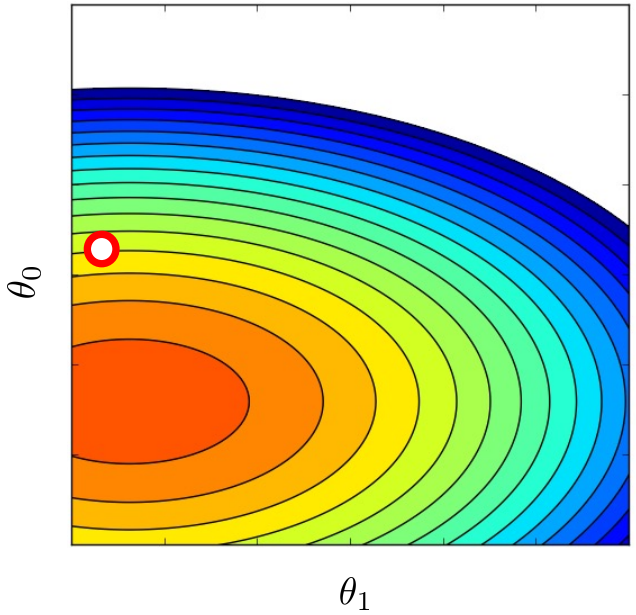
$$\theta_0 = 600000$$

$$\theta_1 = -90$$

$$h_\theta(x^{(i)}) = 600000 - 90x$$

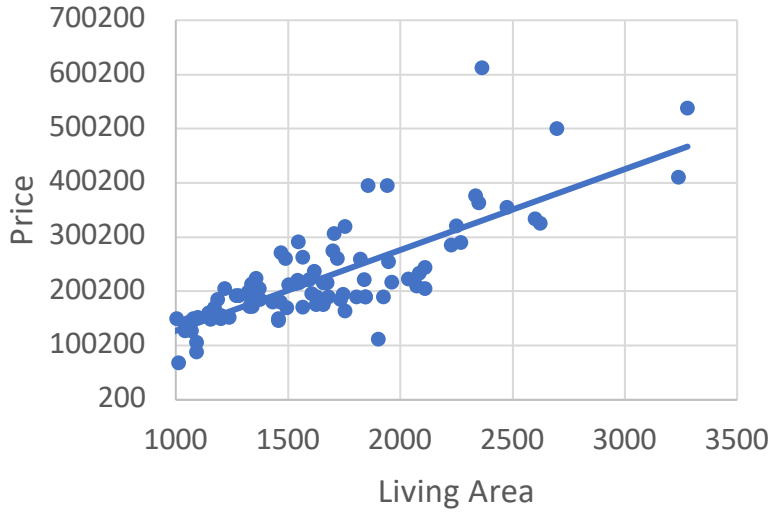
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



$h_\theta(x)$

(function of x, with θ_1 fixed)



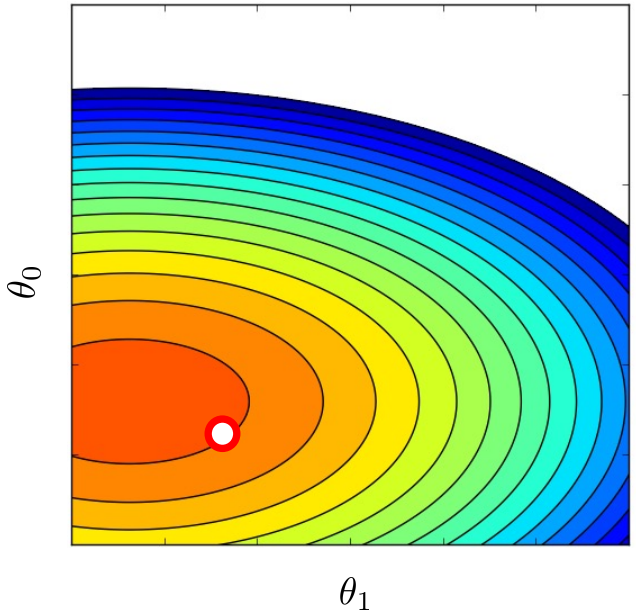
$$\theta_0 = 200000$$

$$\theta_1 = 70$$

$$h_\theta(x^{(i)}) = 200000 + 70x$$

$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



Politecnico di Bari

Linear regression with one variable

Gradient Descent

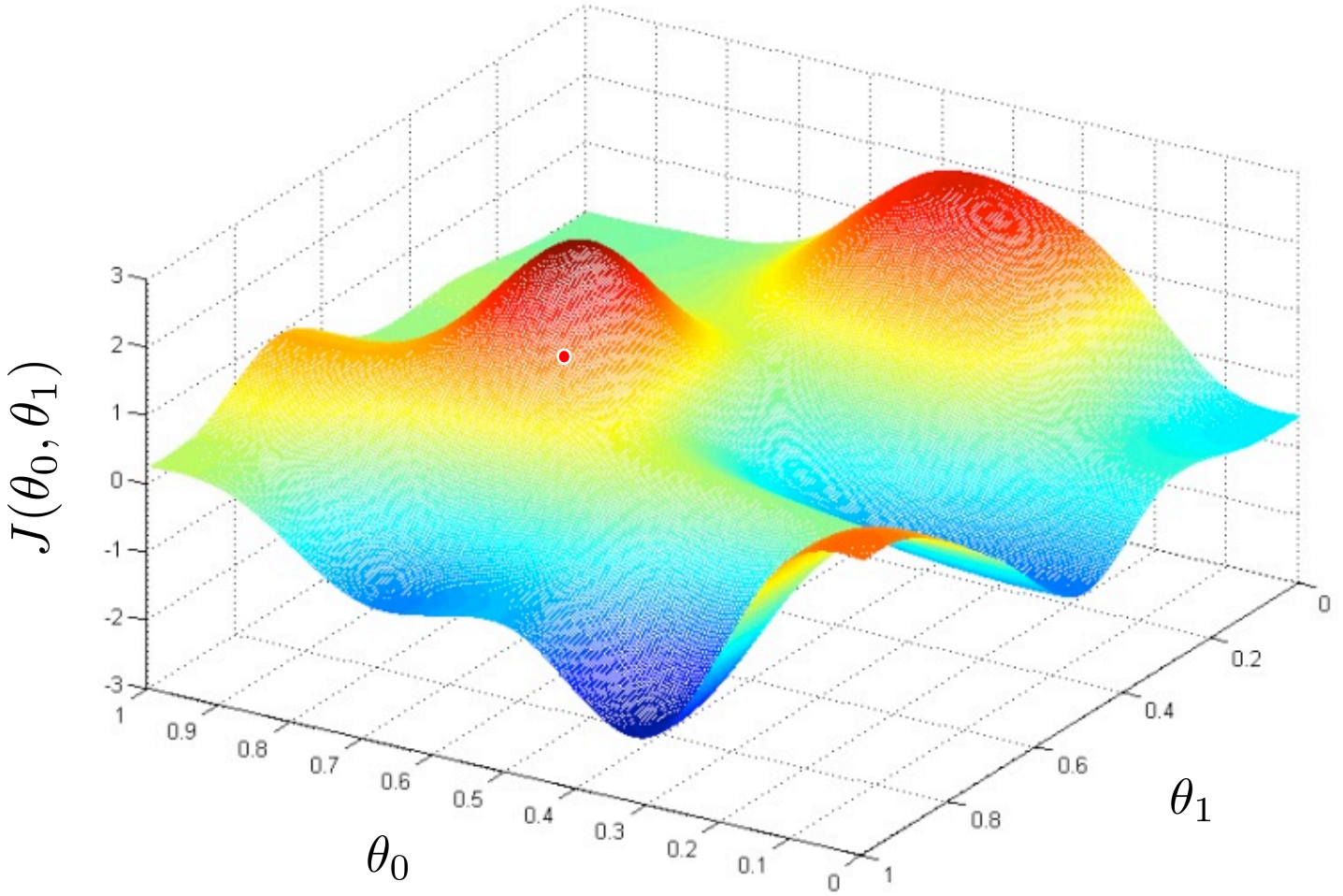
Cost function (we will deal with it as a generic function)

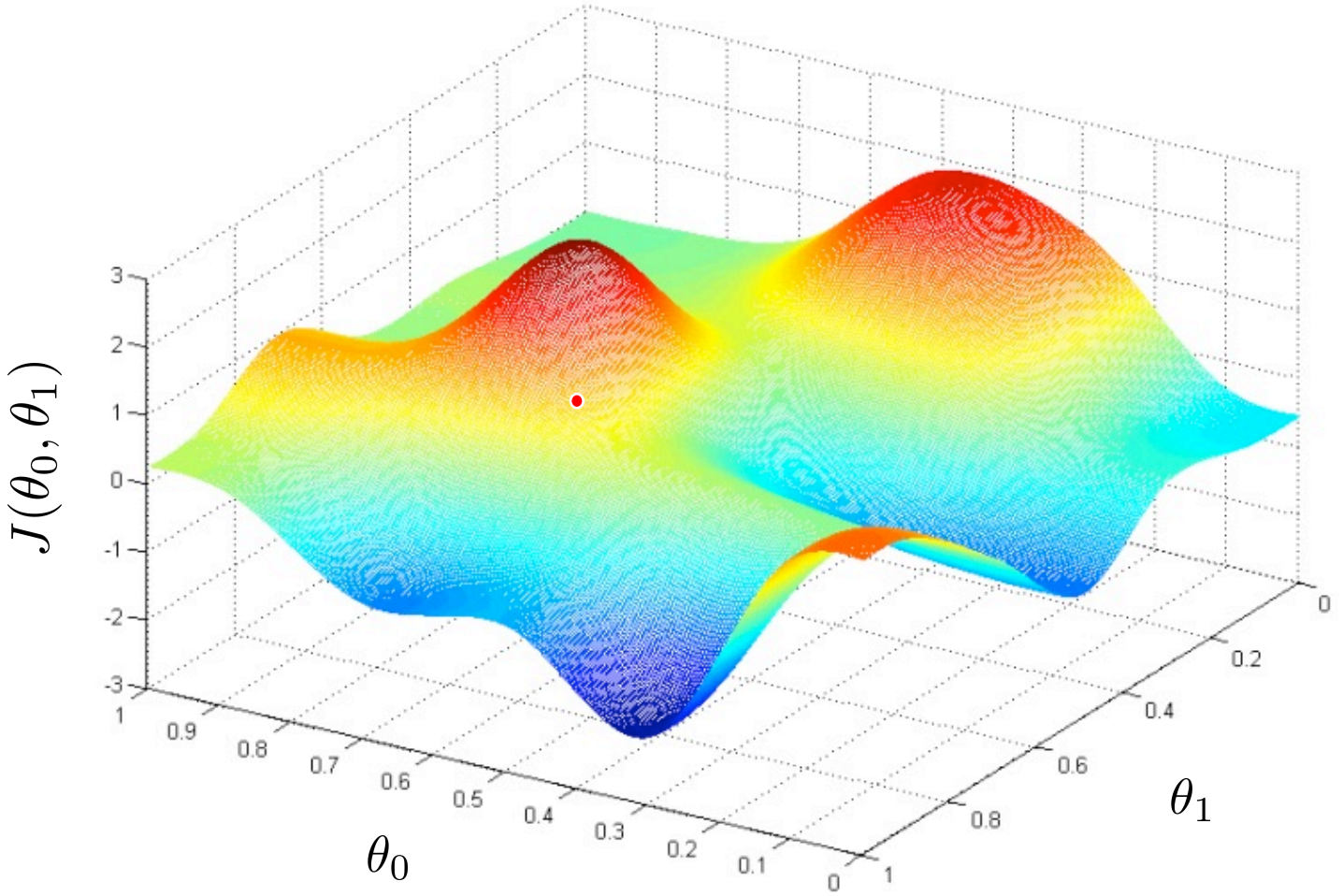
$$J(\theta_0, \theta_1) \qquad\qquad J(\theta_0, \theta_1, ..., \theta_n)$$

