Linear Regression

Fundamentals of Data Science
20, 23, 27, 30 October, 3, 6 November 2023
Prof. Fabio Galasso

SAPIENZA
UNIVERSITÀ DI ROMA

ALVINN’97

Outline

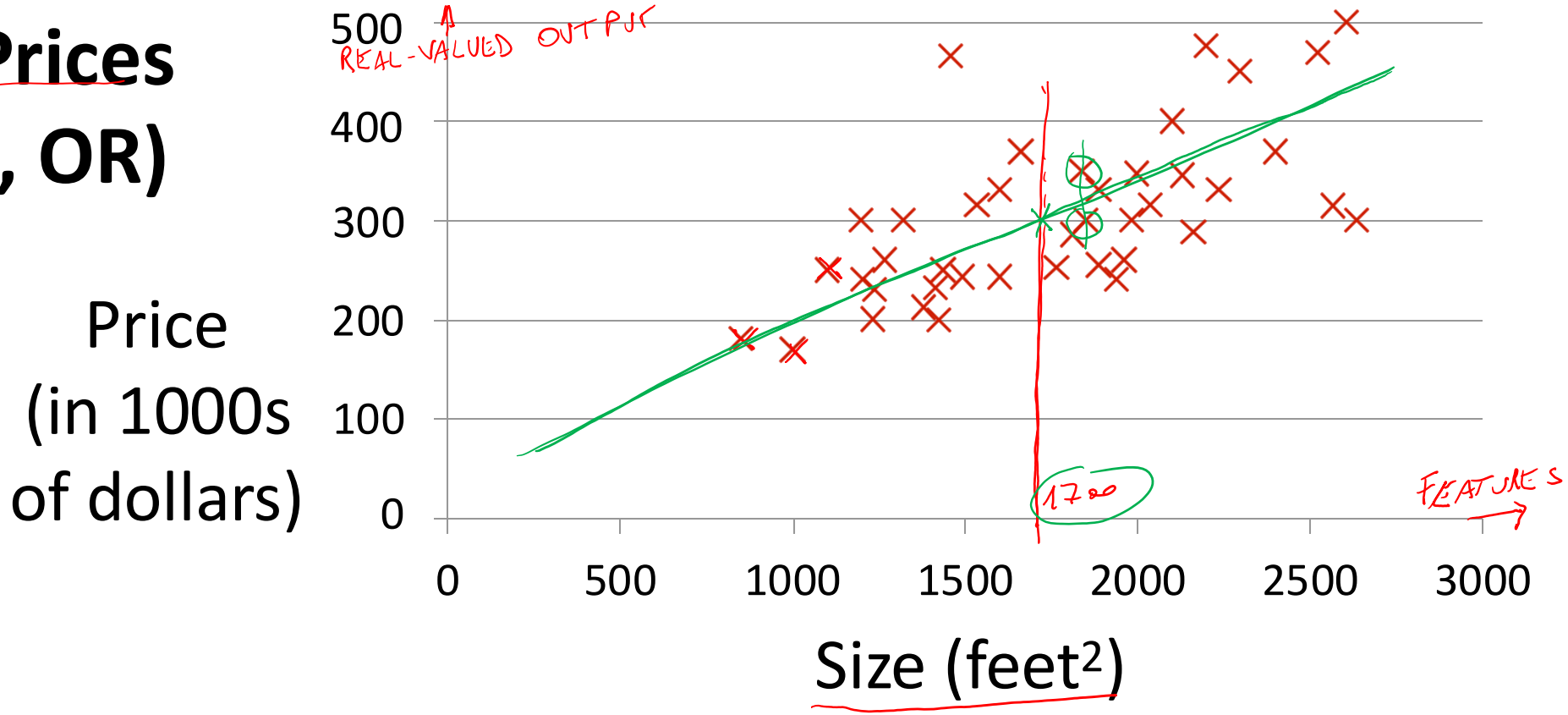
- Linear regression
  - One or multiple variables
  - Cost function
- Gradient descent, incl. batch/stochastic GD
- Normal equation
- MSE and Correlation
- Locally-Weighted Regression
- Probabilistic Interpretation of Least Squares

Outline

- Linear regression
  - One or multiple variables
  - Cost function
- Gradient descent, incl. batch/stochastic GD
- Normal equation
- MSE and Correlation
- Locally-Weighted Regression
- Probabilistic Interpretation of Least Squares

Linear Regression

Housing Prices (Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real--valued output

CLASSIFICATION : DISCRETE-VALUED OUTPUT

Training set of housing prices (Portland, OR)

	Size in feet² (x)	Price ($) in 1000's (y)
1)	2104	460
2)	1416	232
3)	1534	315
4)	852	178
	…	…

Notation:

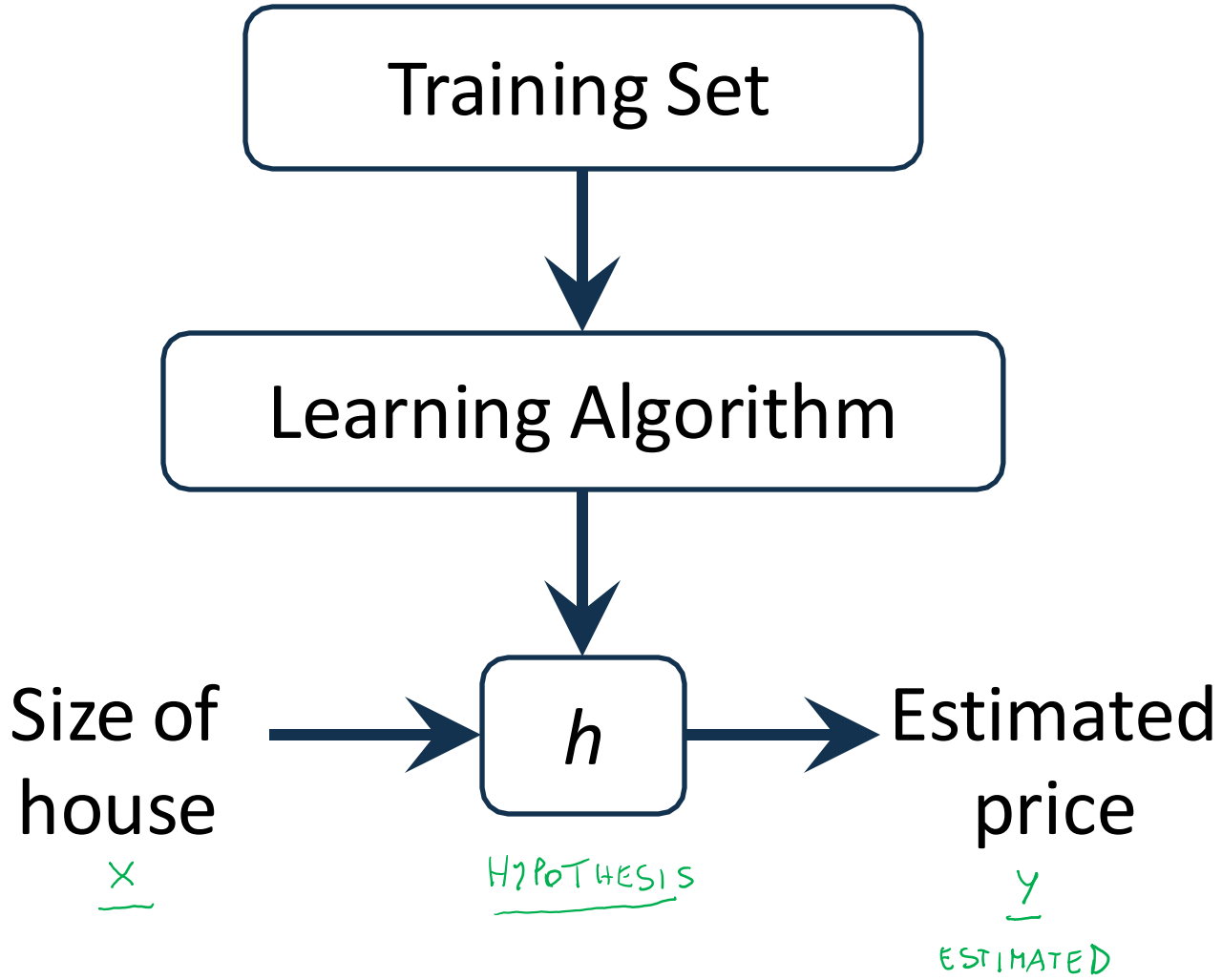
- **m** = Number of training examples
- **x**'s = "input" variable / features
- **y**'s = "output" variable / "target" variable

$x^{(2)} = 1416$

$y^{(2)} = 232$

(x, y) TRAINING EXAMPLE

$(x^{(i)}, y^{(i)})$ i-th TRAINING EXAMPLE

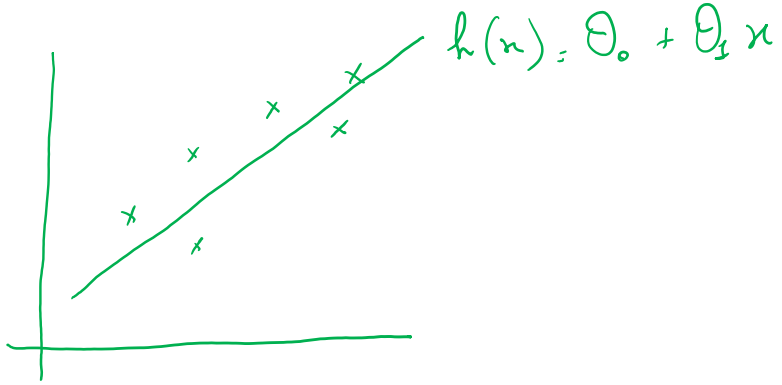


h maps from x's to y's

How do we represent *h* ?

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

SHORTHAND $h(x)$



Linear regression with one variable. (x)

Univariate linear regression.

Multiple features (variables)

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

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

Multiple features (variables)

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
…	…	…	…	…

m TRAINING EXAMPLES

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.

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_3^{(2)} = 2$$

Hypothesis:

With one variable: $h_\theta(x) = \theta_0 + \theta_1 x$

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

$h_\theta(x) = 100 + 0.1 x_1 + 0.6 x_2 + 0.07 x_3 + 10 x_4$

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

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

$x_0^{(i)} = 1$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{bmatrix} \in \mathbb{R}^{m+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \in \mathbb{R}^{m+1}$$

$$\underbrace{\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_m \end{bmatrix}}_{\theta^T} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_2 \end{bmatrix}}_{x}$$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m$$

$$= \theta^T x$$

Multivariate linear regression

Linear Regression:
Cost Function

Training Set

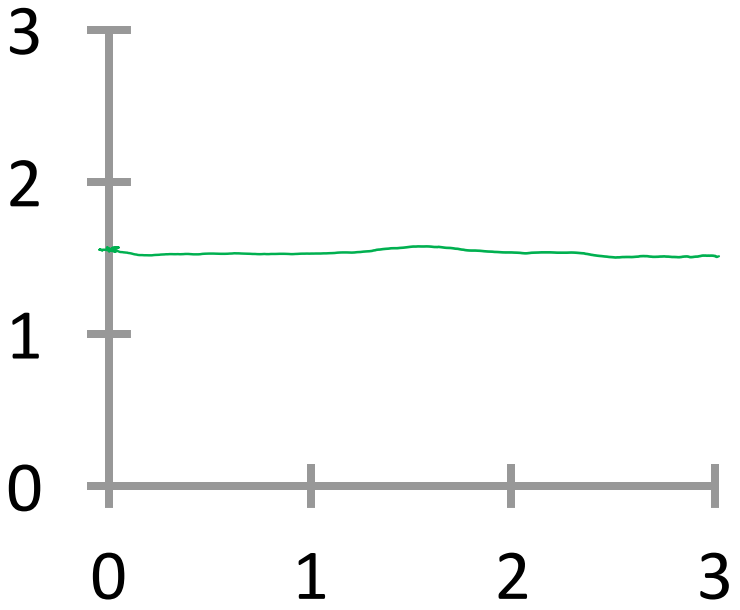
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

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

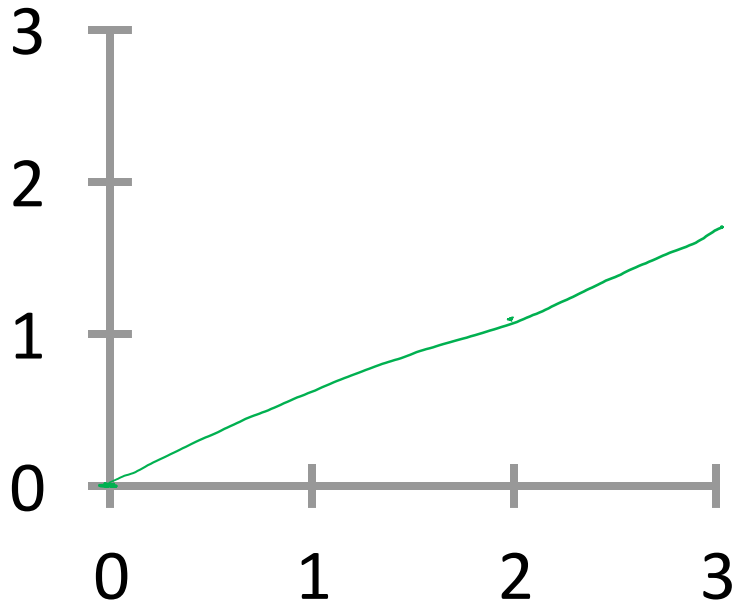
θ_i's: Parameters

How to choose θ_i's ?