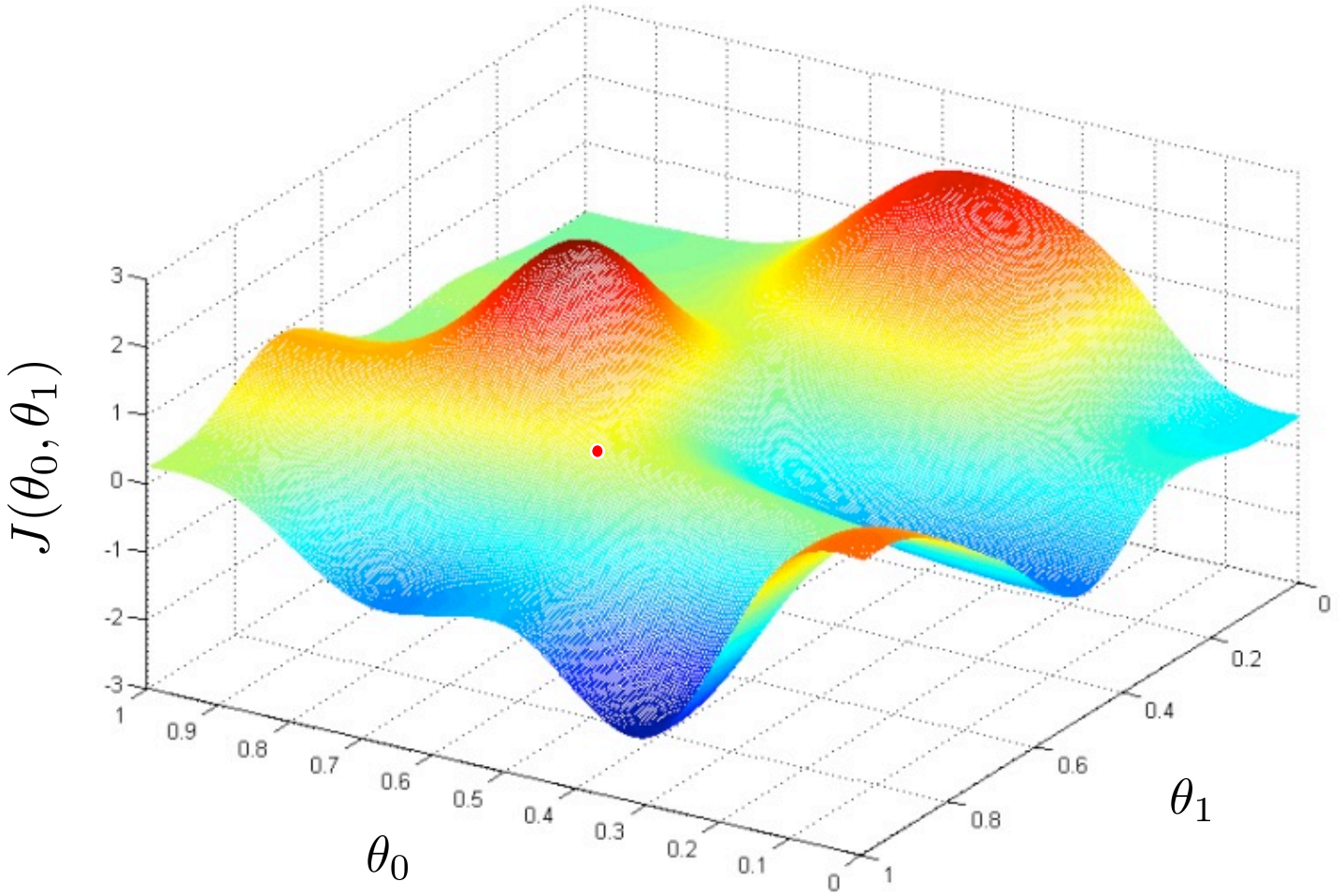
Goal $\min_{\theta_0,\theta_1}(J(\theta_0, \theta_1))$ $\qquad$ $\min_{\theta_0,\theta_1,...,\theta_n}(J(\theta_0, \theta_1, ..., \theta_n))$

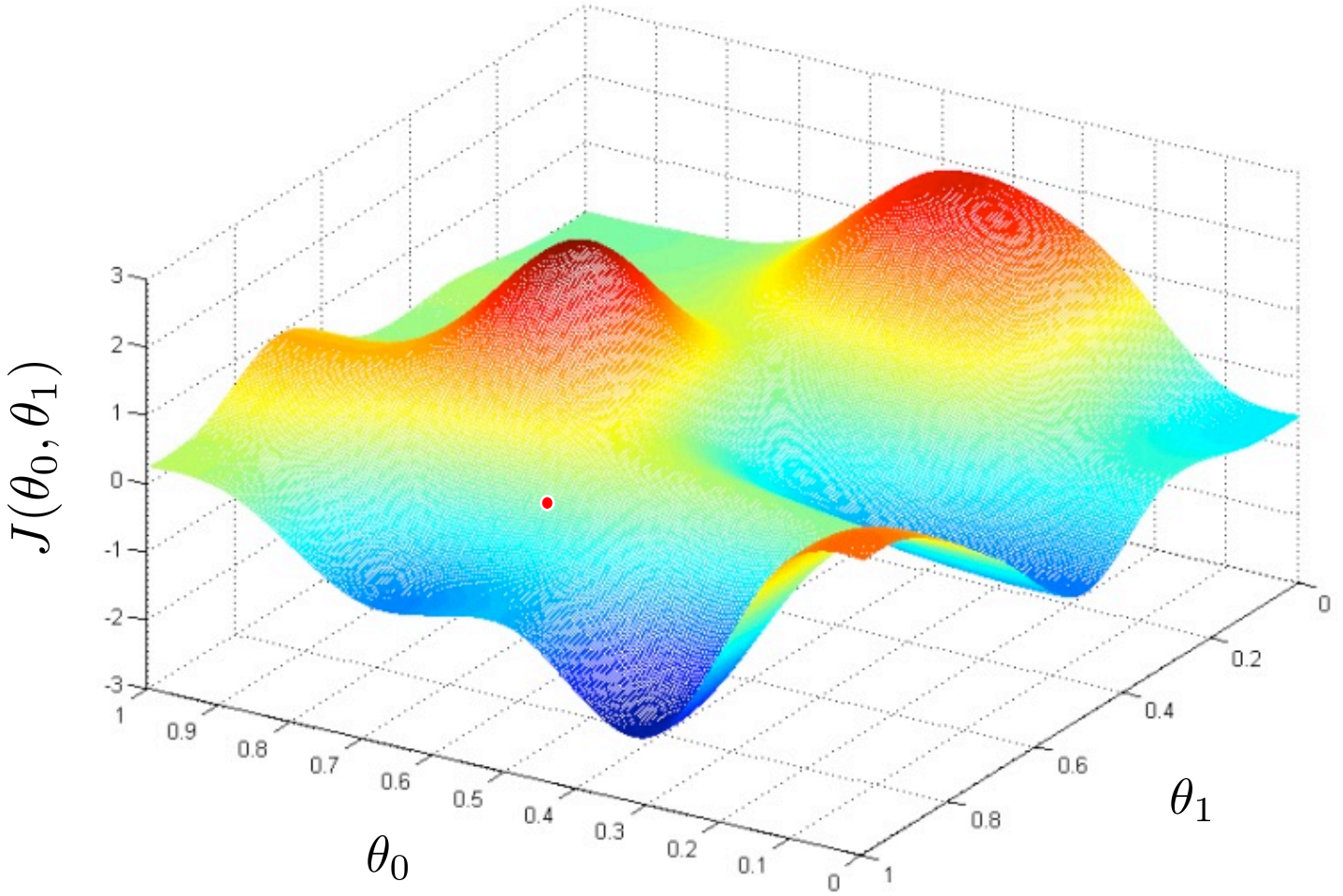
How to:

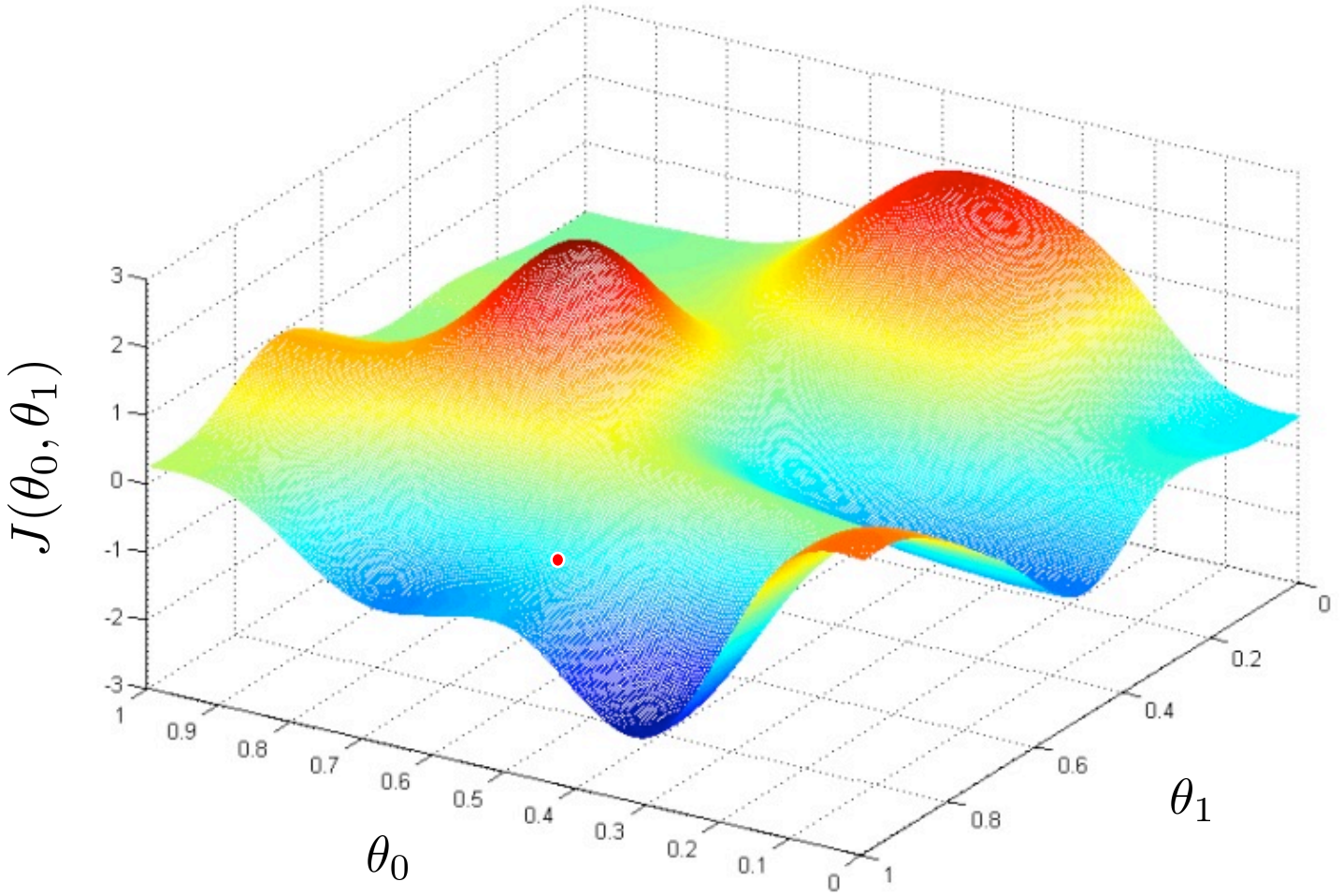
- Initialize θ_0, θ_1 to some values
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ to get closer and reach the minimum

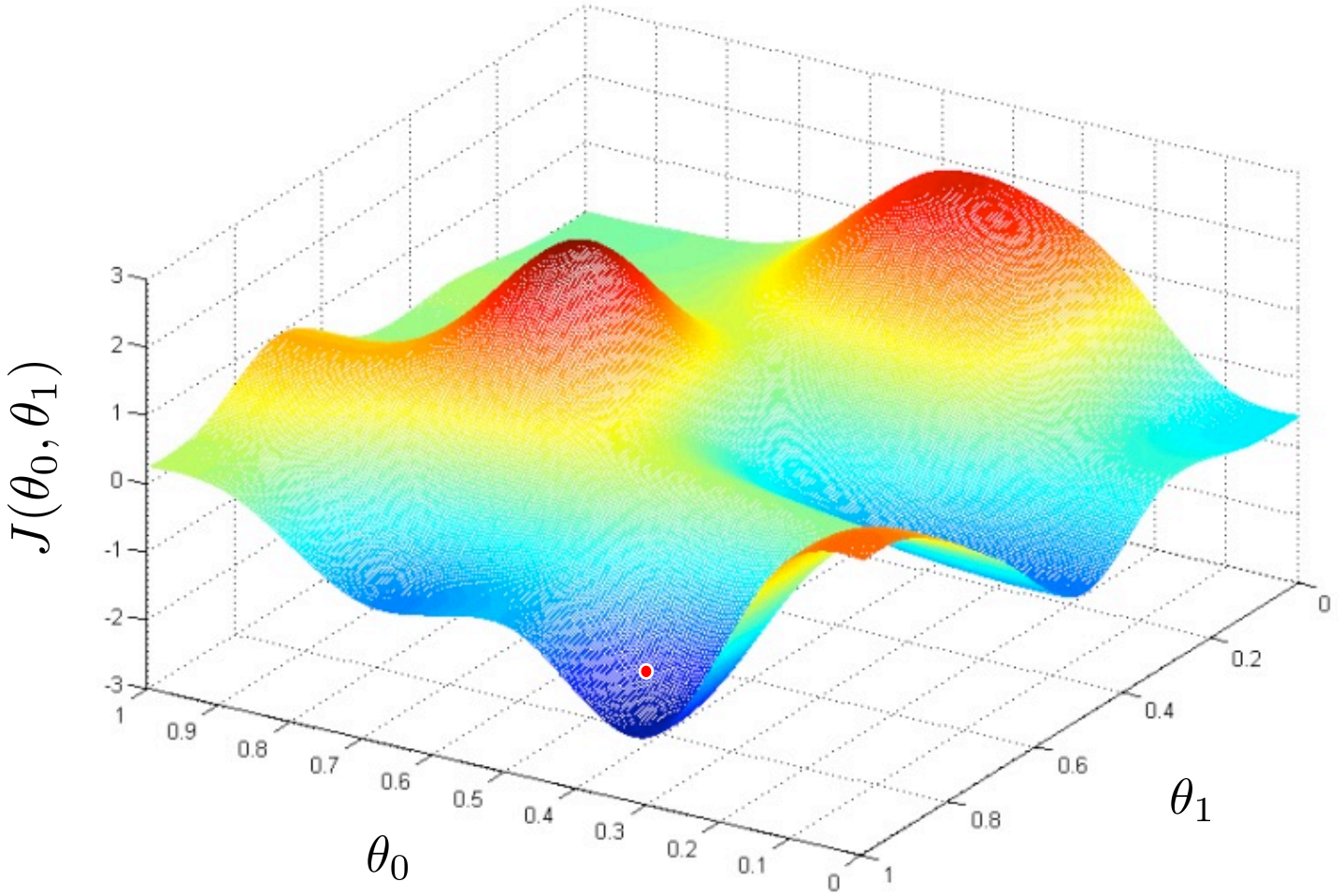

$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

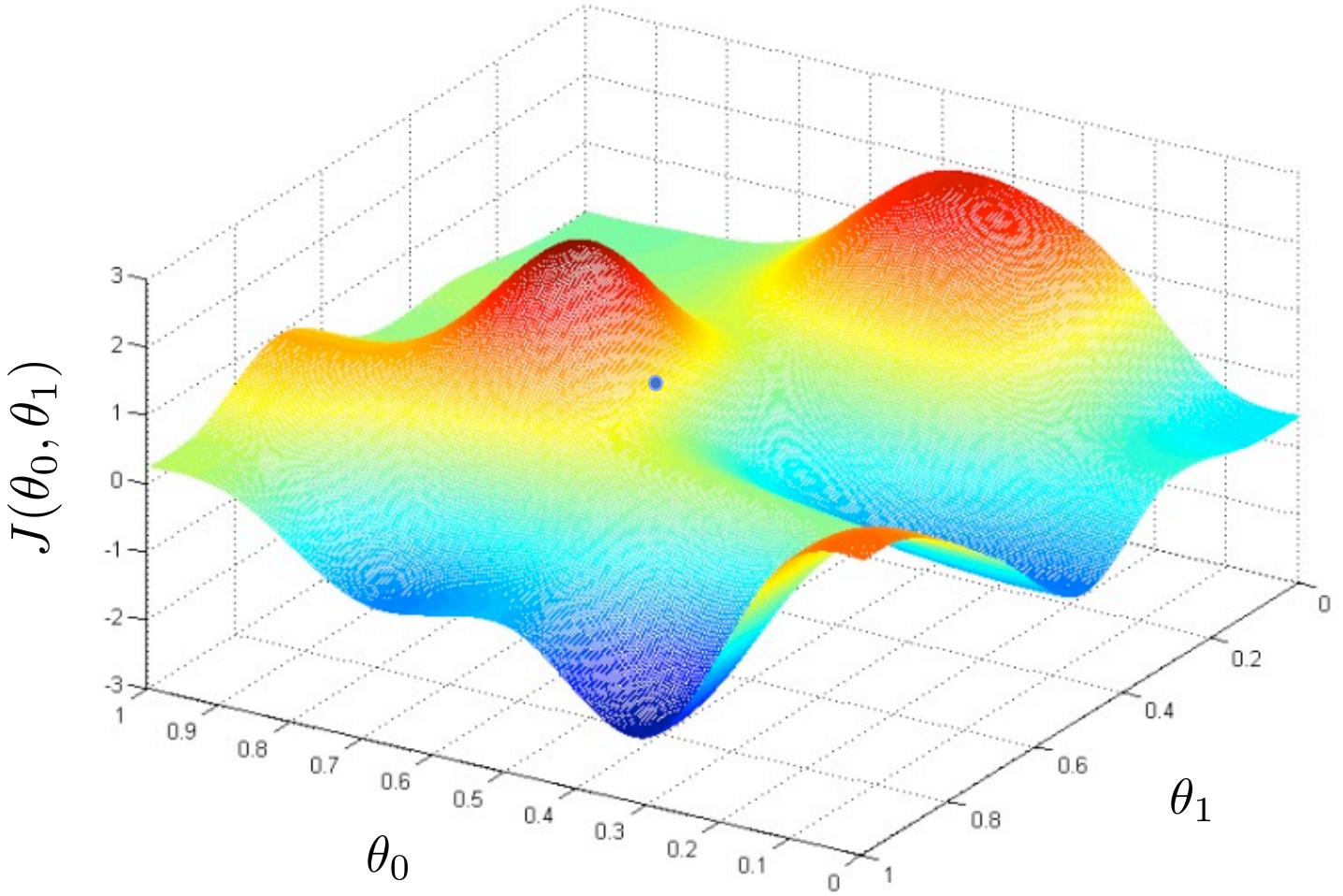

$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

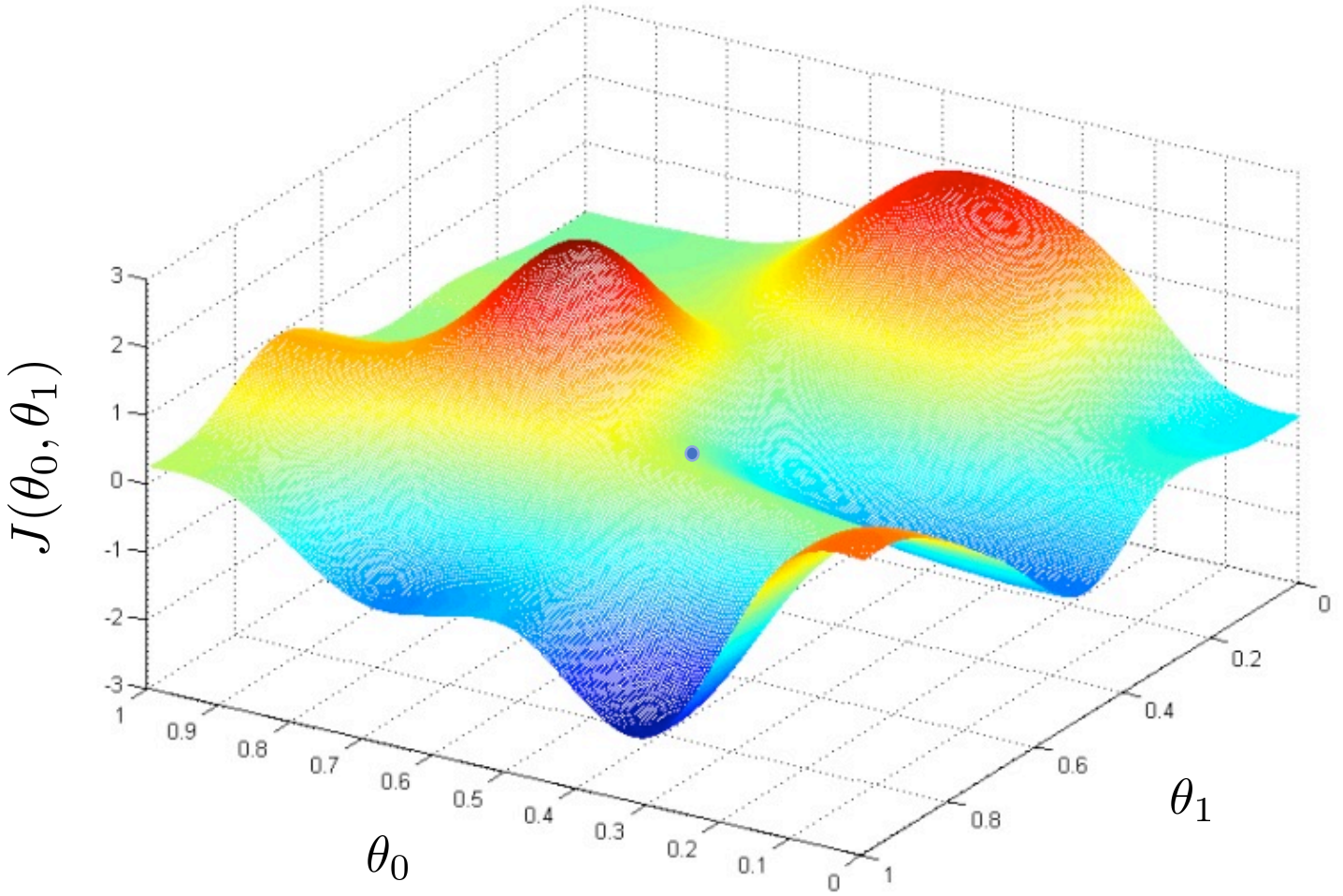

$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

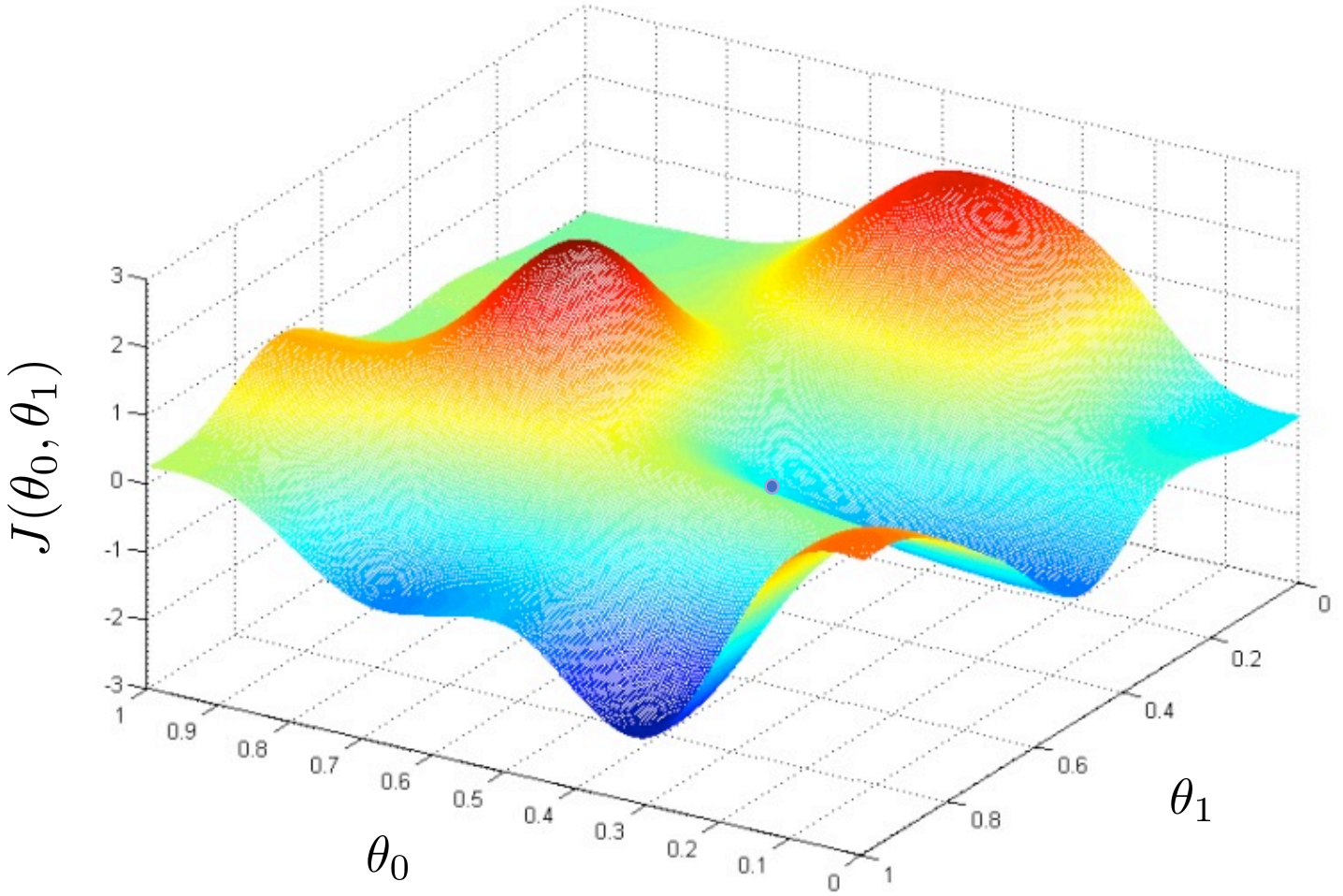

$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

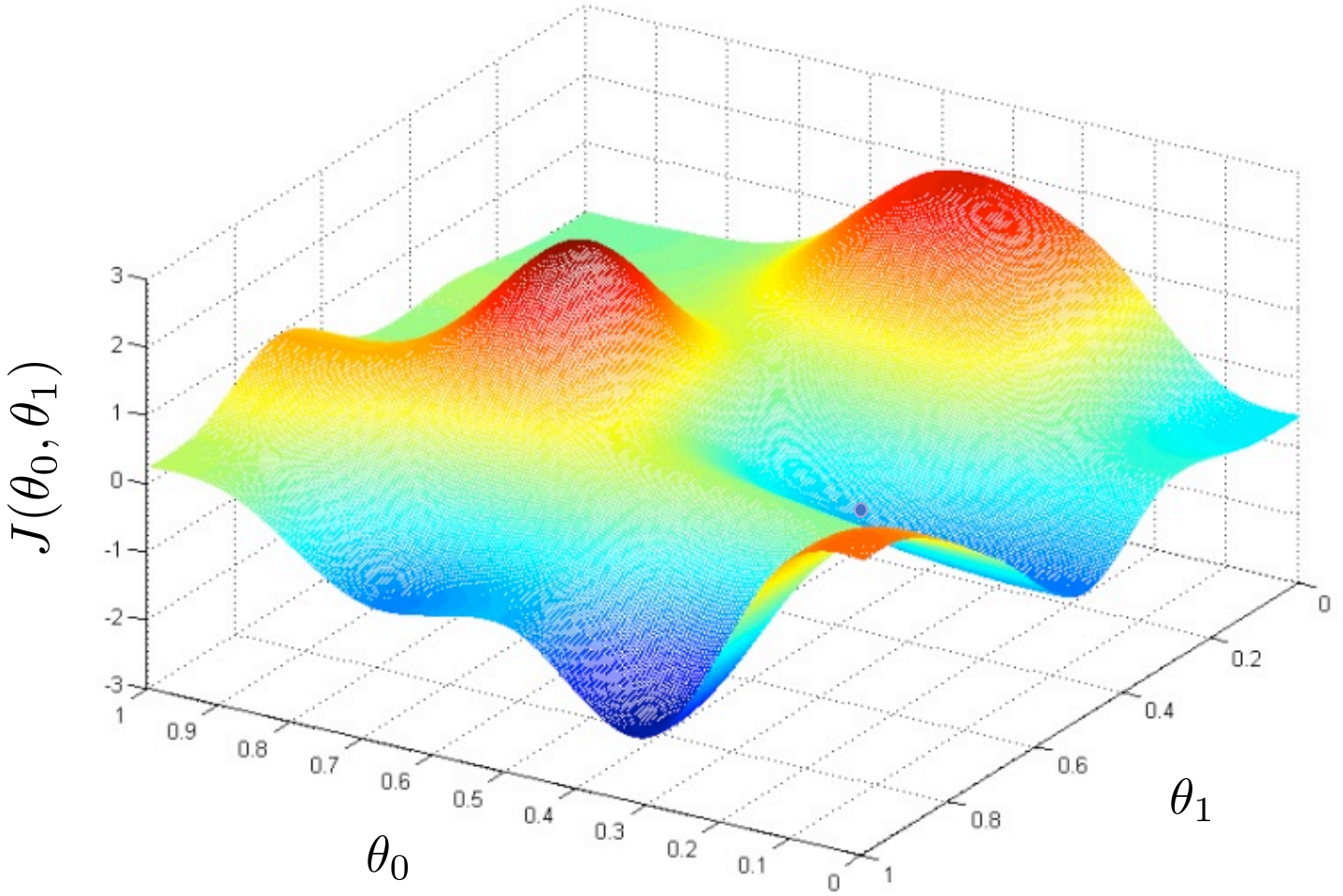

$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

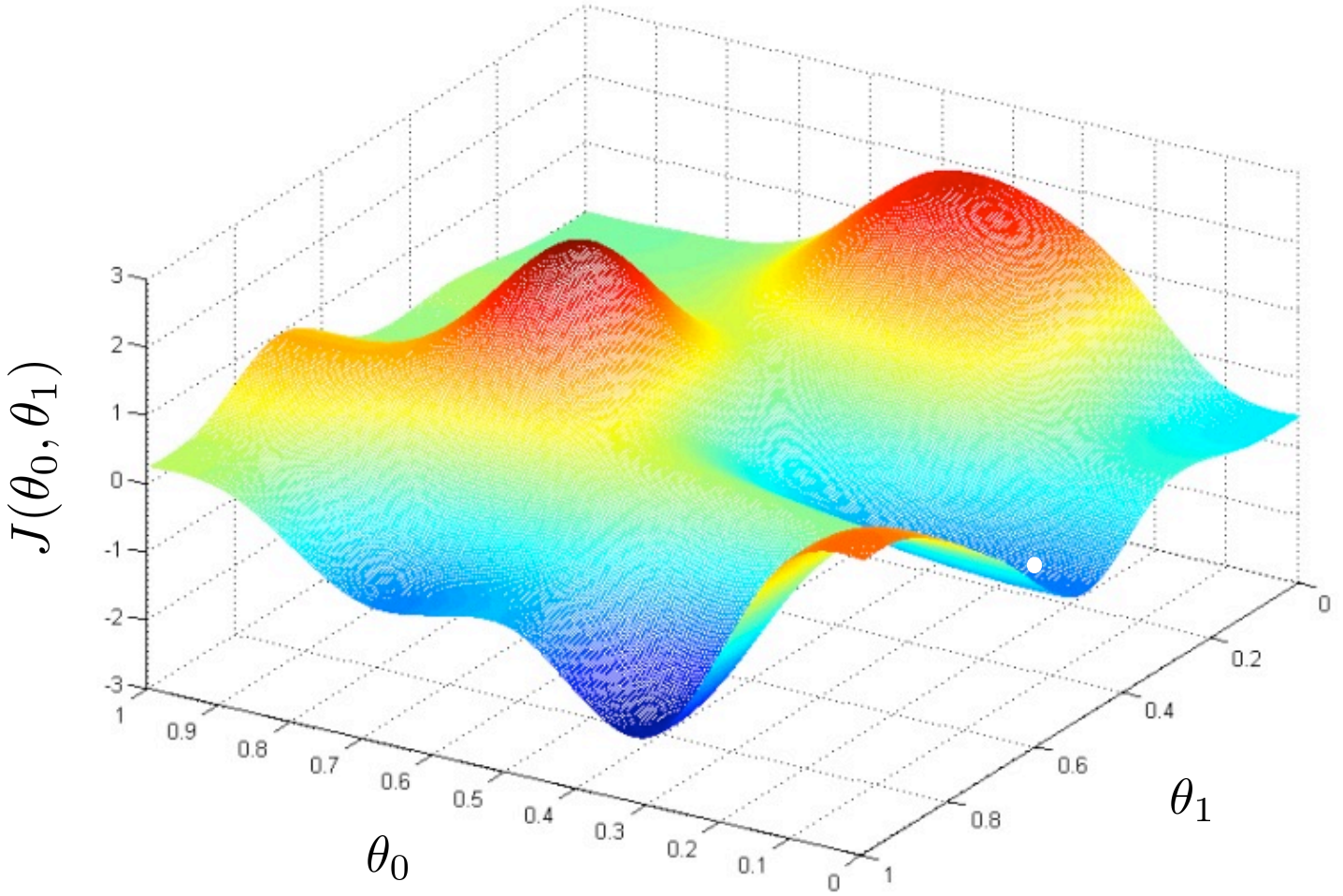

$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Gradient descent algorithm

Repeat *until convergence* {

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

}

Correct: Simultaneous update

Temp0 := $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

Temp1 := $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