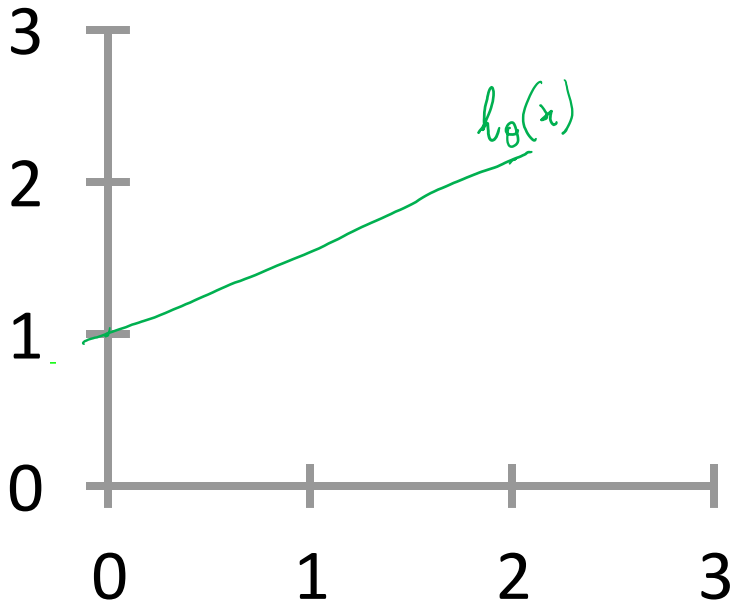
$$h_\theta(x) = \theta_0 + \theta_1 x$$



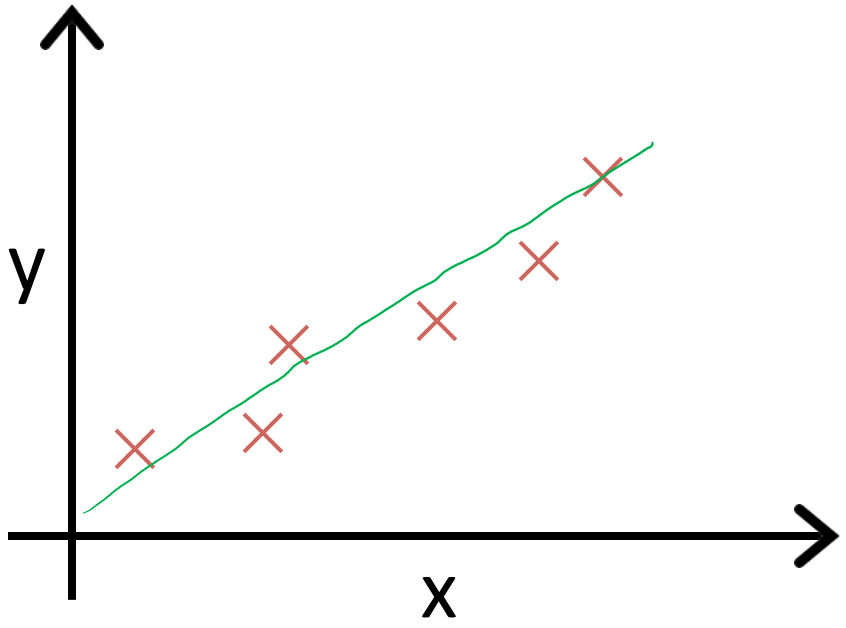
$\theta_0 = 1.5$
$\theta_1 = 0$



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



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



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

TRAINING EXAMPLES

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

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

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

COST FUNCTION

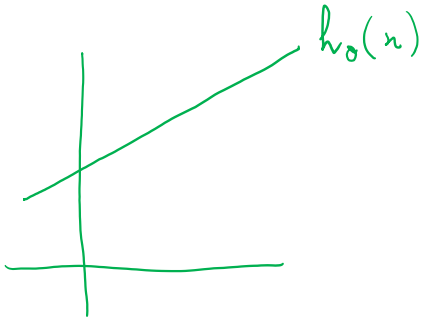
Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

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

TRAINING EXAMPLES

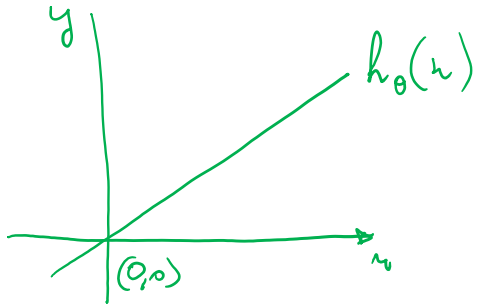
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Simplified

$\theta_0 = 0$

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

θ_1



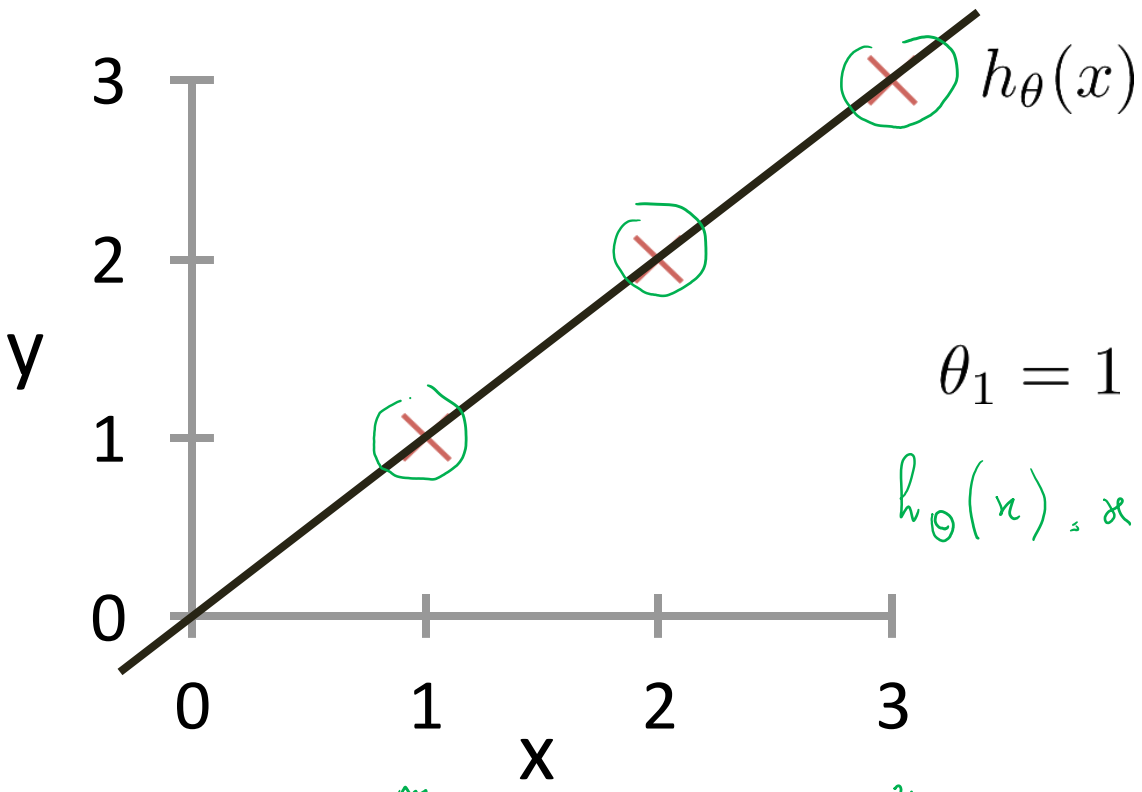
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$\theta_1 x^{(i)}$

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

$h_\theta(x)$

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

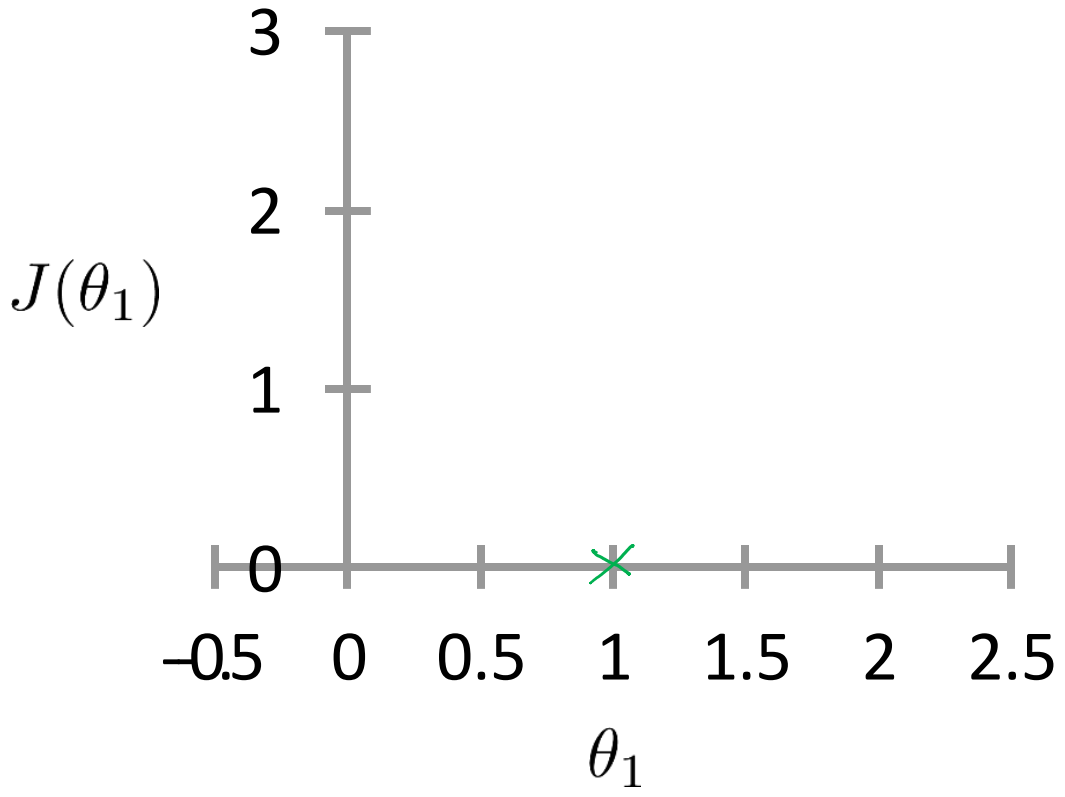


$$J(\theta_1) = \frac{1}{2m} \sum^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{1}{2 \cdot 3} \left(0^2 + 0^2 + 0^2 \right) = 0$$

$J(\theta_1)$

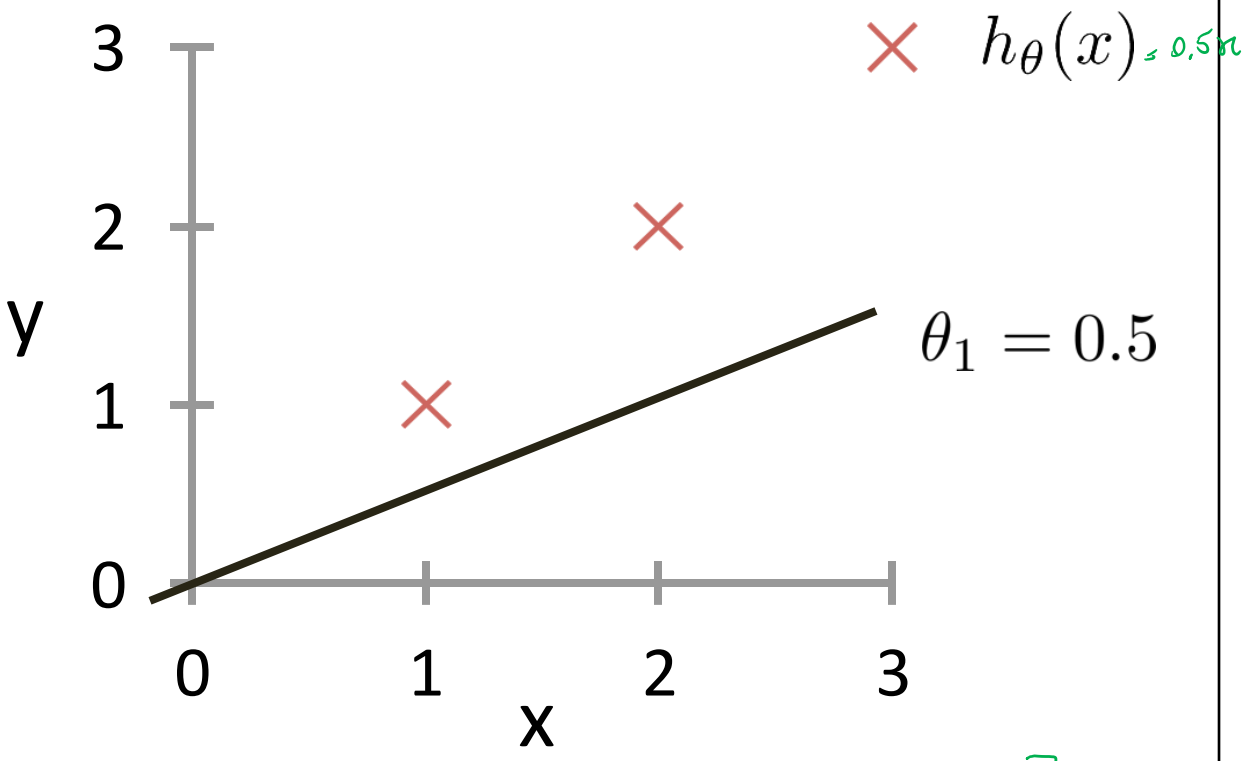
(function of the parameter θ_1)



$J(1) = 0$

$h_\theta(x)$

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

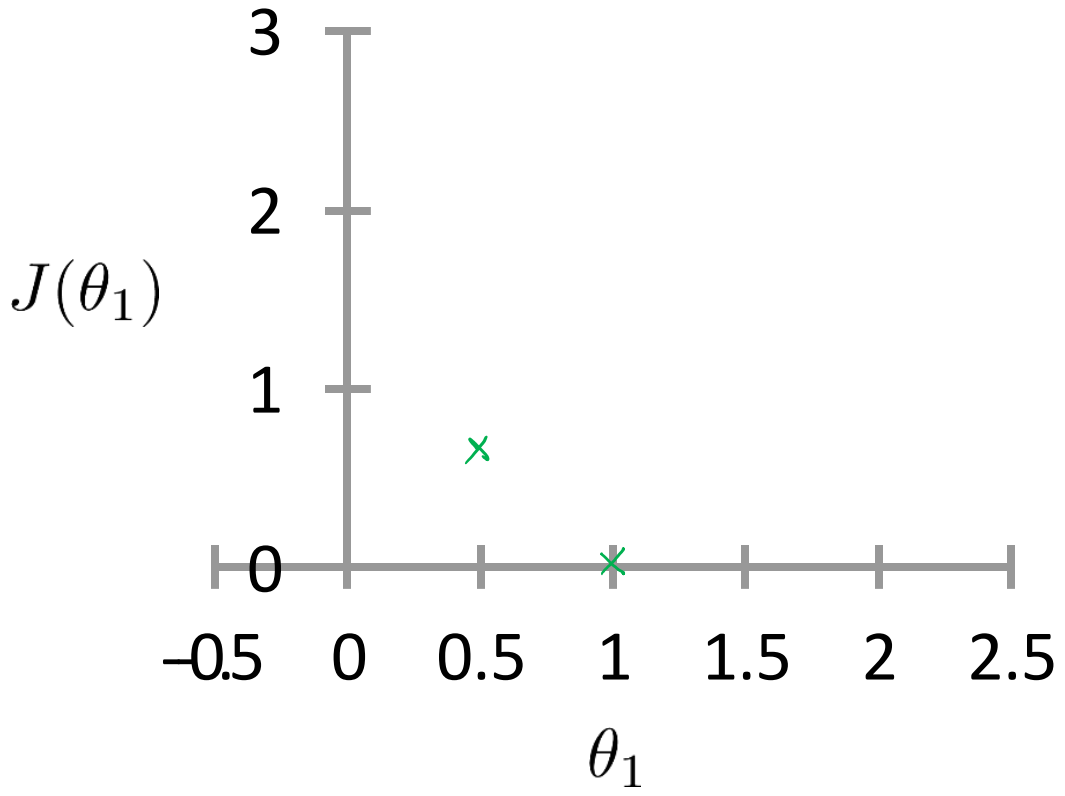


$$J(0.5) = \frac{1}{2\times 3}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