θ_0 := temp0

θ_1 := temp1

Incorrect:

Temp0 := $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

θ_0 := temp0

Temp1 := $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

θ_1 := temp1

Politecnico di Bari

Linear regression with one variable

Gradient Descent Intuition

Gradient descent algorithm

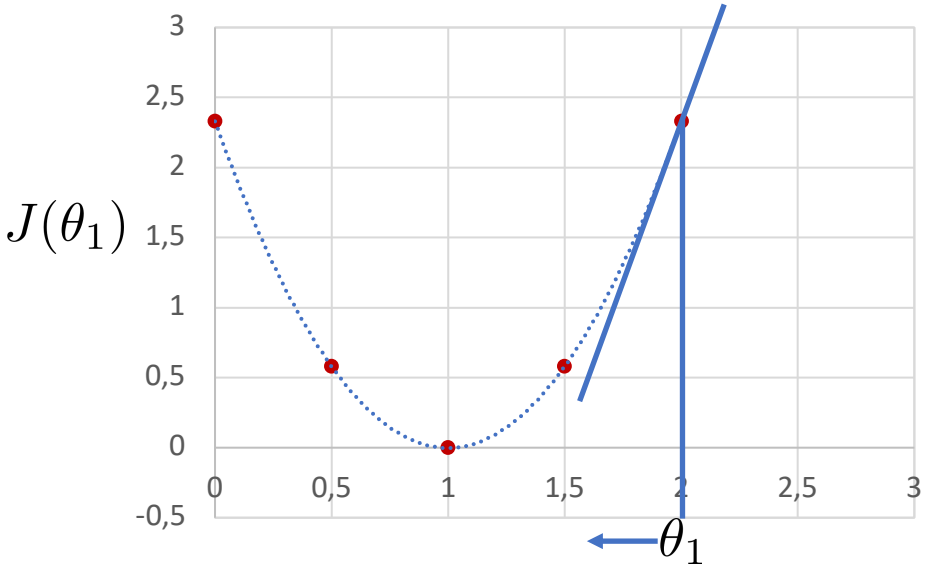
Repeat until convergence {

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

}

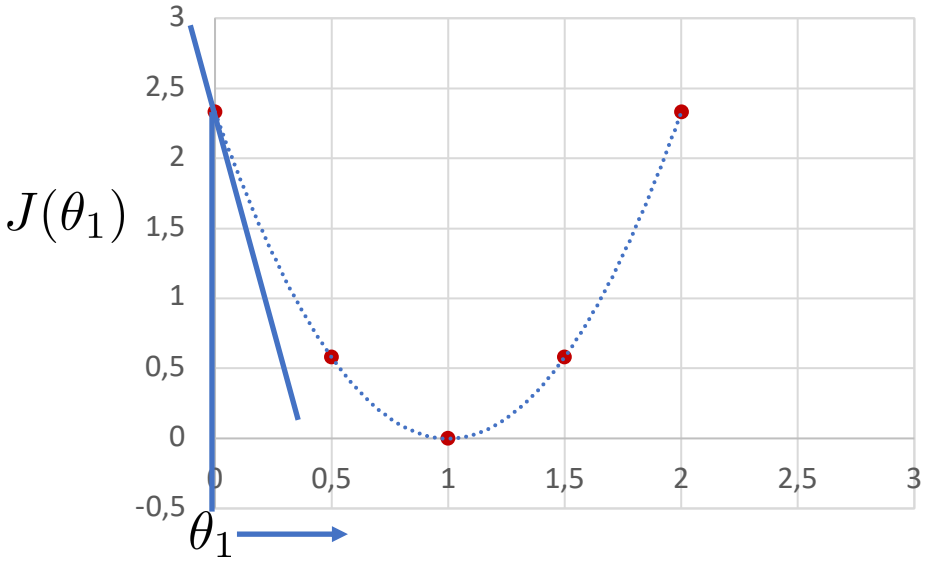
learning rate derivative

Gradient intuition



$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{\partial}{\partial \theta_1} J(\theta_1)}_{\geq 0}$$

$$\theta_1 := \theta_1 - \alpha * (positive \quad number)$$



$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{\partial}{\partial \theta_1} J(\theta_1)}_{\leq 0}$$