$$= \frac{1}{2\times 3}\left(\frac{1}{4} + 1 + \frac{9}{4}\right) \simeq 0.58$$

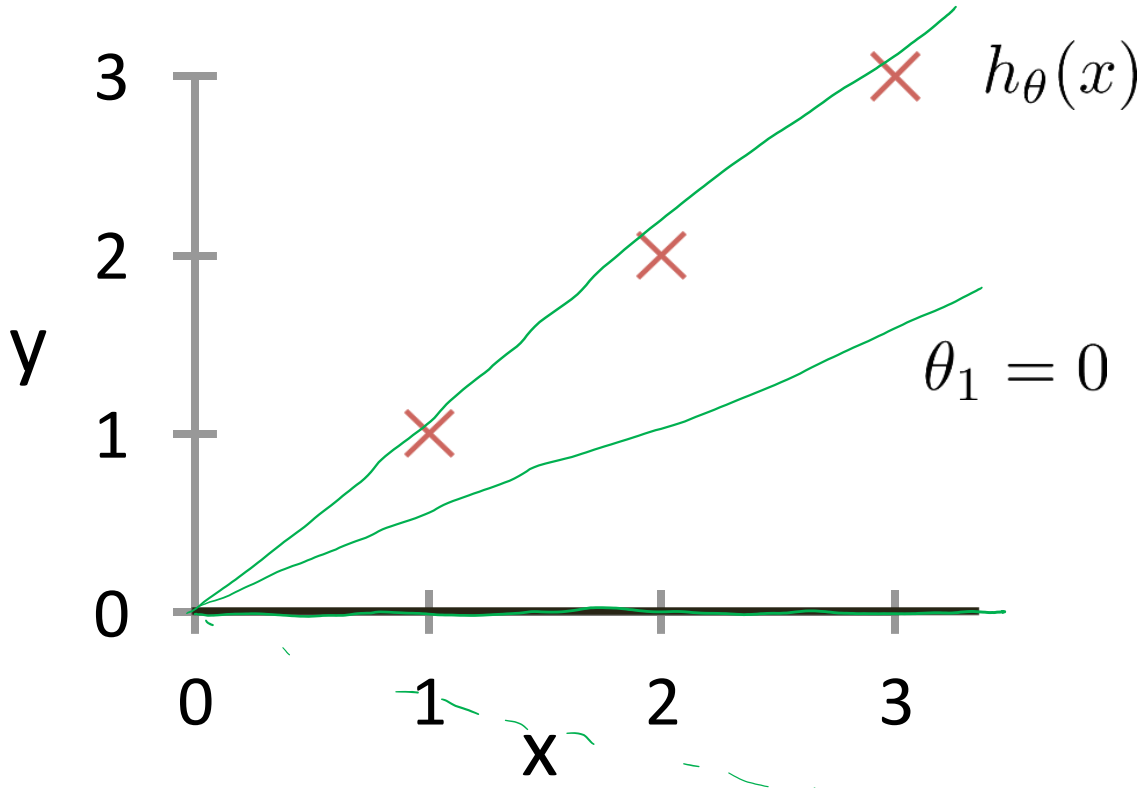
$J(\theta_1)$

(function of the parameter θ_1)



$h_\theta(x)$

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

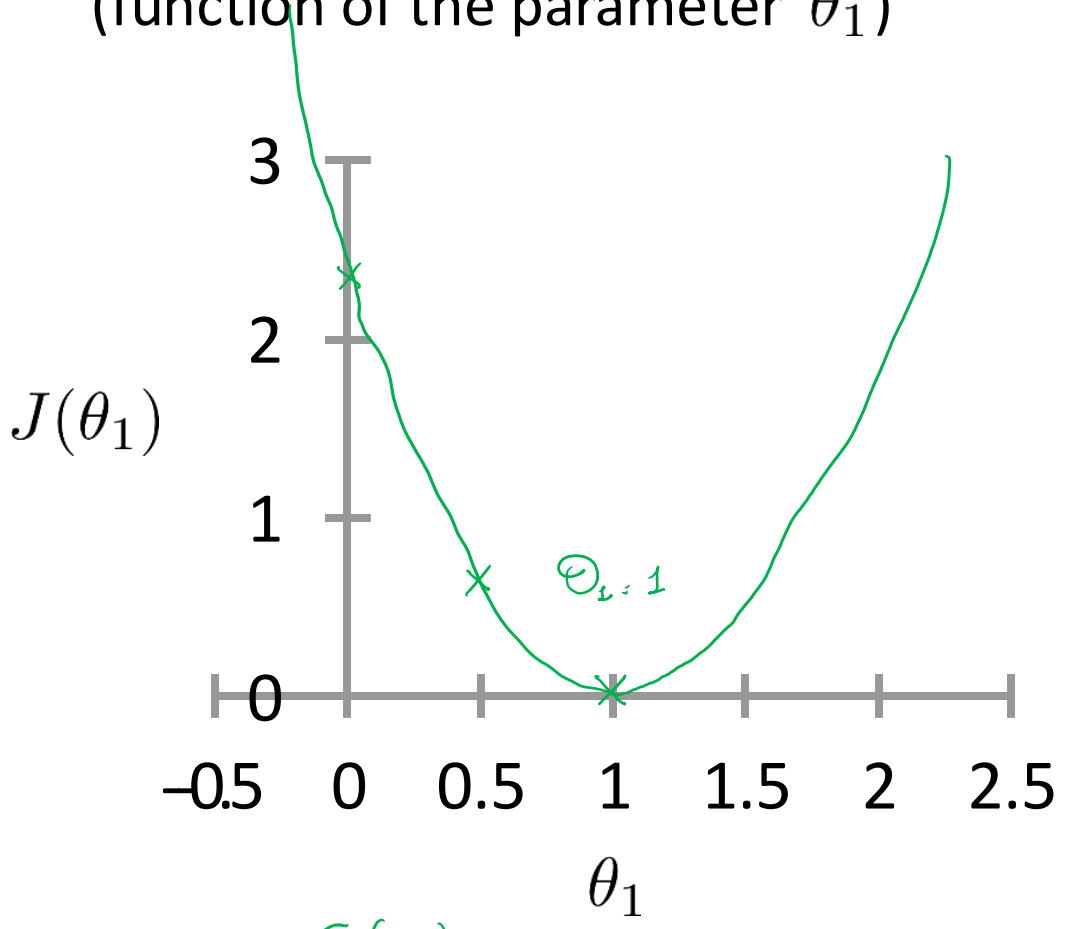


$J(0) = \frac{1}{6}(1^2 + 2^2 + 3^3) \approx 2.3$

$J(-0.5)$

$J(\theta_1)$

(function of the parameter θ_1)



minimize $J(\theta_1)$

θ_1

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

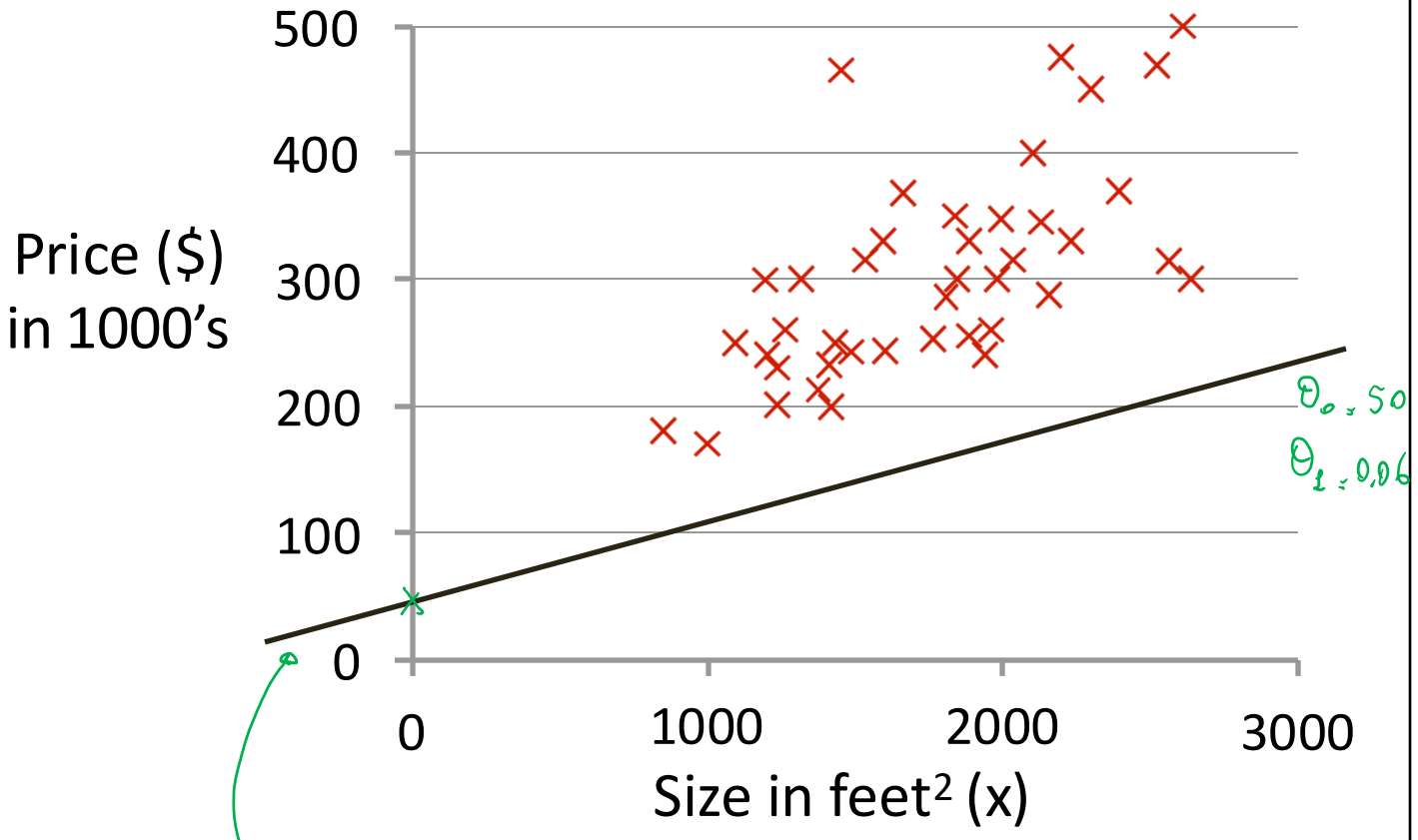
Parameters: θ_0, θ_1

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

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

$h_\theta(x)$

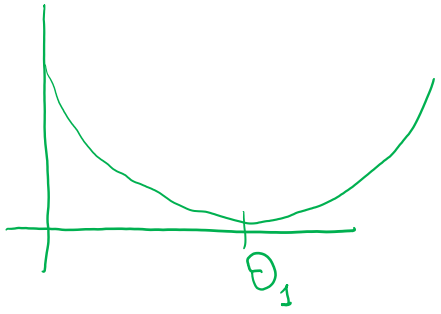
(for fixed θ_0, θ_1, this is a function of x)



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

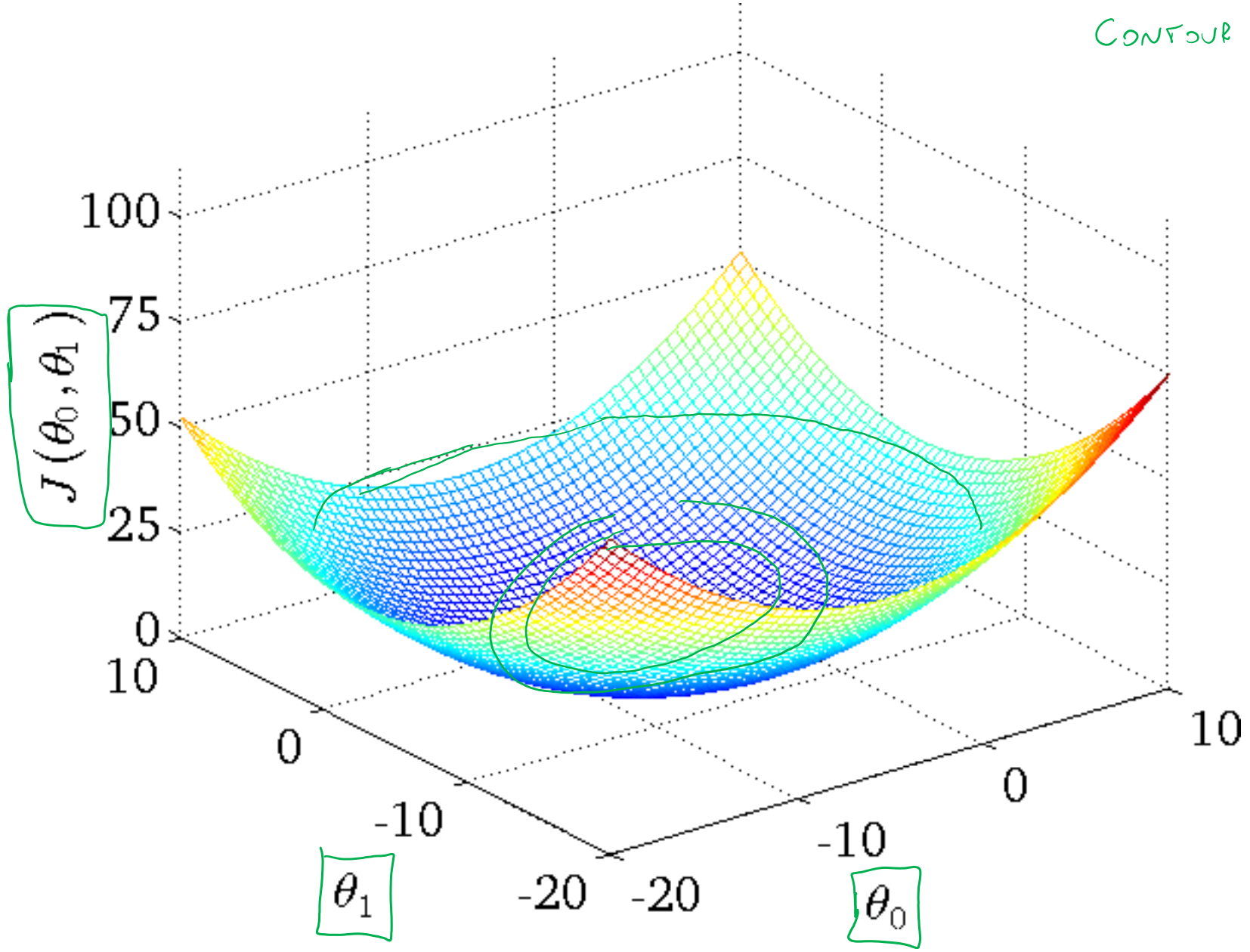
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



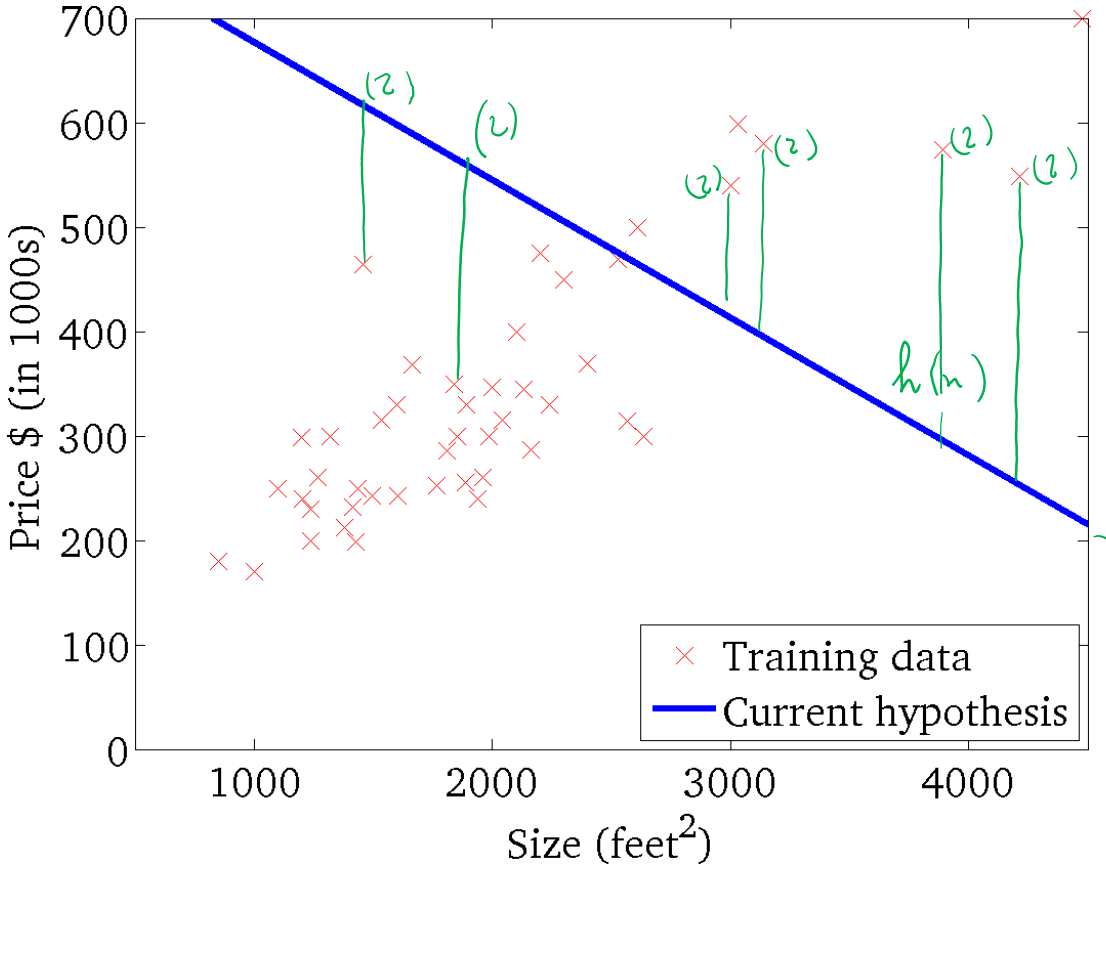
(θ_0, θ_1)

Contour Plots



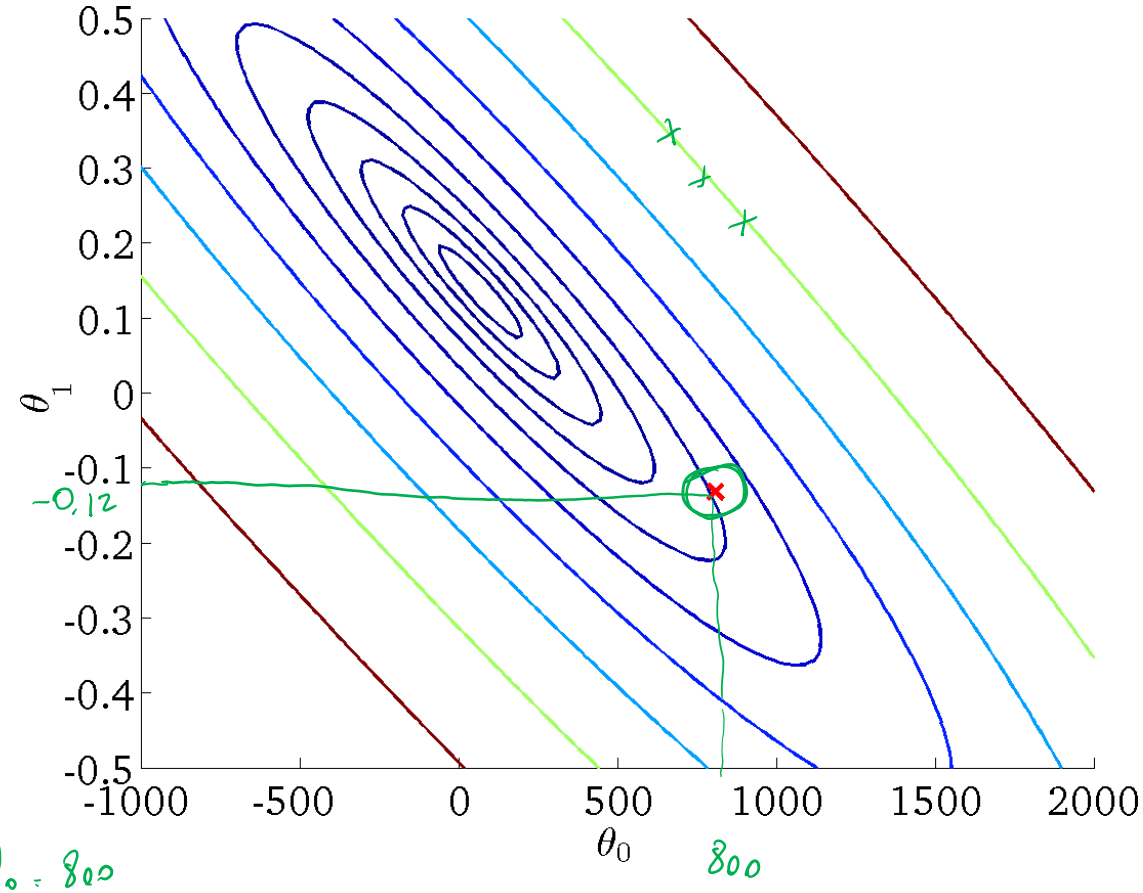
$h_\theta(x)$

(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)

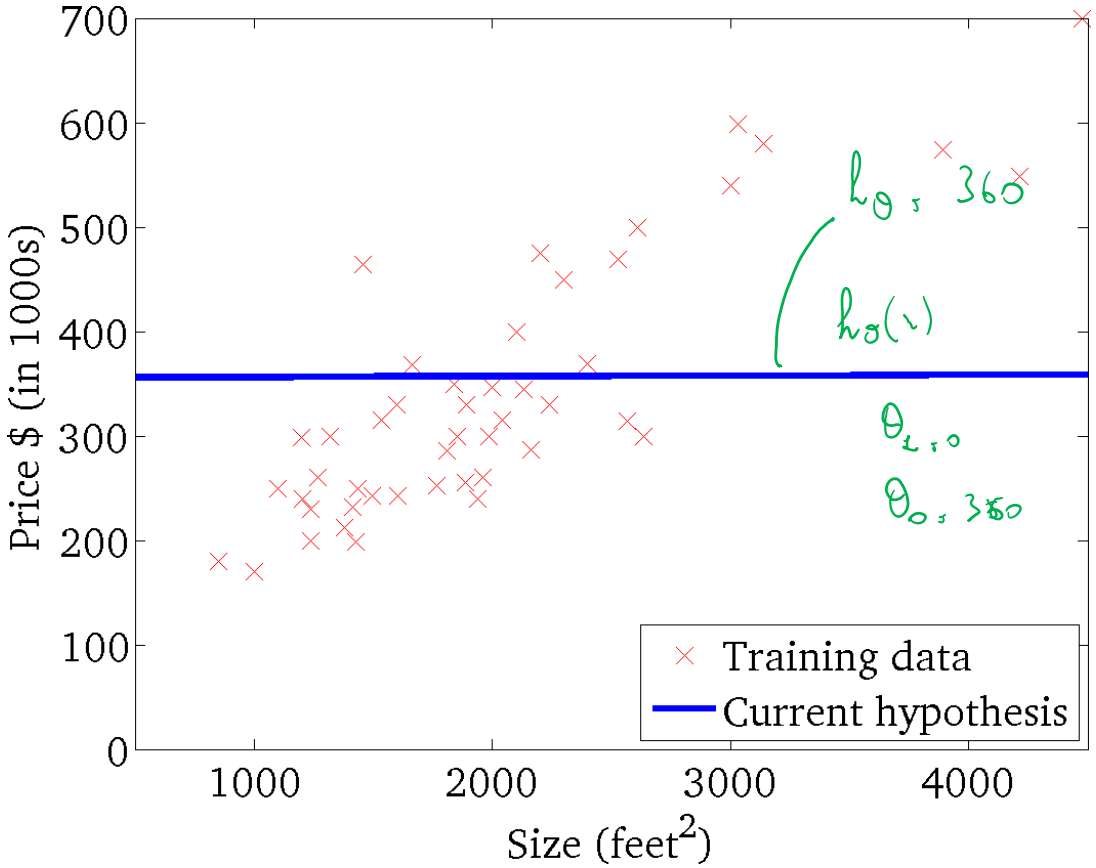


$\theta_0 = 800$

$\theta_1 = -0.12$

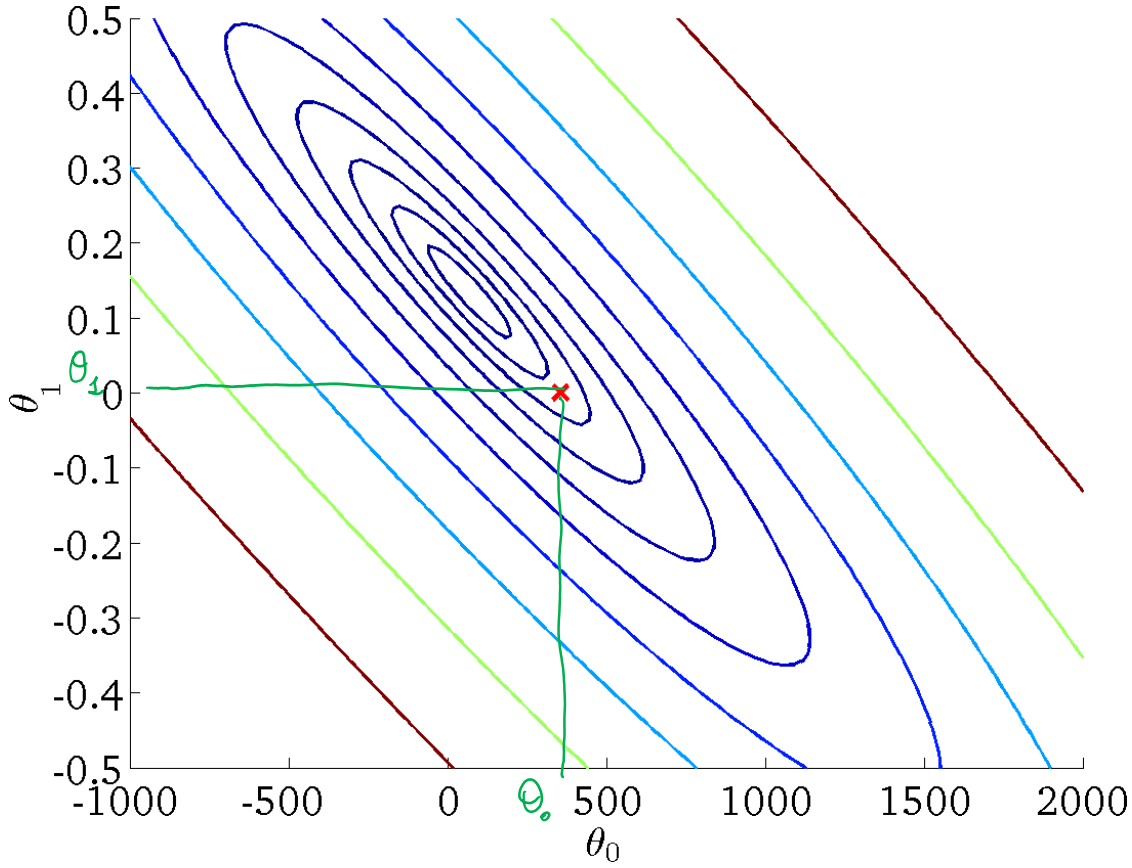
$h_\theta(x)$

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



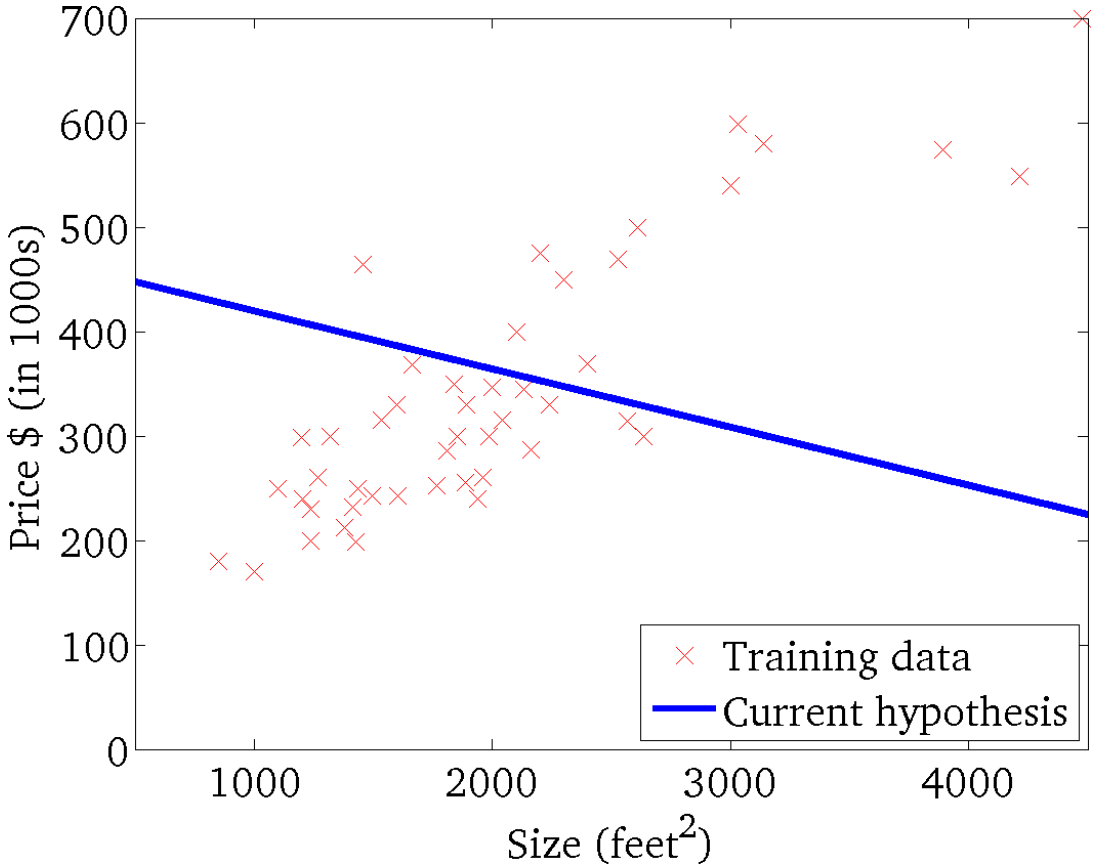
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