$$\theta_1 := \theta_1 - \alpha * (negative \quad number)$$

Learning rate

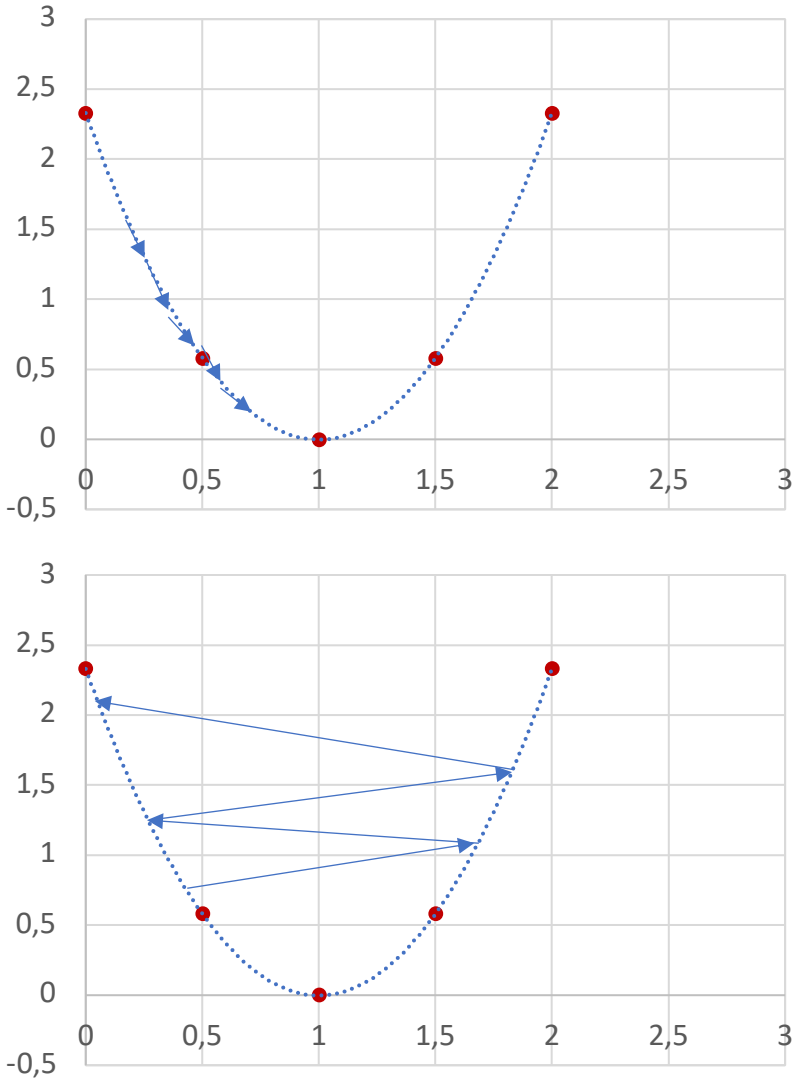
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If alpha is too small,

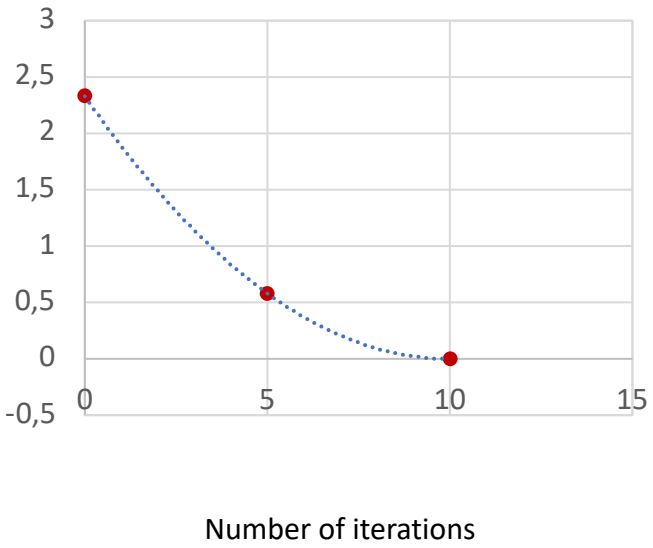
gradient descent is slow

If alpha is too large,

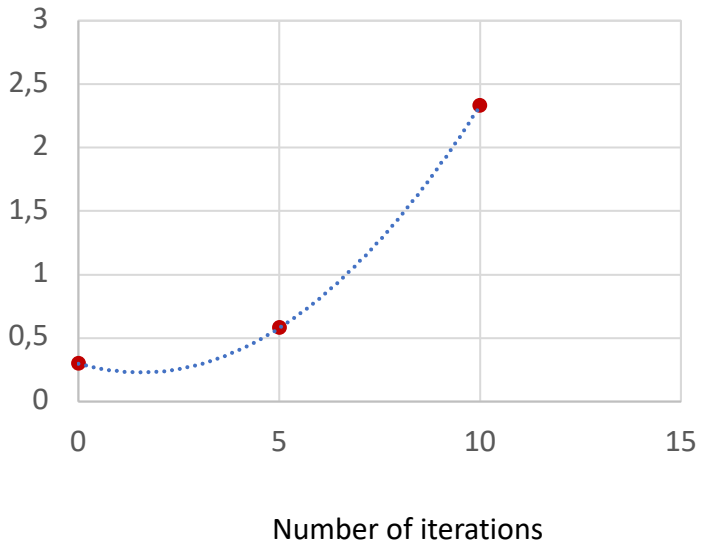
gradient descent can diverge



Learning rate (cont.d)

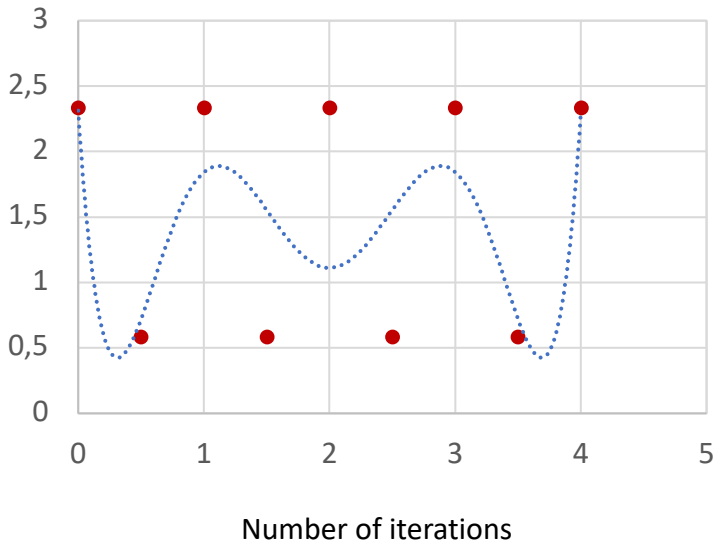


Gradient descent is working



Gradient descent is **NOT** working

Use smaller alpha



Gradient descent is **NOT** working

use smaller alpha

Politecnico di Bari

Linear regression with one variable

Gradient Descent for linear regression

Gradient descent for a linear regression model

Repeat *until convergence* {

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

$$(for j = 0 \quad and \quad j = 1)$$

}

Linear Regression Model

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

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

Deriving the updates

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

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

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

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

Gradient descent algorithm

Repeat until convergence {

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

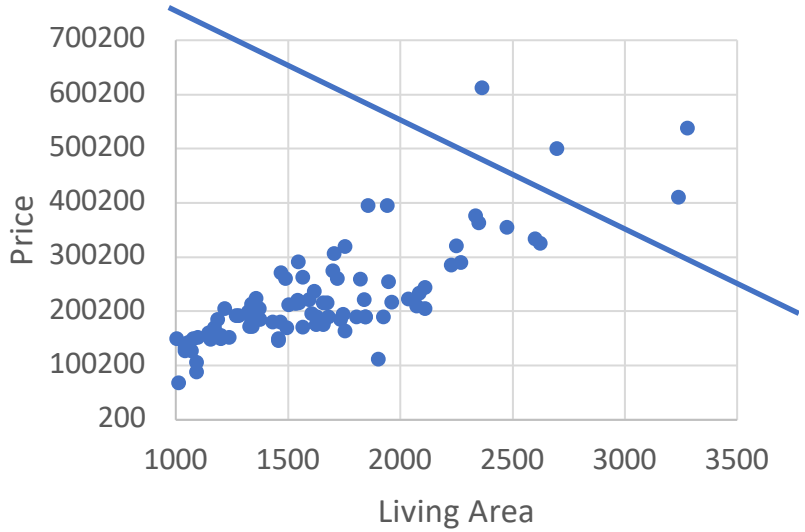
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)})\mathbf{x^{(i)}}$$

simultaneous update

}

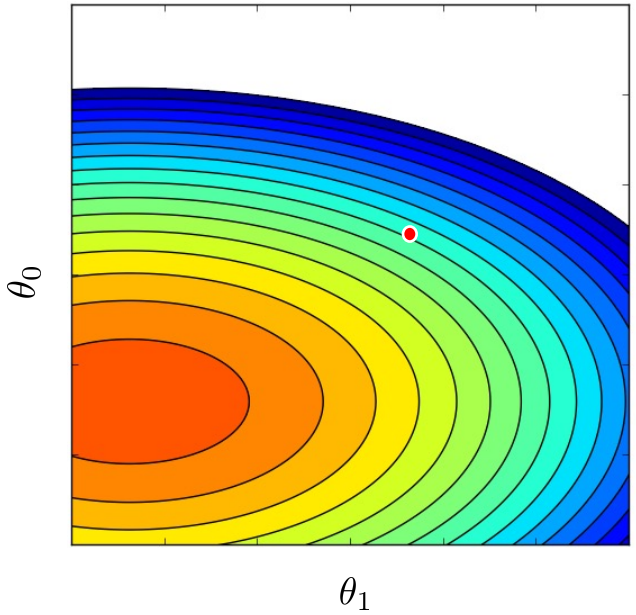
$h_\theta(x)$

(function of x, with θ_0, θ_1 fixed)



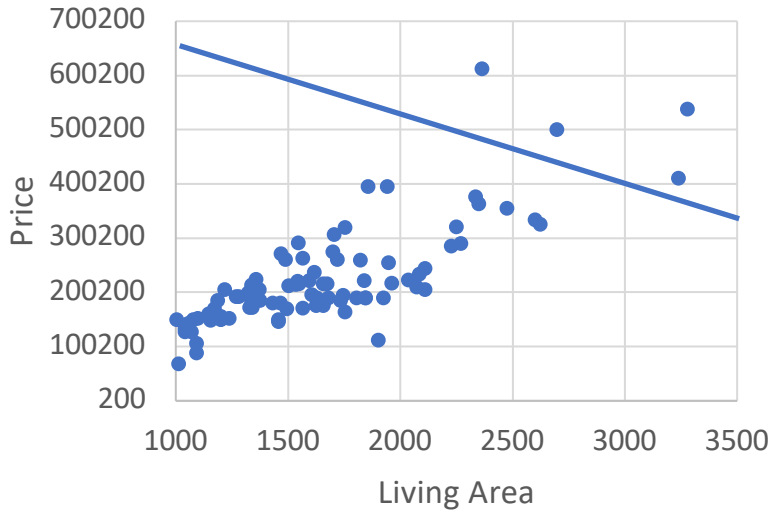
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



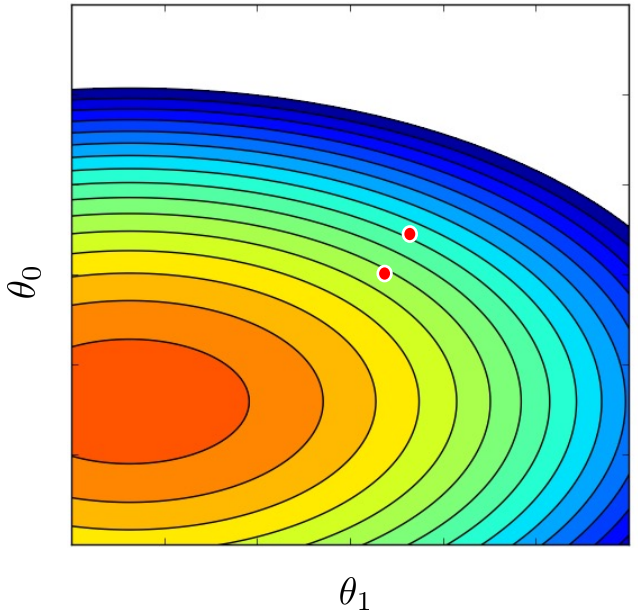
$h_\theta(x)$

(function of x, with θ_0, θ_1 fixed)