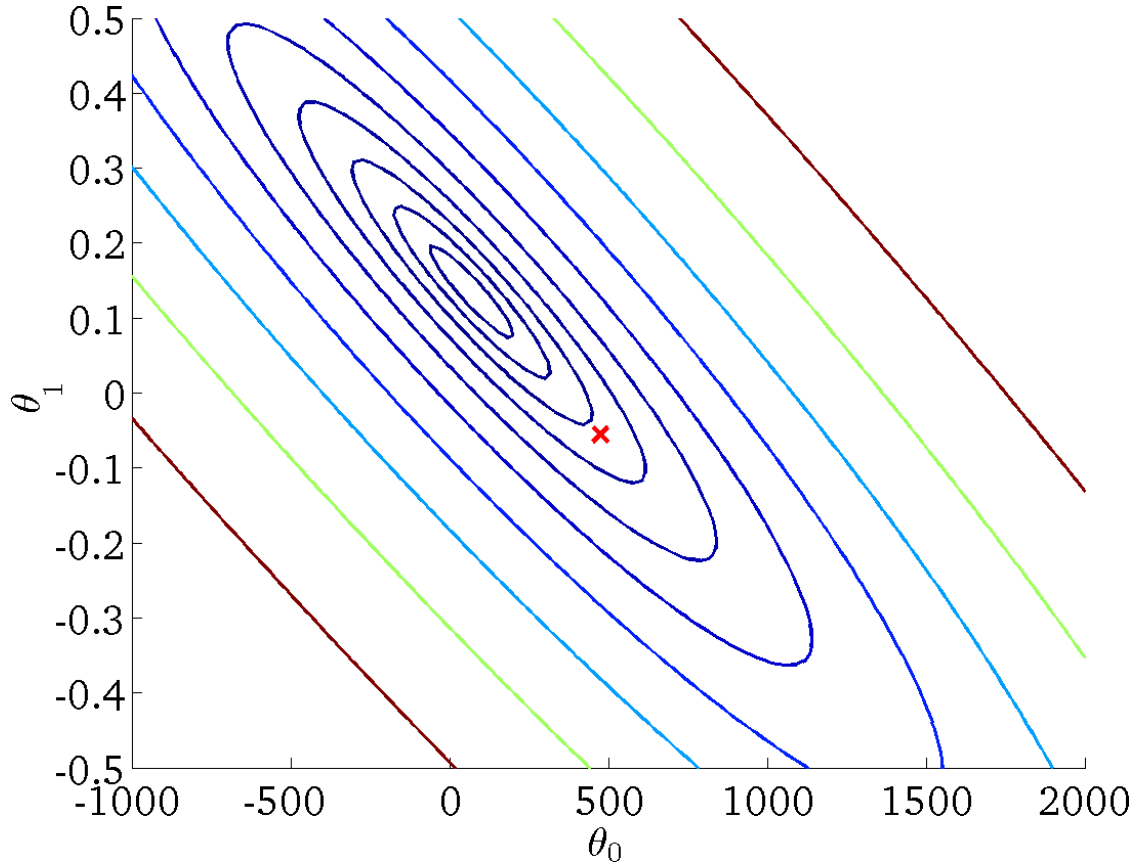
$h_\theta(x)$

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



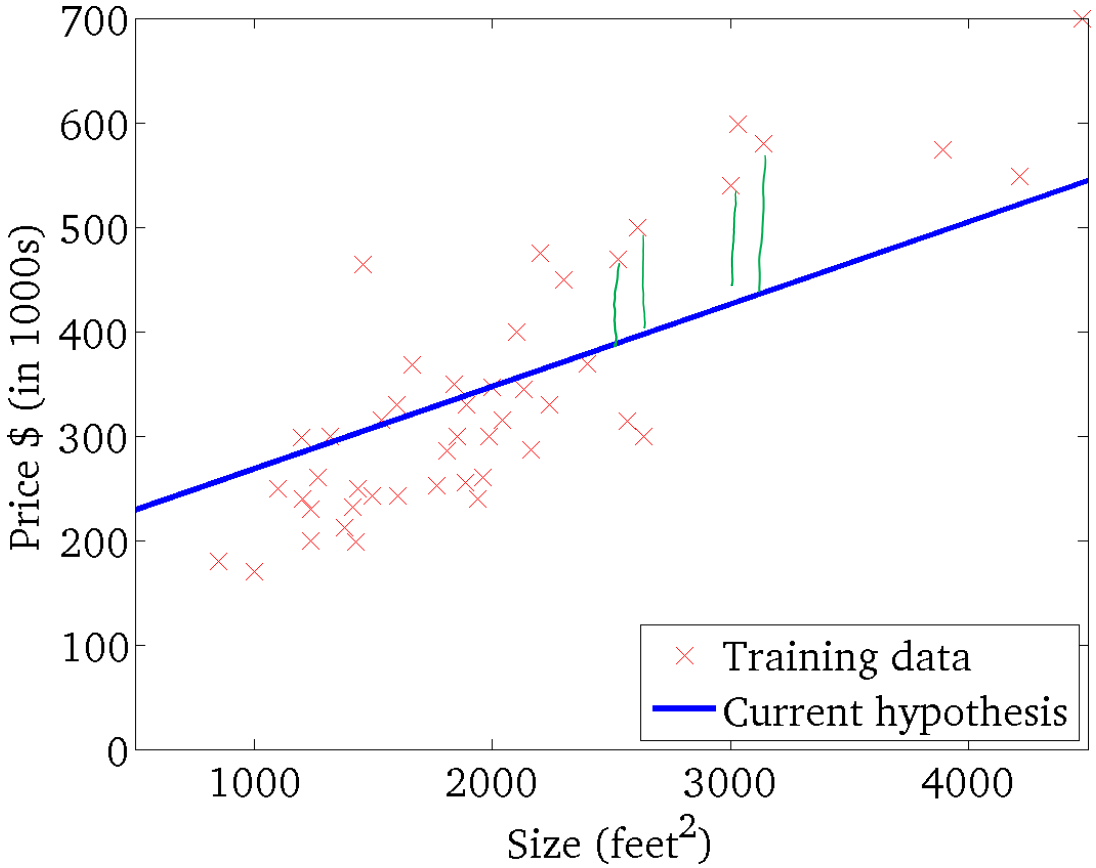
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



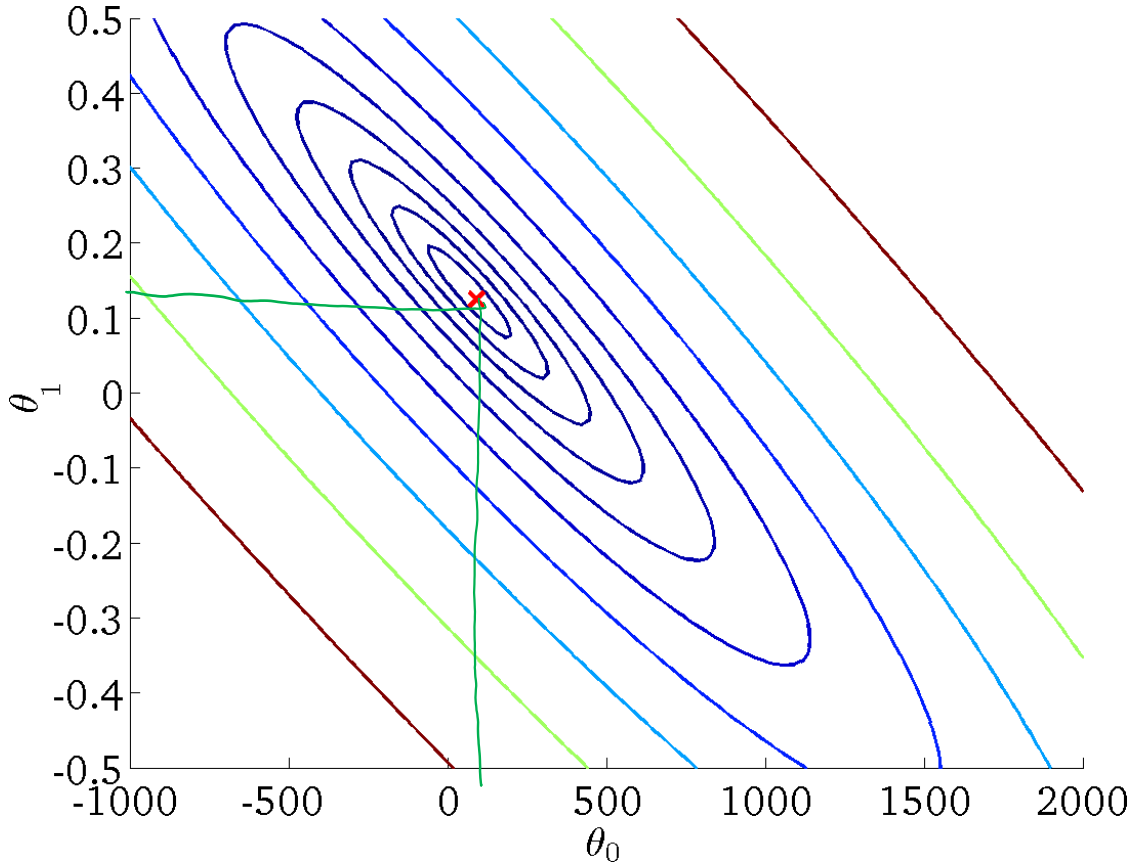
$h_\theta(x)$

(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)



Outline

- Linear regression
  - One or multiple variables
  - Cost function
- Gradient descent, incl. batch/stochastic GD
- Normal equation
- MSE and Correlation
- Locally-Weighted Regression
- Probabilistic Interpretation of Least Squares

Gradient Descent

n FEATURES

Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2 \ldots \theta_n)$

Want $\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$ $\min_{\theta_0,\theta_1 \ldots \theta_n} J(\theta_0, \theta_1, \ldots \theta_n)$

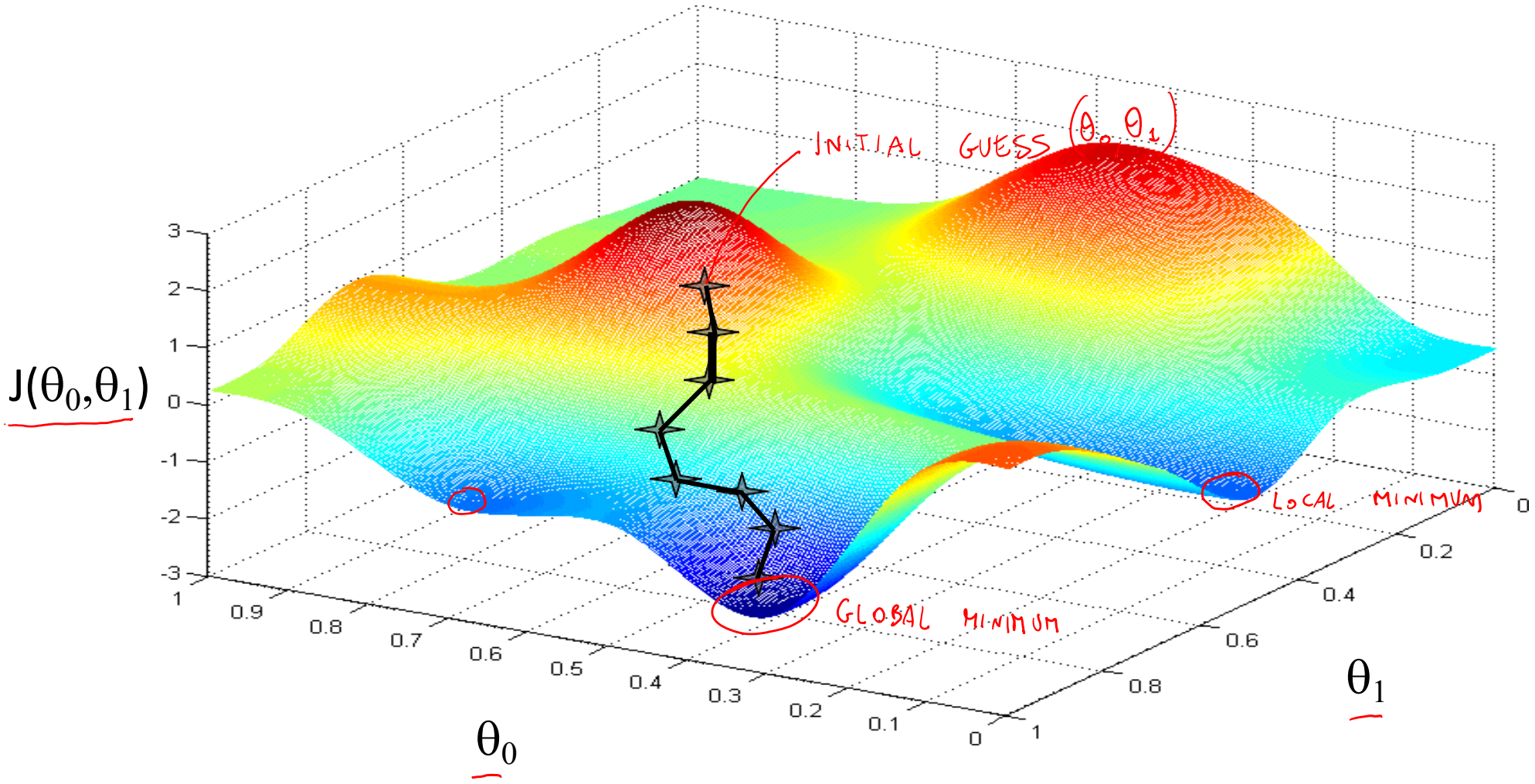
$y = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \ldots = \theta^T x$

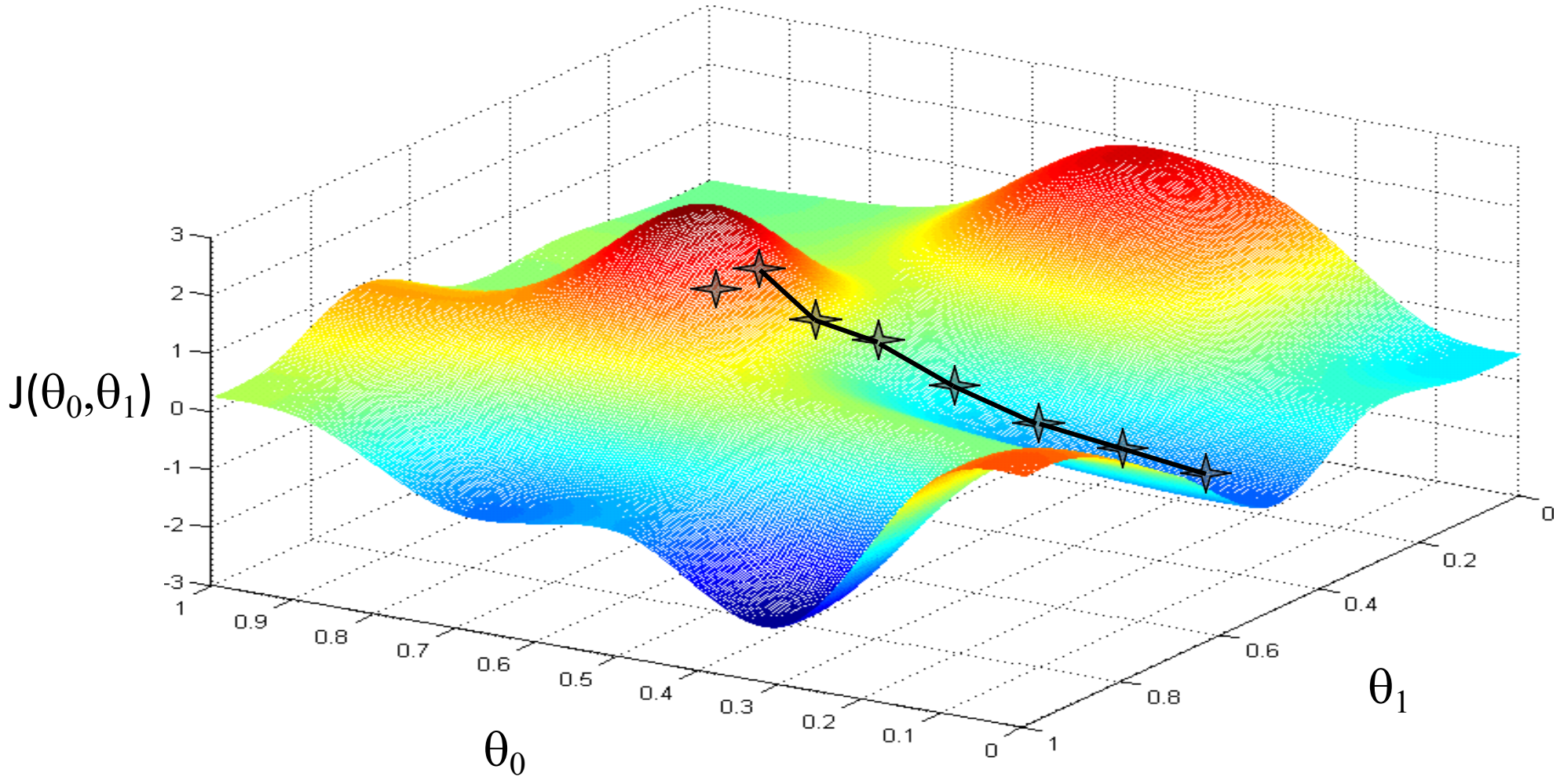
($x_0 = 1$) $=$

Outline:

- Start with some θ_0, θ_1 (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

  until we hopefully end up at a minimum


$J(\theta_0,\theta_1)$
INITIAL GUESS (θ_0, θ_1)
LOCAL MINIMUM
GLOBAL MINIMUM
θ_0
θ_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
0
0.2
0.4
0.6
0.8
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 {

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

}

LEARNING RATE

SIMULTANEOUSLY

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

$a := b$ ASSIGNMENT

$a = b$ ASSERTION

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 algorithm

repeat until convergence {

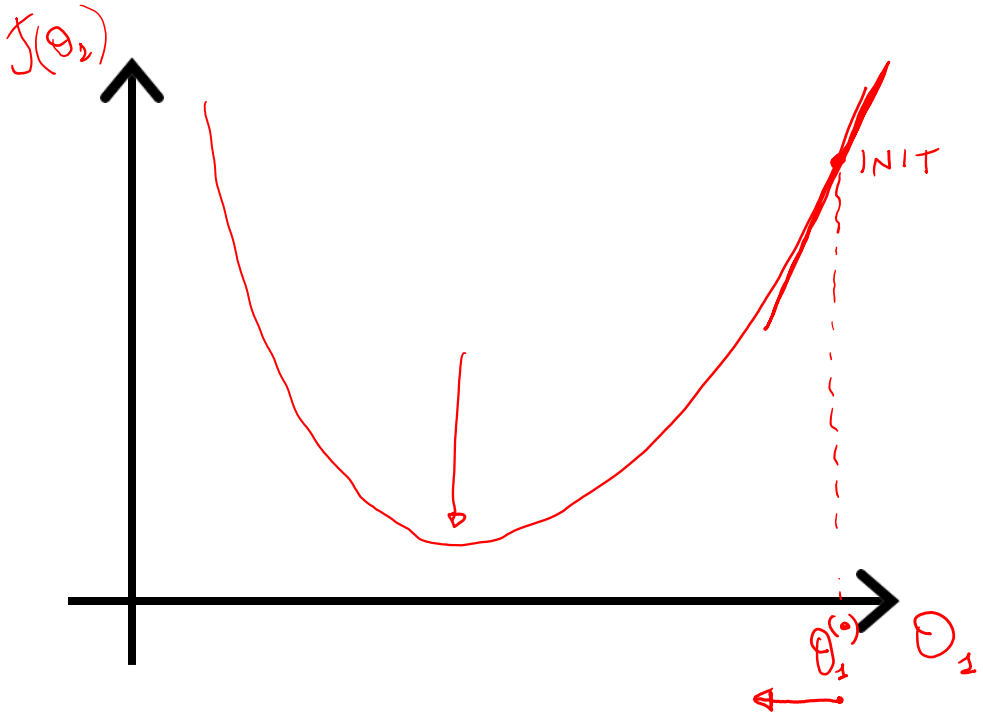
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

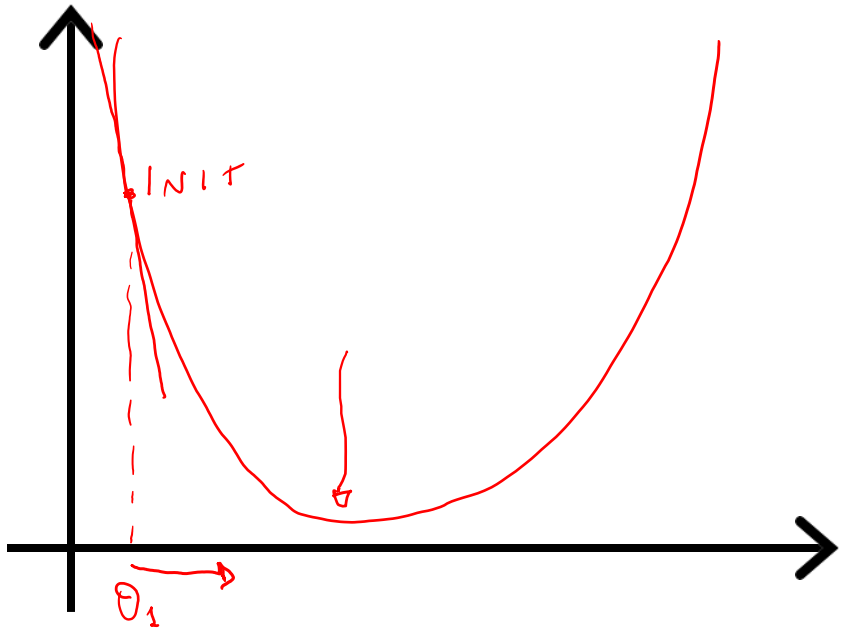
} (simultaneously update $j = 0$ and $j = 1$)

LR

DERIVATIVE

$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$





$\theta_1 \in \mathbb{R}$

$$\theta_1^{(t+1)} := \theta_1^{(t)} - \alpha \underbrace{\frac{d}{d\theta_1} J(\theta_1^{(t)})}_{\geq 0} \qquad \frac{d}{d\theta_1} = \frac{\partial}{\partial \theta_1}$$

$$\theta_1 := \theta_1 - \alpha \,(\text{POSITIVE NUMBER})$$

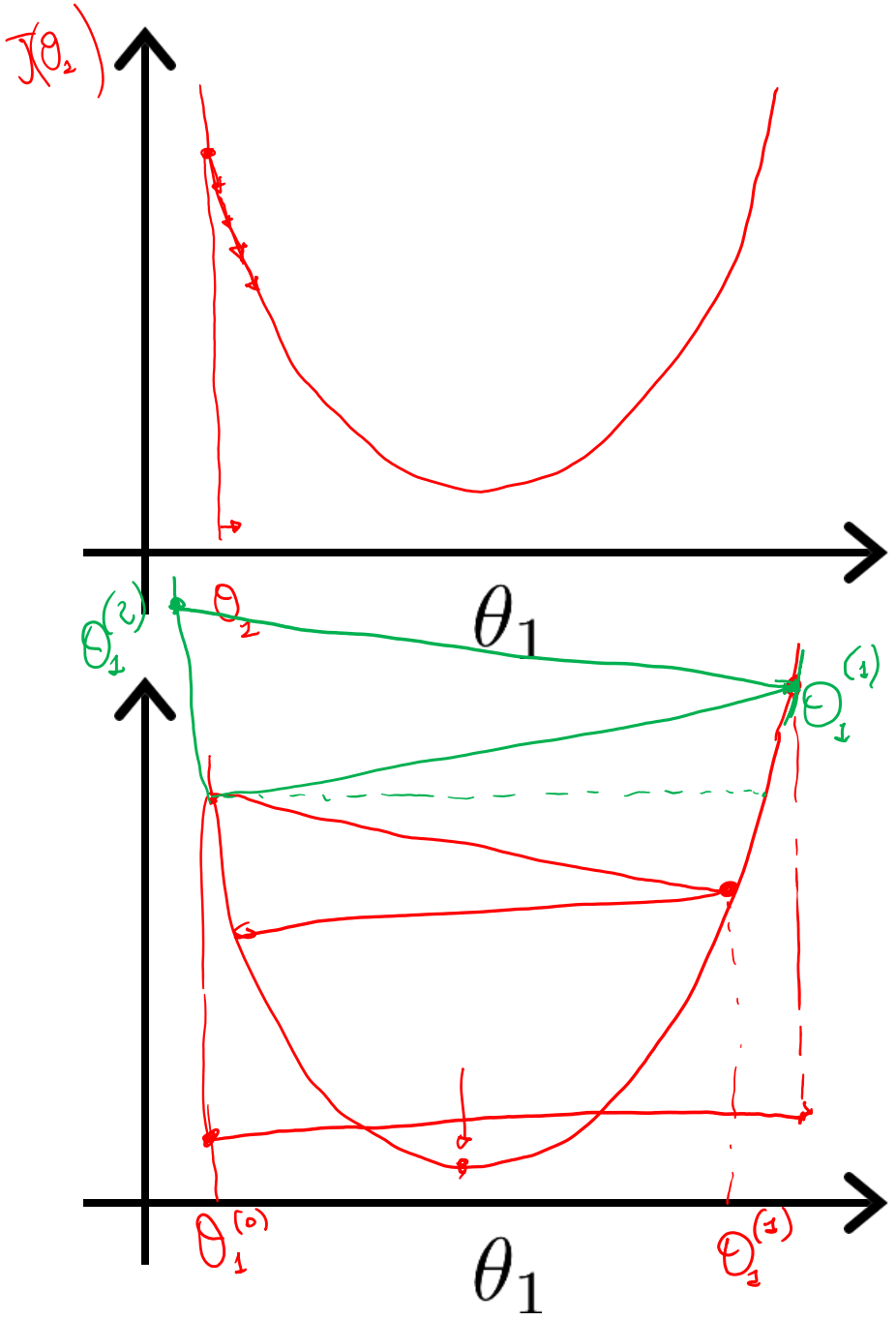
$$\underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\leq 0}$$

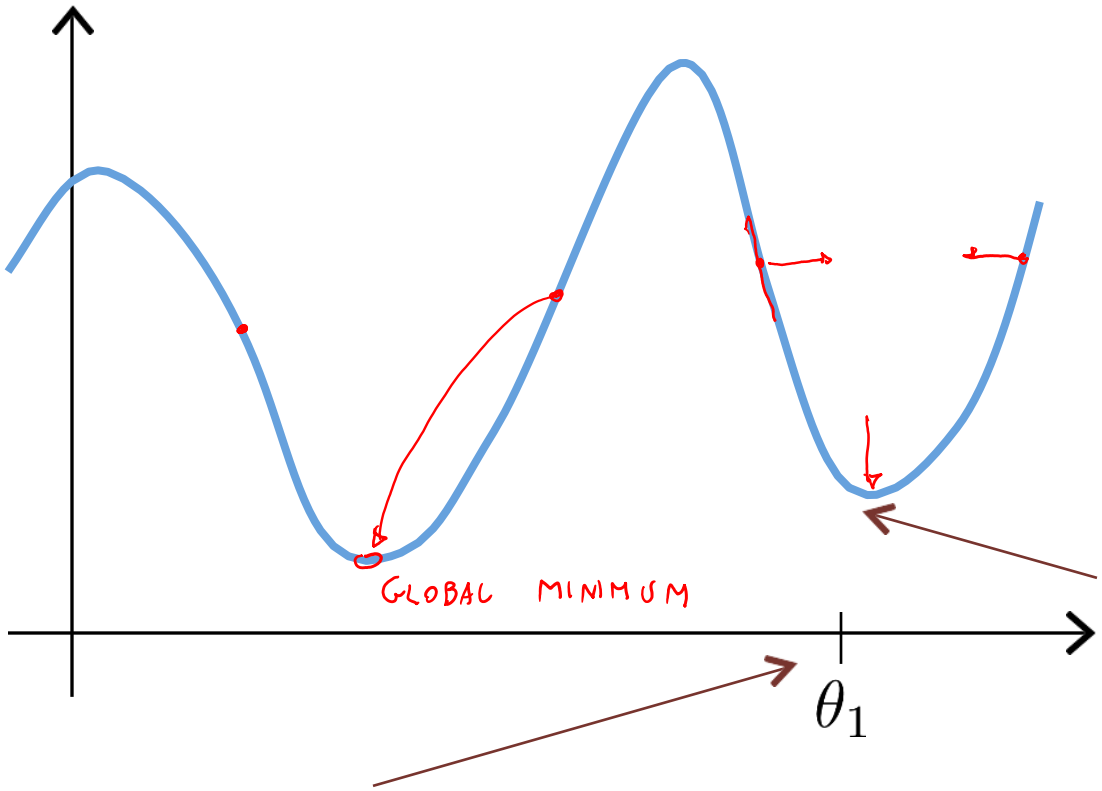
$$\theta_1 := \theta_1 - \alpha \,(\text{NEGATIVE NUMBER})$$

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

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.