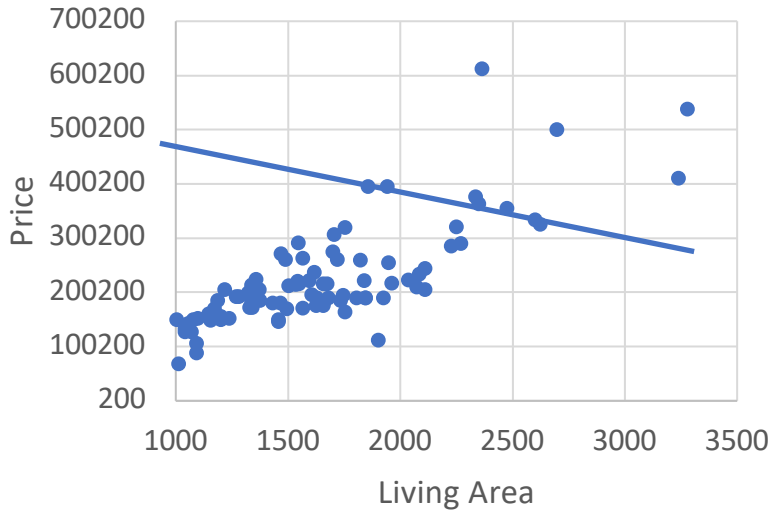
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



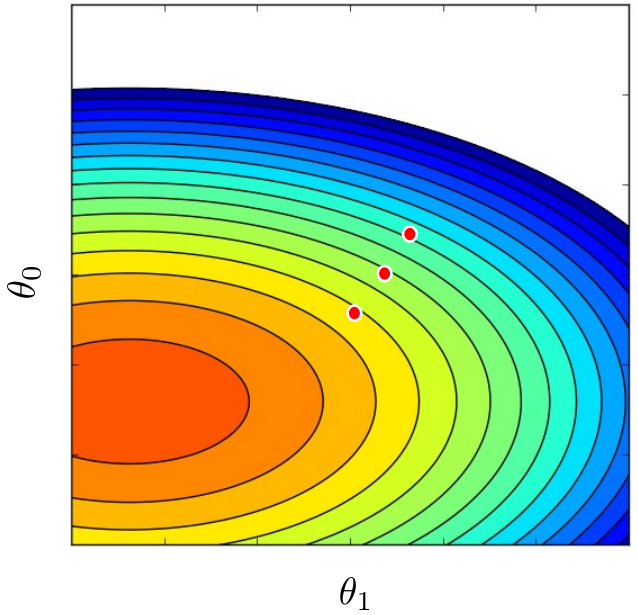
$h_\theta(x)$

(function of x, with θ_0, θ_1 fixed)



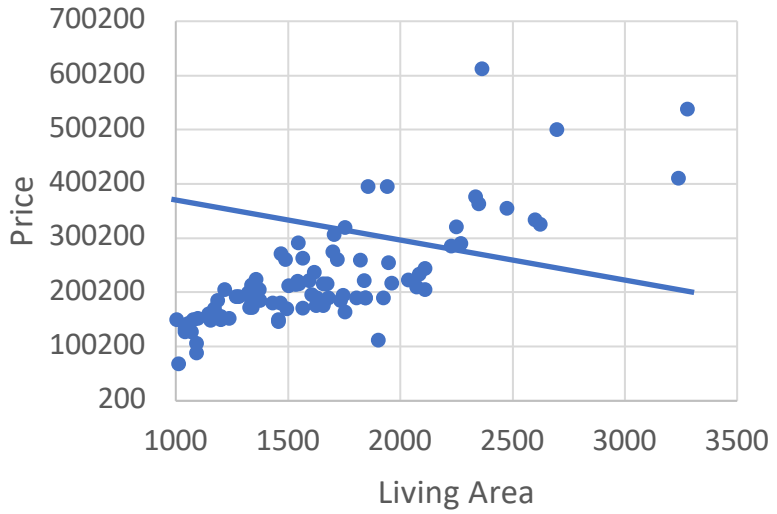
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



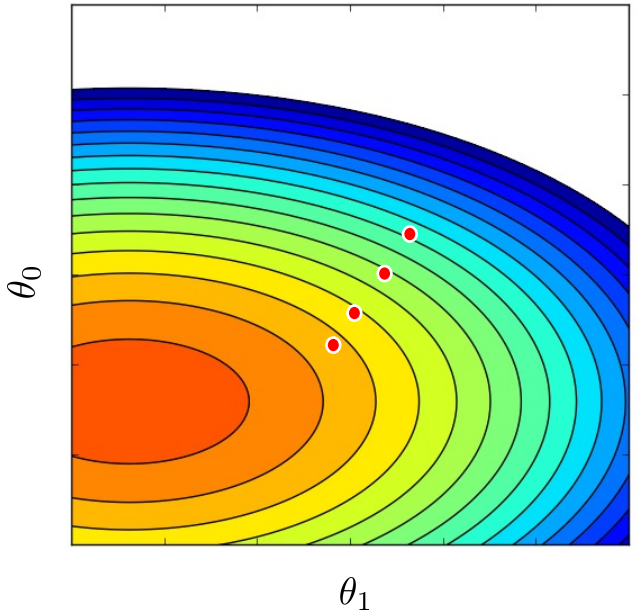
$h_\theta(x)$

(function of x, with θ_0, θ_1 fixed)



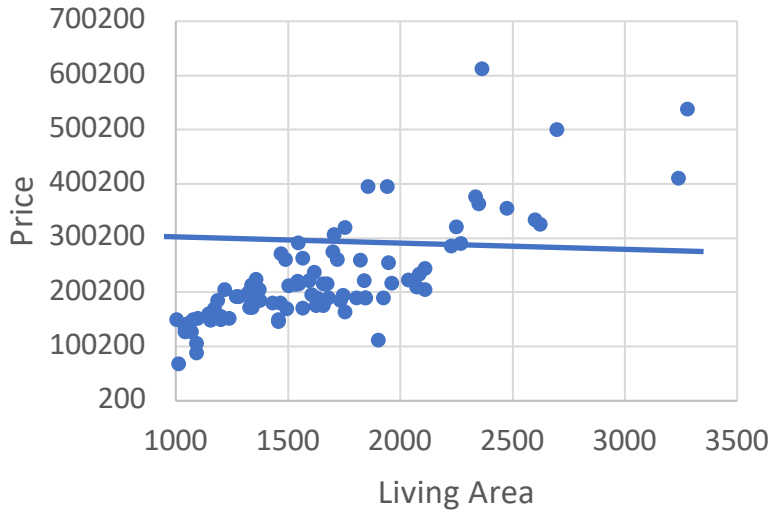
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



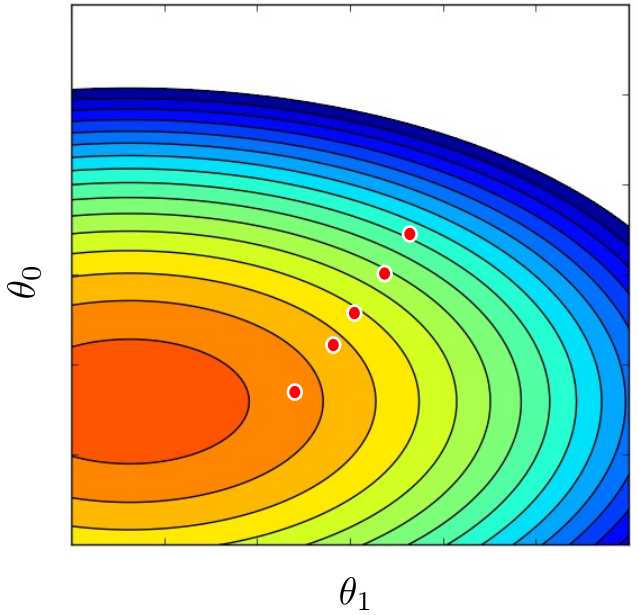
$h_\theta(x)$

(function of x, with θ_0, θ_1 fixed)



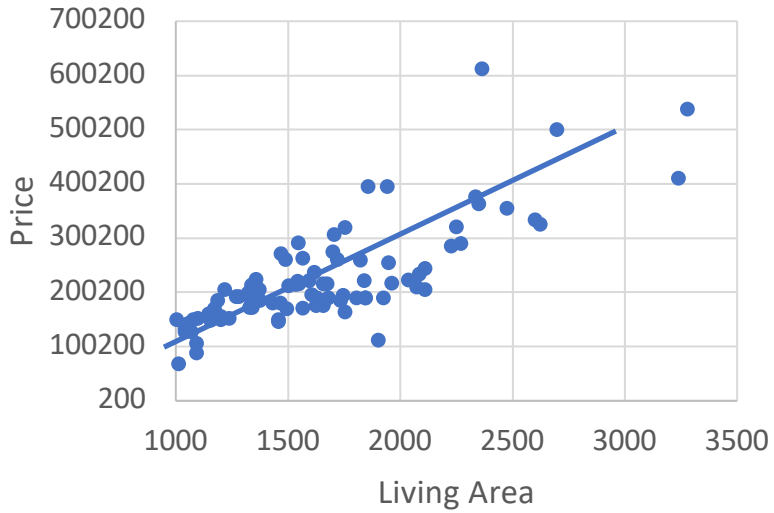
$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



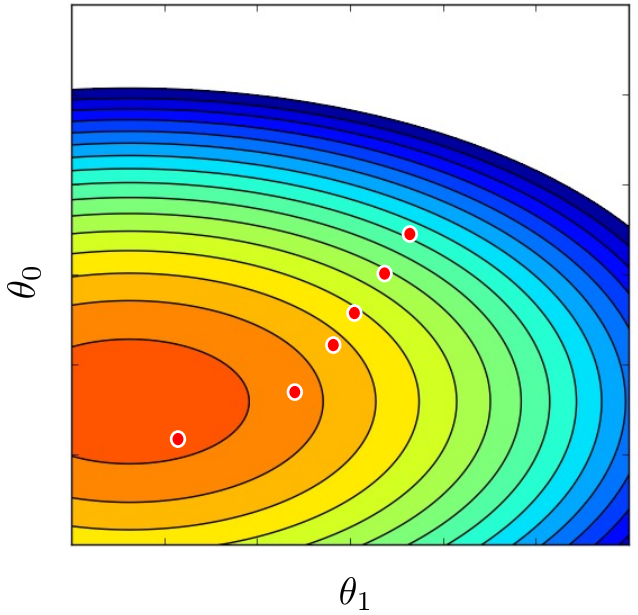
$h_\theta(x)$

(function of x, with θ_0, θ_1 fixed)



$J(\theta_0, \theta_1)$

(function of θ_0, θ_1, an aggregate function over all x's)



Politecnico di Bari

Linear regression

Multiple features

Housing prediction problem in Ames

Training set of housing prices

Living area ($feet^2$)	Number of bedrooms	Number of floors	Price (1000$)
1656	5	1	215
896	3	2	105
1329	4	2	172
2110	2	1	244
…	…	…	…

Notation:

m = Number of training examples

n = Number of features

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

$y^{(i)}$ = "output" variable / "target" variable

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

$x_1^{(0)}$	1656
$x_1^{(1)}$	896
$x_2^{(3)}$	2

Hypothesis with multiple features

Previous hypothesis:

$$h_\theta(\mathbf{x^{(i)}}) = \theta_0 + \theta_1 x^{(i)}$$

$$h_\theta(\mathbf{x^{(i)}}) = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \theta_3 x_3^{(i)}$$

Multivariate linear regression

$$h_\theta(\mathbf{x^{(i)}}) = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \ldots + \theta_n x_n^{(i)}$$

For convenience we set $x_0^{(i)} = 1$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_0} \\ \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_n} \end{bmatrix} \in \mathbb{R}^{n+1} \quad \theta = \begin{bmatrix} \theta_\mathbf{0} \\ \theta_\mathbf{1} \\ \theta_\mathbf{2} \\ \vdots \\ \theta_\mathbf{n} \end{bmatrix} \in \mathbb{R}^{n+1}$$

Multivariate linear regression

$$\underbrace{\underbrace{\begin{bmatrix}\theta_0 & \theta_1 & \dots & \theta_n\end{bmatrix}}_{\theta^T}\underbrace{\begin{bmatrix}x_0\\x_1\\\vdots\\x_n\end{bmatrix}}_{\mathbf{x}}}_{\theta^T\mathbf{x}}$$

$$\begin{aligned} h_\theta(\mathbf{x}^{(\mathbf{i})}) &= \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \ldots + \theta_n x_n^{(i)} \\ &= \theta^T x. \end{aligned}$$

Batch Gradient descent

1. Randomly initialize parameters
2. Repeat *until convergence* {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)}) \mathbf{x_j^{(i)}} \quad (for \quad j = 0, 1, ..., n)$$

}

Stochastic Gradient descent

1. Randomly initialize parameters
2. Repeat *until convergence* {

    for each **training case** {

$$\theta_j := \theta_j - \alpha(h_\theta(\mathbf{x^{(i)}}) - y^{(i)})\mathbf{x_j^{(i)}} \quad (for \quad j = 0, 1, ..., n)$$

    }

}

- This version is called ***stochastic*** or ***incremental*** because the parameters are updated after each training sample, therefore the gradient is a "stochastic approximation" of the "true" cost gradient.
- The optimal solution is generally approximated faster, but the algorithm could swing around it without never getting convergence (*zig-zag* problem). A solution close to the optimal one is often acceptable.
- For large datasets, Stochastic gradient descent must be preferred over Batch version.
- The parameters are continuously updated. It means that the function *h* will use different values at each iteration

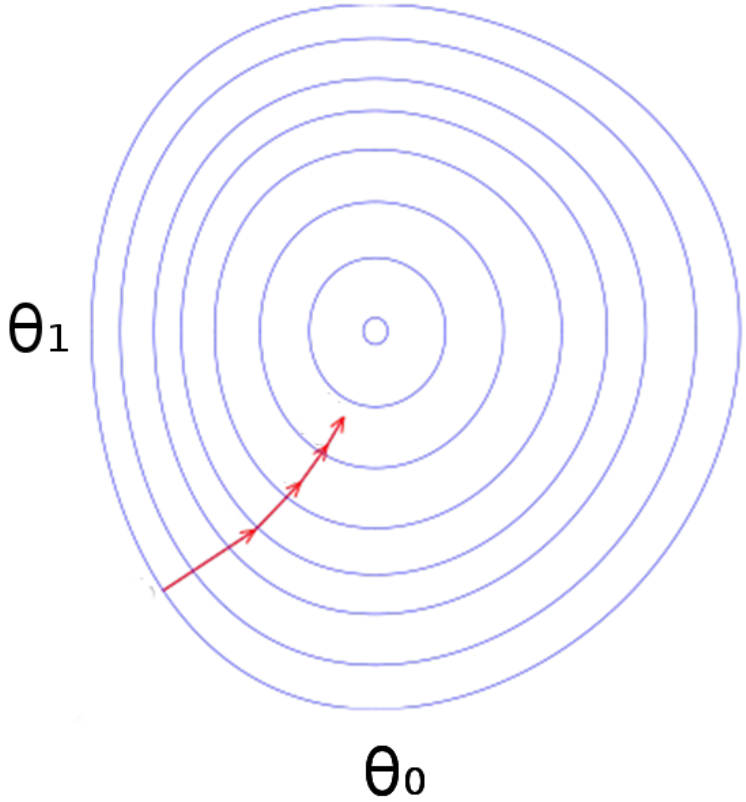
Batch GD vs Stochastic GD

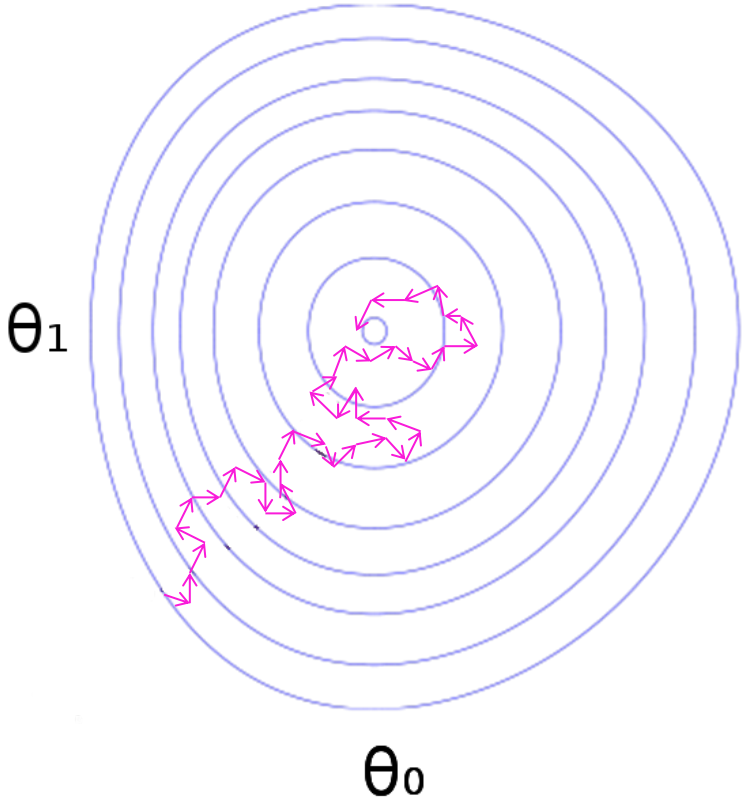
```
theta = rand();
while (not convergence){
 # iteration over theta parameters
  for( j from 1 to  n){
    update = 0;
    # iteration over training samples
    for( i from 1 to  m)}
      update = update + (h(x[i]) - y[i]) x[i];
    }
    theta_new[j] = theta[j] – alpha * update / m;
  }
  theta = theta_new;
}
```

```
theta = rand();
while (not convergence){
 # iteration over training samples
  for(i from 1 to  m){
    # iteration over theta parameters
    for(j from 1 to  n){
      theta_new[j] = theta[j] – alpha * (h(x[i]) - y[i]) x[i];
    }
    theta = theta_new;
  }
}
```

Pay attention at the position of the for loops

Batch GD vs Stochastic GD





Each “jump” is direct toward the minimum, but the relative update step goes through all the examples

Slow, but convergence is sure

Each “jump” may go potentially everywhere, but it is relative to just one training sample

Fast, but convergence is not sure

Mini-Batch gradient descent

Compromise between stochastic and batch gradient descent: takes the advantage of both.

Performs an update for every mini-batch of b training examples

where $1 < b << m$ (m is the total number of training samples)

Usually $b \in [2,100]$

In this way it reduces the variance of the parameter updates, leading to more stable convergence (more robust), and allows the use of vectorization libraries rather than computing each step separately (more efficient)

Mini-Batch gradient descent

```
b = 50;
while (not convergence){
 # iteration over training samples
  while(i  <  m){
    # iteration over theta parameters
    for(j from 1 to  n){
        update = 0;
        # iteration over the b samples
        for( z from i to i + b ){
           update = update + (h(x[z]) - y[z]) x[z];
         }
         theta[j] = theta[j] – alpha * update / b;
    }
  i = i + b;
  }
}
```

Gradient descent algorithm stopping conditions

- **Max iteration**: the algorithm stops after a fixed number of iterations
- **Absolute tolerance**: the algorithm stops when a fixed value of the cost function is reached
- **Relative tolerance**: the algorithm stops when the decreasing of the cost function w.r.t. the previous step is below a fixed rate.
- **Gradient Norm tolerance**: the algorithm stops when the norm of the gradient is lower than a fixed value

Politecnico di Bari

Linear regression

Features and Polynomial regression

Adding features

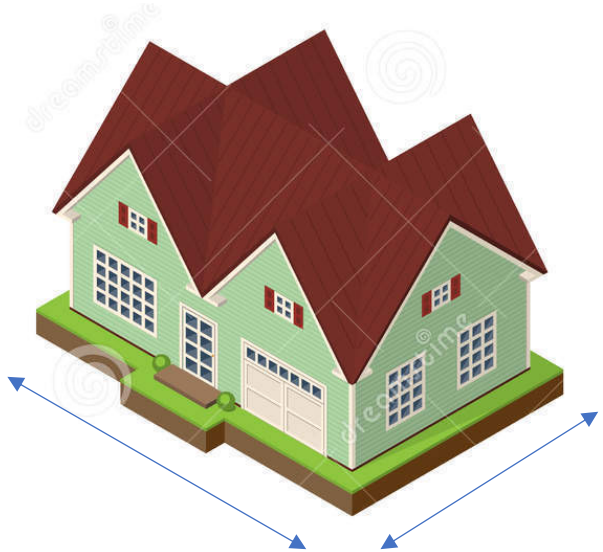
$$h_\theta(\mathbf{x}^{(\mathbf{i})}) = \theta_0 + \theta_1 \cdot front^{(i)} + \theta_2 \cdot side^{(i)}$$

Front and side information are not informative but Area could be:

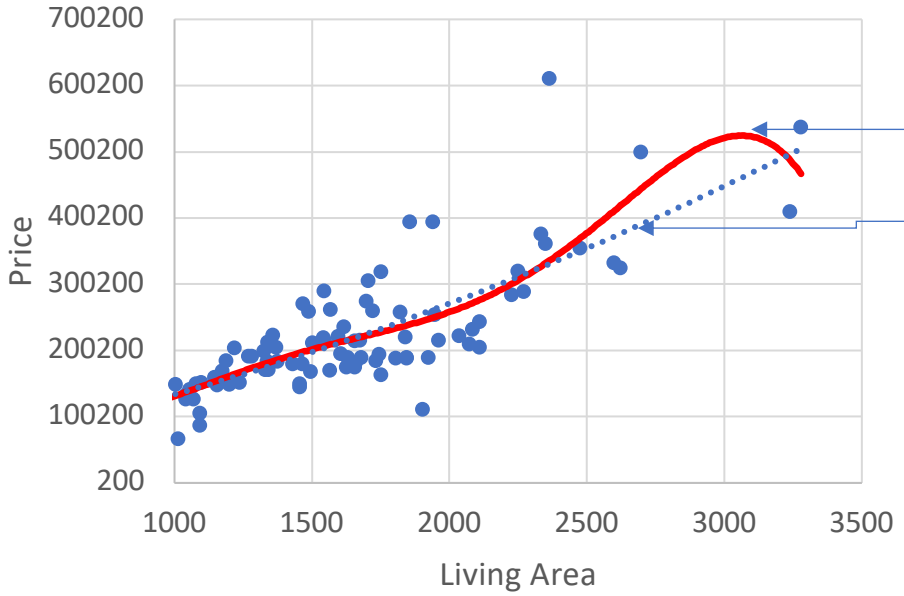
Area = front * side

We could inject Area or substitute the former features:

$$h_\theta(\mathbf{x}^{(\mathbf{i})}) = \theta_0 + \theta_1 \cdot area^{(i)}$$



Polynomial regression



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

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

$$\begin{aligned} h_\theta(\mathbf{x}^{(\mathbf{i})}) &= \theta_0 + \theta_1 \cdot x_1^{(i)} + \theta_2 \cdot x_2^{(i)} + \theta_2 \cdot x_3^{(i)} \\ &= \theta_0 + \theta_1 (area^{(i)}) + \theta_2 (area^{(i)})^2 + \theta_2 (area^{(i)})^3 \\ x_1^{(i)} &= (area) \\ x_2^{(i)} &= (area)^2 \\ x_3^{(i)} &= (area)^3 \end{aligned}$$

Politecnico di Bari

Linear regression

Gradient descent for multiple variables

Univariate Linear regression

Hypothesis:

$$h_\theta(\mathbf{x^{(i)}}) = \theta_0 + \theta_1 x^{(i)}$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

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

Goal:

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

Multivariate Linear regression

Hypothesis:

$$h_\theta(\mathbf{x^{(i)}}) = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + ... + \theta_n x_n^{(i)}$$

Parameters:

$$\theta_0, \theta_1, \theta_2, ..., \theta_n$$

Cost Function:

$$J(\theta) = J(\theta_0, \theta_1, \theta_2, ..., \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)})^2$$

Gradient descent:

repeat {

$$\theta_j := \theta_j - \alpha J(\theta_0, \theta_1, \theta_2, ..., \theta_n)$$

} (simultaneous updates of parameters)