θ_1 at local optima

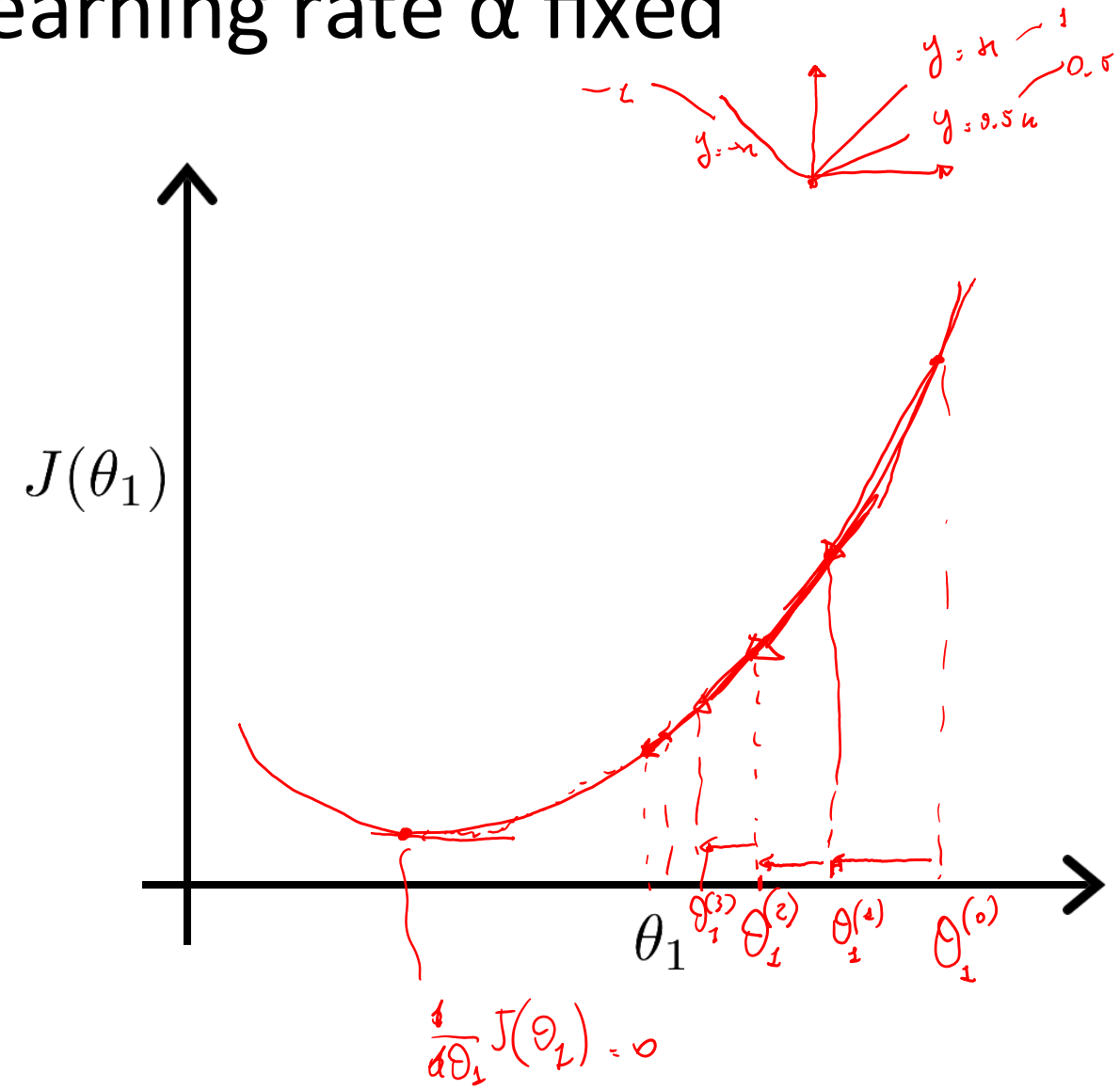
Current value of θ_1

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

Gradient descent can converge to a local minimum, even with the learning rate α fixed

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

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

Gradient descent algorithm

repeat until convergence {

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

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

}

Linear Regression Model

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

PARAMETRIZED BY

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

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

$$= \frac{\partial}{\partial \Theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\Theta_0 + \Theta_1 x^{(i)} - y^{(i)} \right)^2$$

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

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

Gradient descent algorithm

$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

repeat until convergence {

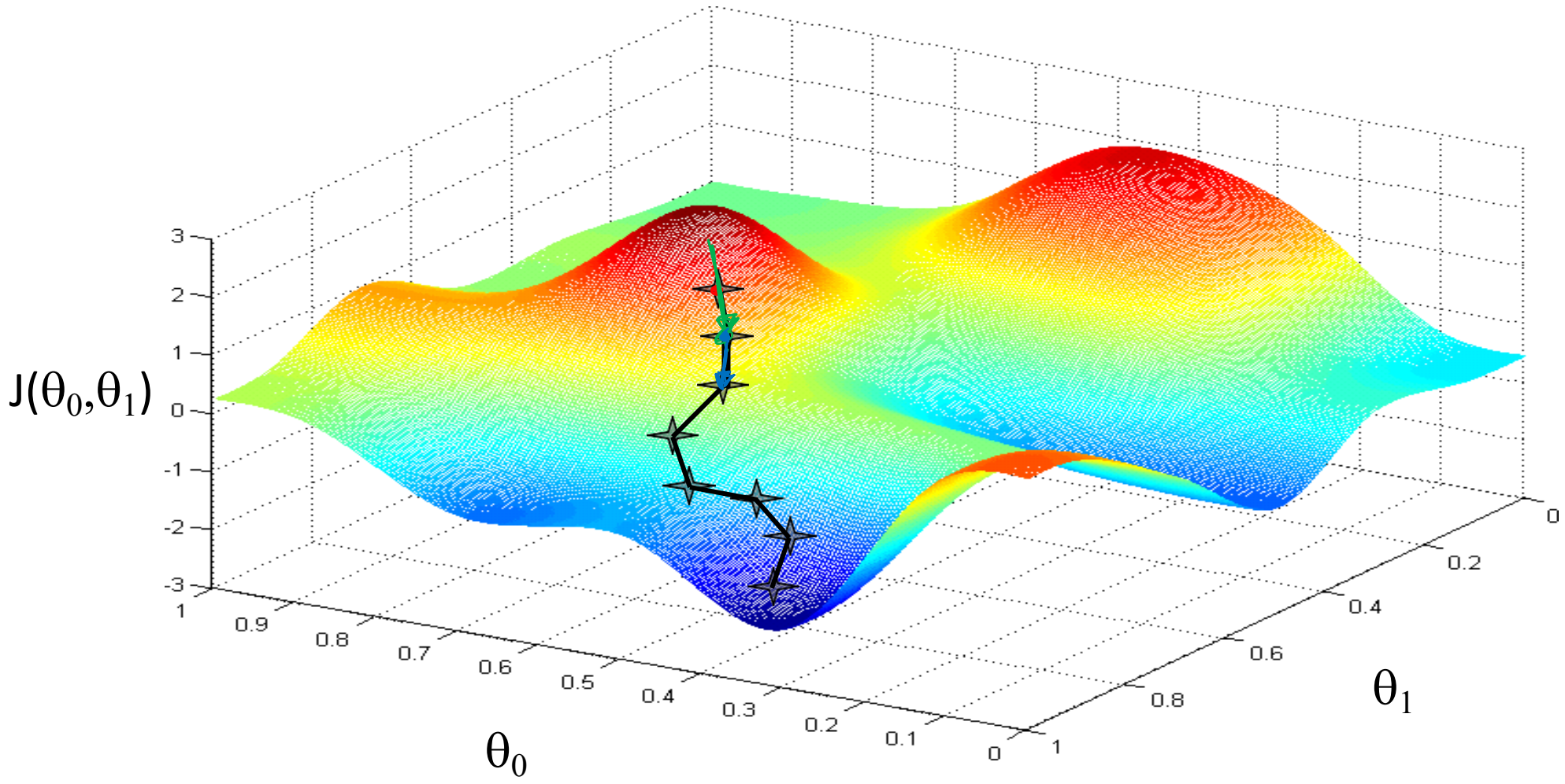
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

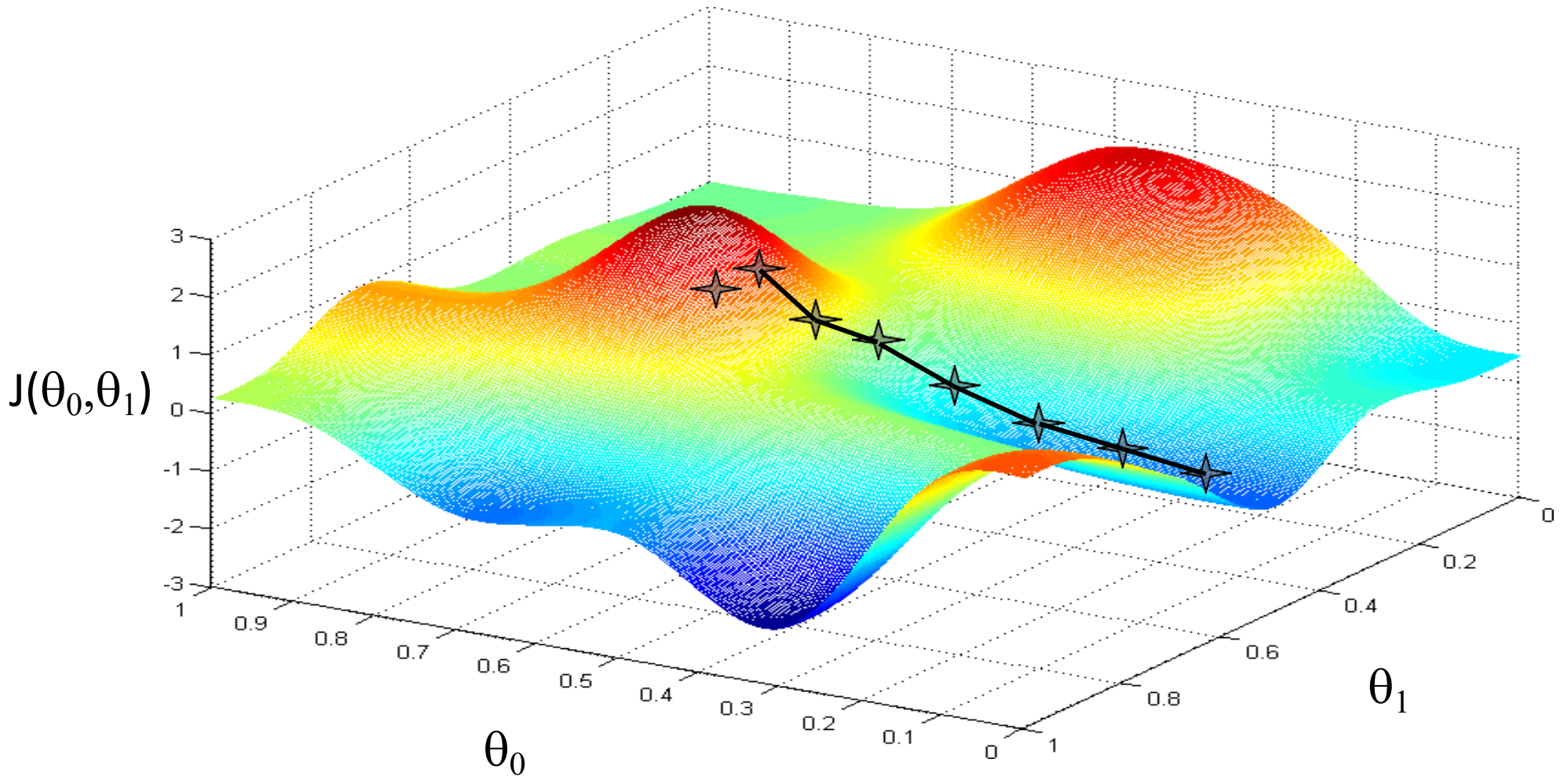
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