Politecnico di Bari

Linear regression

Gradient descent: feature scaling

Feature scaling

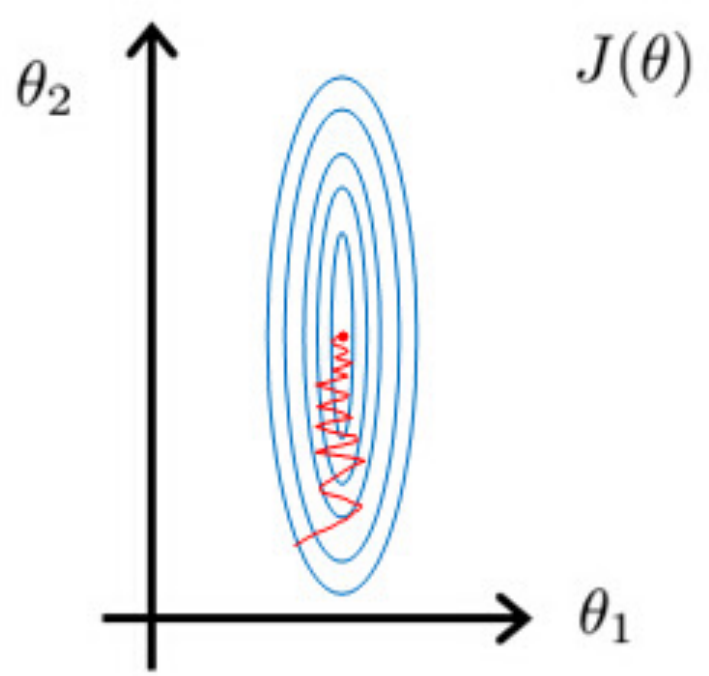
E.g. x_1= Living Area (0-2110)

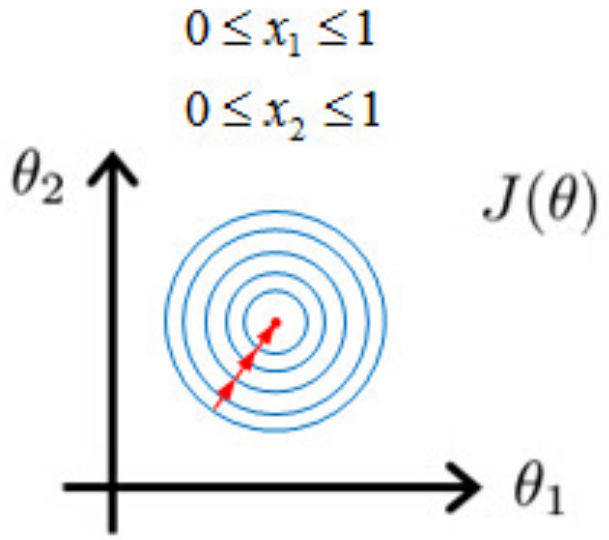
x_2= number of bedrooms (1-5)

Idea:

$$x_1 = \frac{area(feet^2)}{2110}$$

$$x_2 = \frac{numberofbedrooms}{5}$$





Min-max normalization

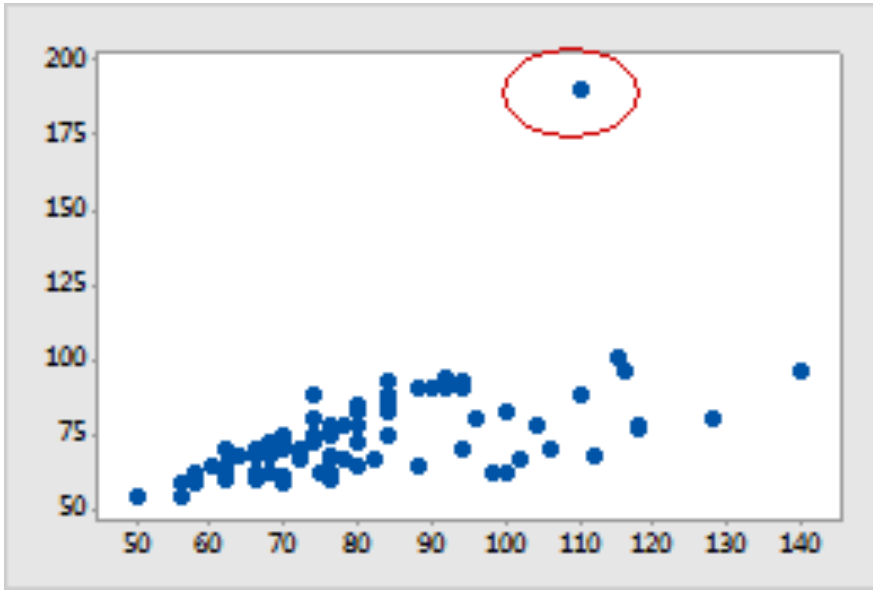
Forces the values in the interval [a,b].

This approach is negatively influenced by outliers.

You have to set up min and max values and in the prediction phase you could meet a new input greater than max.

$$x = \frac{x - min}{max - min} \cdot (b - a) + a$$

Ex: x=(1,2,3) -> [0,1] x=((x-1)/(3-1))*1 ->x=(0,0.5,1)



Z-score normalization

It produces a zero mean distribution through subtraction of the mean and dividing by the standard deviation.

This approach is less influenced by outliers and do not ask for min and max setting.

$$x = \frac{x - mean(x)}{std\_dev(x)}$$

Politecnico di Bari

Linear regression with one variable

Normal equations

A vectorial perspective

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)\mathbf{T}} \\ \mathbf{x}^{(2)\mathbf{T}} \\ \vdots \\ \mathbf{x}^{(i)\mathbf{T}} \\ \vdots \\ \mathbf{x}^{(m)\mathbf{T}} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(i)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

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

$$\nabla J(\theta) = \mathbf{X}^T\mathbf{X}\theta - \mathbf{X}^T\mathbf{y} = 0$$

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

Drawbacks

- Matrix Inverse (computationally expensive O(n^3))

- Non invertible matrix:
  - Linear dependent features
  - Training samples are not enough
  - There are too many features

Proof of Normal equations derivations

The original cost function is $J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(\mathbf{x}^{(\mathbf{i})}) - y^{(i)})^2.$

It is trivial to move to $J(\theta) = \frac{1}{2}(\mathbf{X}\theta - \mathbf{y})^T(\mathbf{X}\theta - \mathbf{y}).$

Because the latter can be explicitly re-written as

$$\mathbf{X}\theta - \mathbf{y} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} h_\theta(\mathbf{x}^{(\mathbf{1})}) - y^{(1)} \\ h_\theta(\mathbf{x}^{(\mathbf{2})}) - y^{(2)} \\ \vdots \\ h_\theta(\mathbf{x}^{(\mathbf{m})}) - y^{(m)} \end{bmatrix}$$

Proof of N.E. (cont.d)

It is easy to obtain the squared variant:

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

$$= \begin{bmatrix}(h_\theta(\mathbf{x^{(1)}}) - y^{(1)}) & \dots & (h_\theta(\mathbf{x^{(m)}}) - y^{(m)})\end{bmatrix} \begin{bmatrix} h_\theta(\mathbf{x^{(1)}}) - y^{(1)} \\ \vdots \\ h_\theta(\mathbf{x^{(m)}}) - y^{(m)} \end{bmatrix}$$

$$= \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)})^2.$$

Proof of N.E. (cont.d)

We want to find the minimum, the point in which the gradient has null value:

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_k} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_k} = \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)}) x_k^{(i)}$$

$$= \begin{bmatrix} (h_\theta(\mathbf{x^{(1)}}) - y^{(1)}) & \dots & (h_\theta(\mathbf{x^{(m)}}) - y^{(m)}) \end{bmatrix} \begin{bmatrix} x_k^{(1)} \\ \vdots \\ x_k^{(m)} \end{bmatrix}$$

Proof of N.E. (cont.d)

We can complete the gradient equation and re-write it as a matrix product.

$$\nabla J(\theta) = \begin{bmatrix} \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)}) \\ \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)}) x_1^{(i)} \\ \vdots \\ \sum_{i=1}^{m} (h_\theta(\mathbf{x^{(i)}}) - y^{(i)}) x_n^{(i)} \end{bmatrix}$$

$$\begin{aligned} \nabla J(\theta) &= \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} h_\theta(\mathbf{x^{(1)}}) - y^{(1)} \\ h_\theta(\mathbf{x^{(2)}}) - y^{(2)} \\ \vdots \\ h_\theta(\mathbf{x^{(m)}}) - y^{(m)} \end{bmatrix} \\ &= \mathbf{X}^T(\mathbf{X}\theta - \mathbf{y}) \\ &= \mathbf{X}^T\mathbf{X}\theta - \mathbf{X}^T\mathbf{y} \end{aligned}$$

$$\begin{aligned} \nabla J(\theta) &= \mathbf{X}^T\mathbf{X}\theta - \mathbf{X}^T\mathbf{y} = 0 \\ \hat{\theta} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \end{aligned}$$

Politecnico di Bari

Linear regression with one variable

Probabilistic interpretation of linear regression

Proof Probabilistic interpretation

Toolbox:

$(\mathbf{x}^{(\mathbf{i})}, y^{(i)})$ m training set istances

$h(\mathbf{x}) = \theta^T \mathbf{x}$ hypothesis that approximates the relation between features vectors and output

Output can be computed as a predicted value plus an error

$$y^{(i)} = \theta^T \mathbf{x}^{(\mathbf{i})} + e^{(i)}. \quad (1)$$

Proof Probabilistic interpretation (cont.d)

Let us assume samples are independent so the errors, that we can model as an exponential random variable, and it is assumed that the variables show all a normal distribution, with 0 mean and variance σ^2

$$p(e^{(i)}) = \mathcal{N}(0, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(e^{(i)})^2}{2\sigma^2}} \qquad i = 1 \dots m. \qquad (2)$$

Proof Probabilistic interpretation (cont.d)

Now we fix the expressiveness of a model (the hypothesis) and so the parameters. This implies that the probability that the input produces the output has the same probability of the error. Substituting (1) in (2) we get:

$$e^{(i)} = y^{(i)} - \theta^T \mathbf{x}^{(\mathbf{i})}.$$

$$p(y^{(i)}|\mathbf{x}^{(\mathbf{i})};\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^{(i)}-\theta^T \mathbf{x}^{(\mathbf{i})})^2}{2\sigma^2}} \qquad i = 1 \ldots m.$$

Proof Probabilistic interpretation (cont.d)

We want to find a function (of the parameters) that, given the input, returns the output for each training case. This probability is named «Likelihood»:

$$L(\theta) = L(\theta; \mathbf{X}; \mathbf{y}) = p(\mathbf{y}|\mathbf{X}; \theta)$$

If the independence assumption is given (as we did before) the probability of the joint event is:

$$L(\theta) = p(\mathbf{y}|X; \theta) = \prod_{i=1}^{m} p(y^{(i)}|\mathbf{x^{(i)}}; \theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^{(i)} - \theta^T \mathbf{x^{(i)}})^2}{2\sigma^2}}$$

Proof Probabilistic interpretation (cont.d)

We look for the parameters vector that maximizes the Likelihood function.

We look for the thetas that has the maximum probability to return the output given the input.

This is named «maximum likelihood criterion»:

$$\hat{\theta} = \underset{\theta}{argmax} L(\theta)$$

$$\hat{\theta} = \underset{\theta}{argmax} L(\theta) = \underset{\theta}{argmax} \log L(\theta) = \underset{\theta}{argmax} \quad l(\theta)$$

$$with \quad \log L(\theta) = l(\theta)$$

Proof Probabilistic interpretation (cont.d)

Logarithm operator let us to transform the productory in a summatory

$$
\begin{aligned}
l(\theta) &= \log \prod_{i=1}^{m} p(y^{(i)}|\mathbf{x^{(i)}};\theta) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^{(i)}-\theta^T\mathbf{x^{(i)}})^2}{2\sigma^2}} \\
&= \sum_{i=1}^{m} \left(\log \frac{1}{\sqrt{2\pi}\sigma} + \log e^{-\frac{(y^{(i)}-\theta^T\mathbf{x^{(i)}})^2}{2\sigma^2}} \right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \sum_{i=1}^{m} \frac{(y^{(i)}-\theta^T\mathbf{x^{(i)}})^2}{2\sigma^2} \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)}-\theta^T\mathbf{x^{(i)}})^2
\end{aligned}
$$

Proof Probabilistic interpretation (cont.d)

Moving back to the maximization problem:

$$\begin{aligned}
\max_{\theta} l(\theta) &= \max_{\theta} \left(m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)} - \theta^T \mathbf{x^{(i)}})^2 \right) \\
&= \max_{\theta} \left(-\frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)} - \theta^T \mathbf{x^{(i)}})^2 \right) \\
&= \max_{\theta} \left(-\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T \mathbf{x^{(i)}})^2 \right) \\
&= \min_{\theta} \left(\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T \mathbf{x^{(i)}})^2 \right).
\end{aligned}$$

Proof Probabilistic interpretation (cont.d)

We can compare the latter with the error in (1):

$$\min_{\theta} \left(\frac{1}{2m} \sum_{i=1}^{m} (h(\mathbf{x^{(i)}}) - y^{(i)})^2 \right) = \min_{\theta} \left(\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T \mathbf{x^{(i)}})^2 \right)$$

These problems are identical but a regularization costant that has no influence in the minimization problem.

Solving the least squares problem in the linear regression means solving the problem of finding the best parameters that produce the output given the input