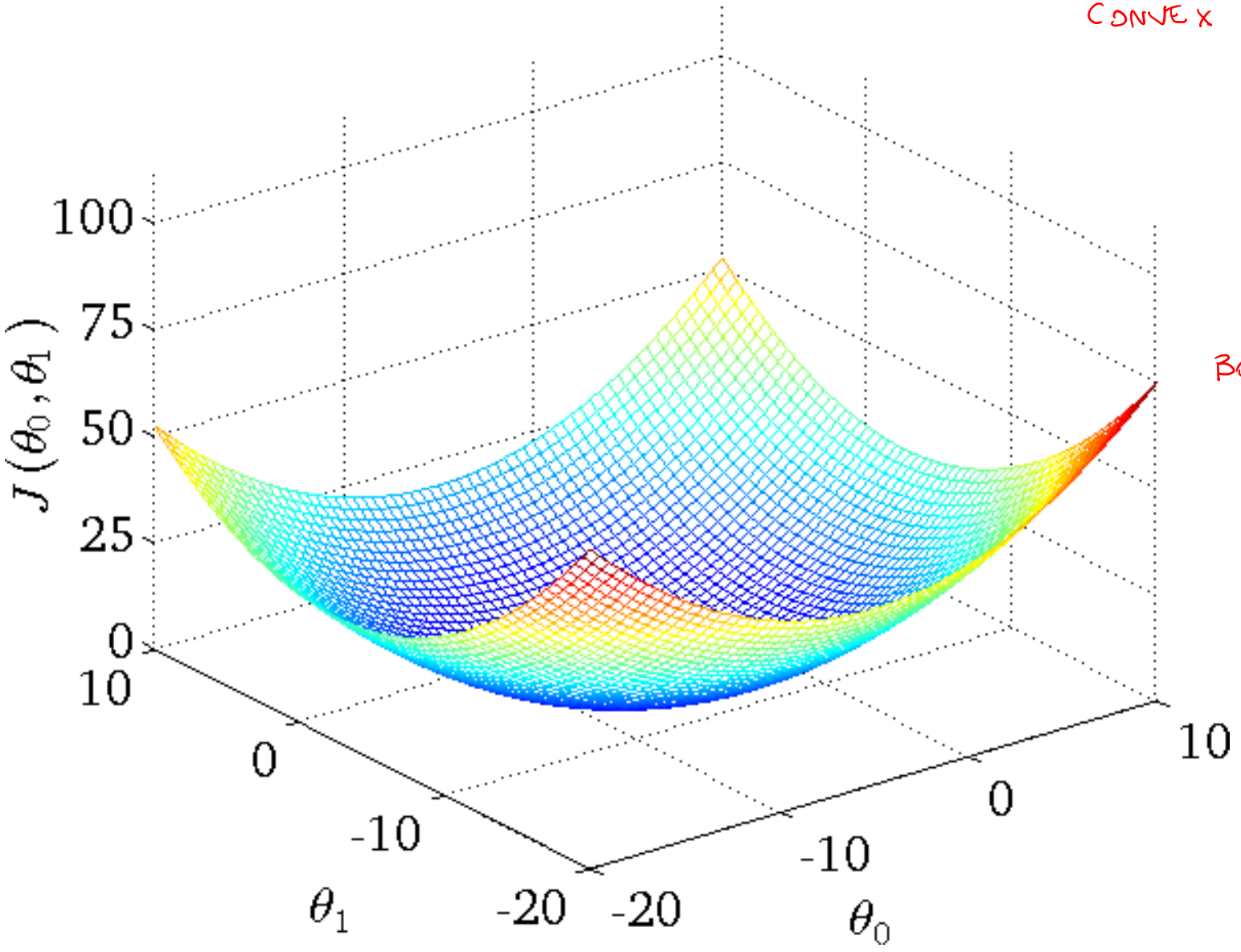
}

update θ_0 and θ_1 simultaneously

$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

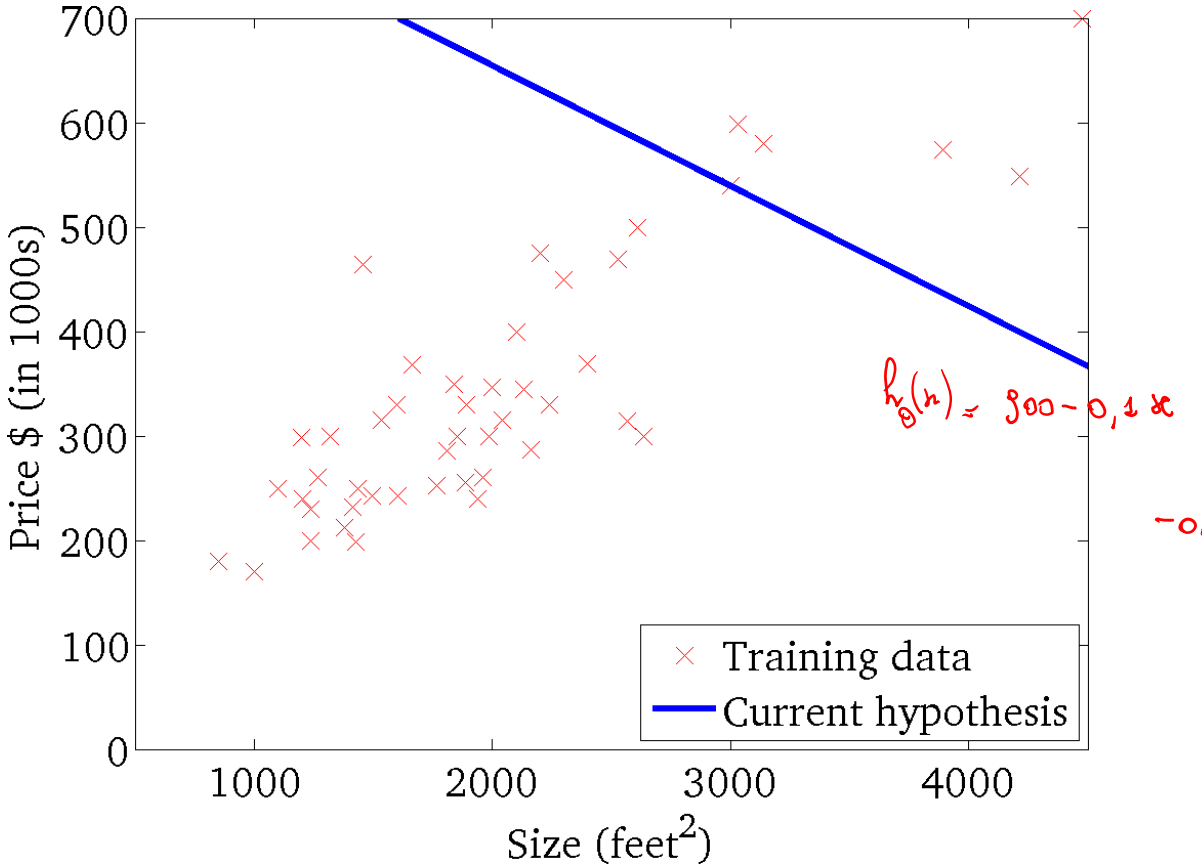

$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0,\theta_1)$
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


CONVEX FUNCTION
BOWL-SHAPED
100
75
50
25
0
$J(\theta_0, \theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0

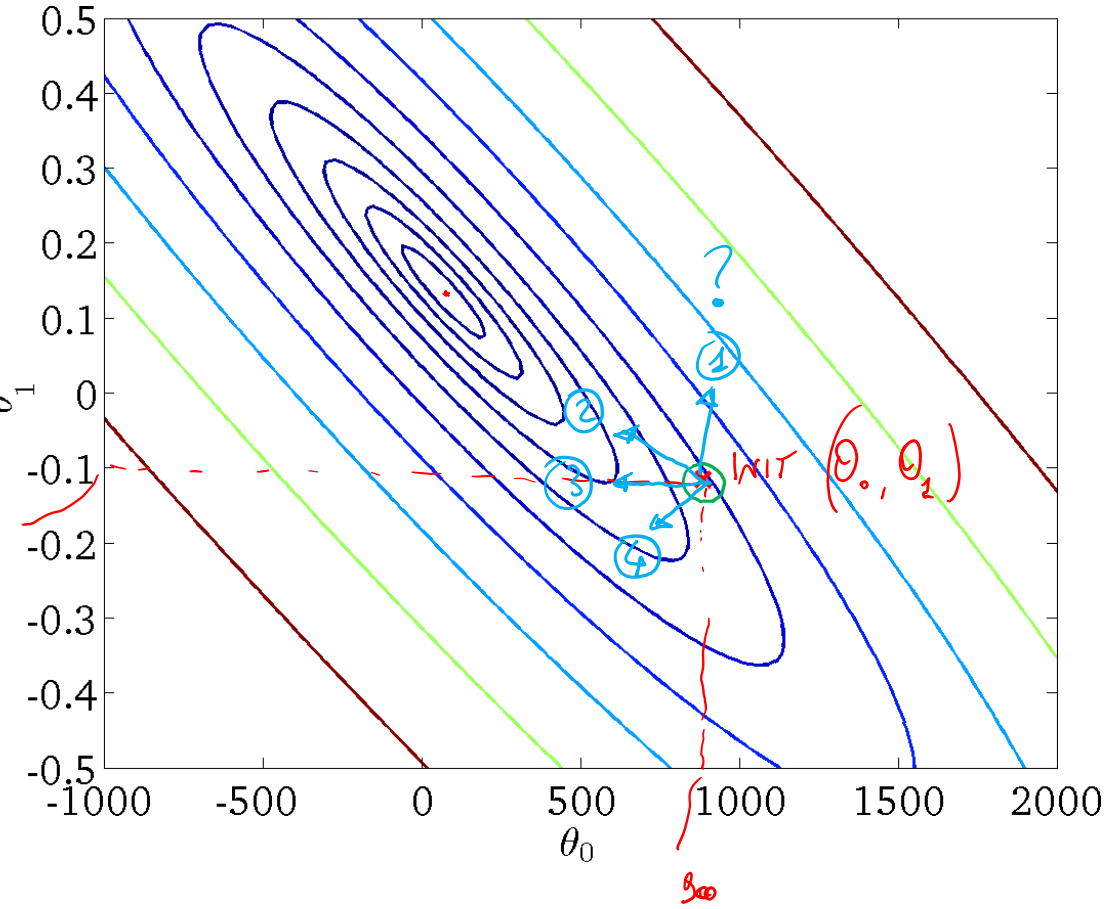
$h_\theta(x)$

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



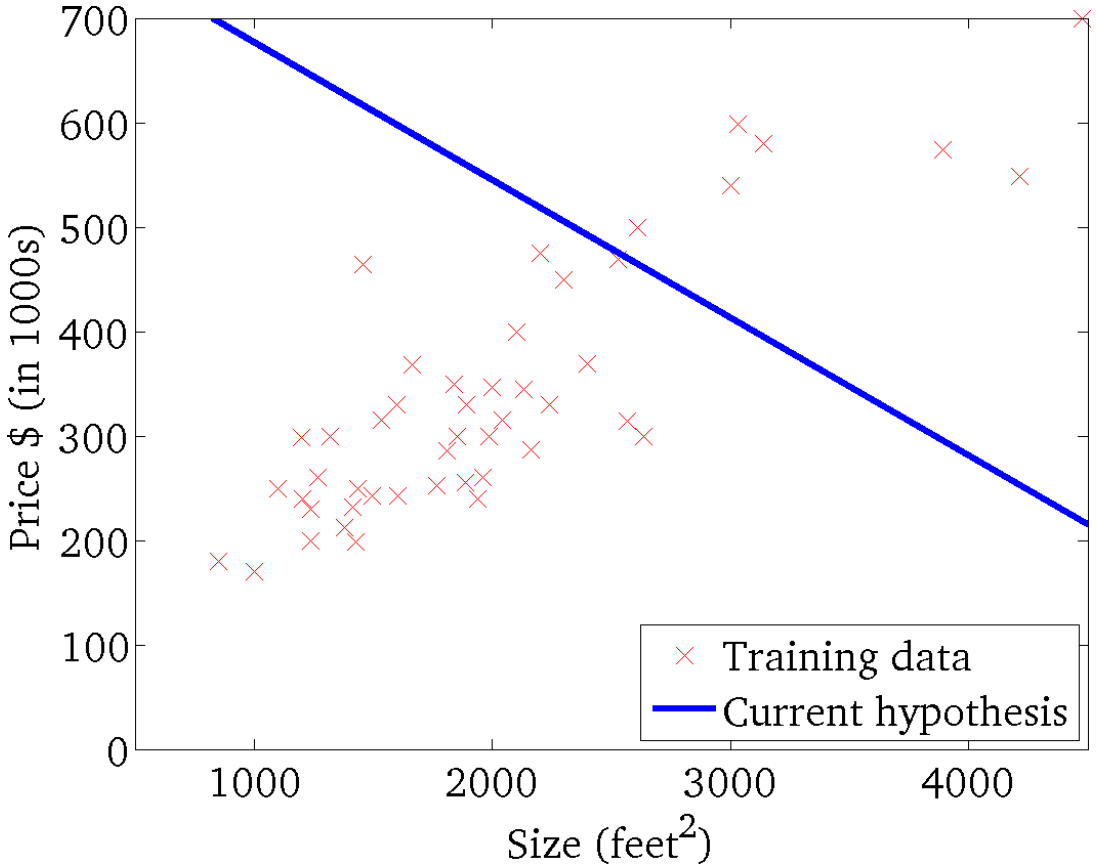
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



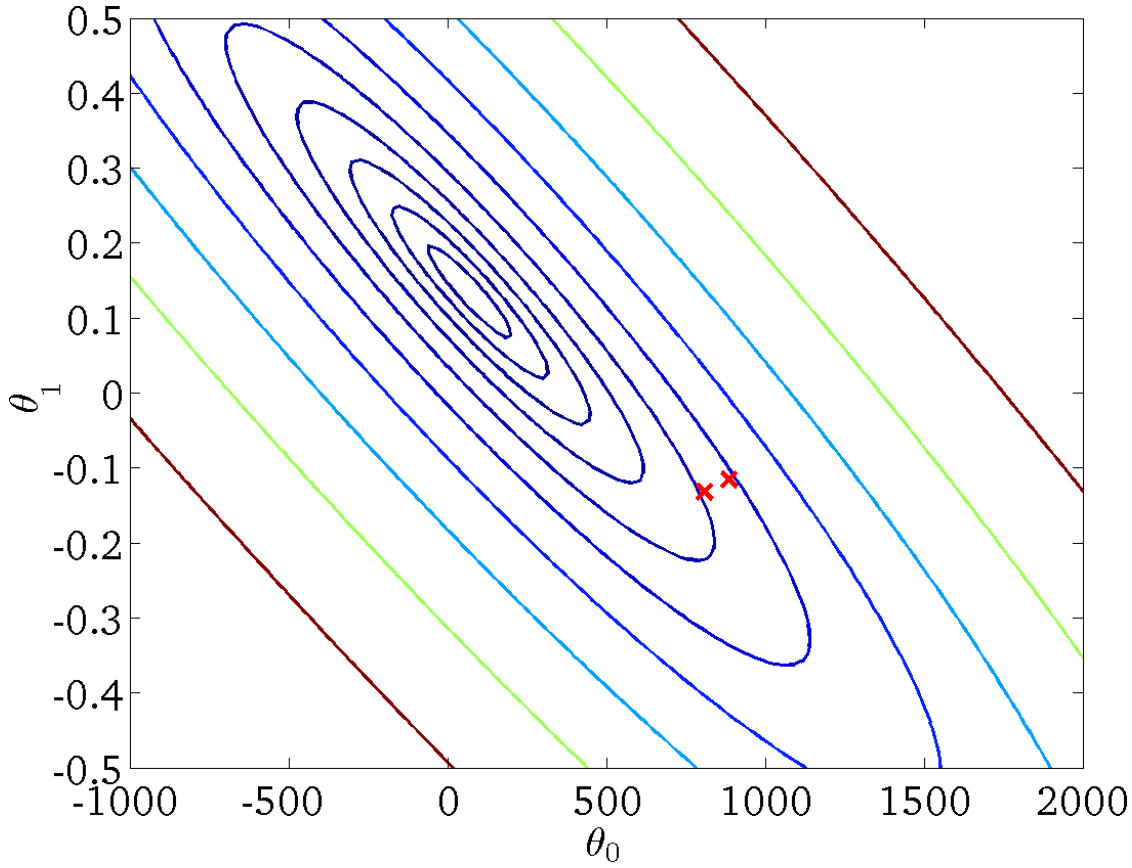
$h_\theta(x)$

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



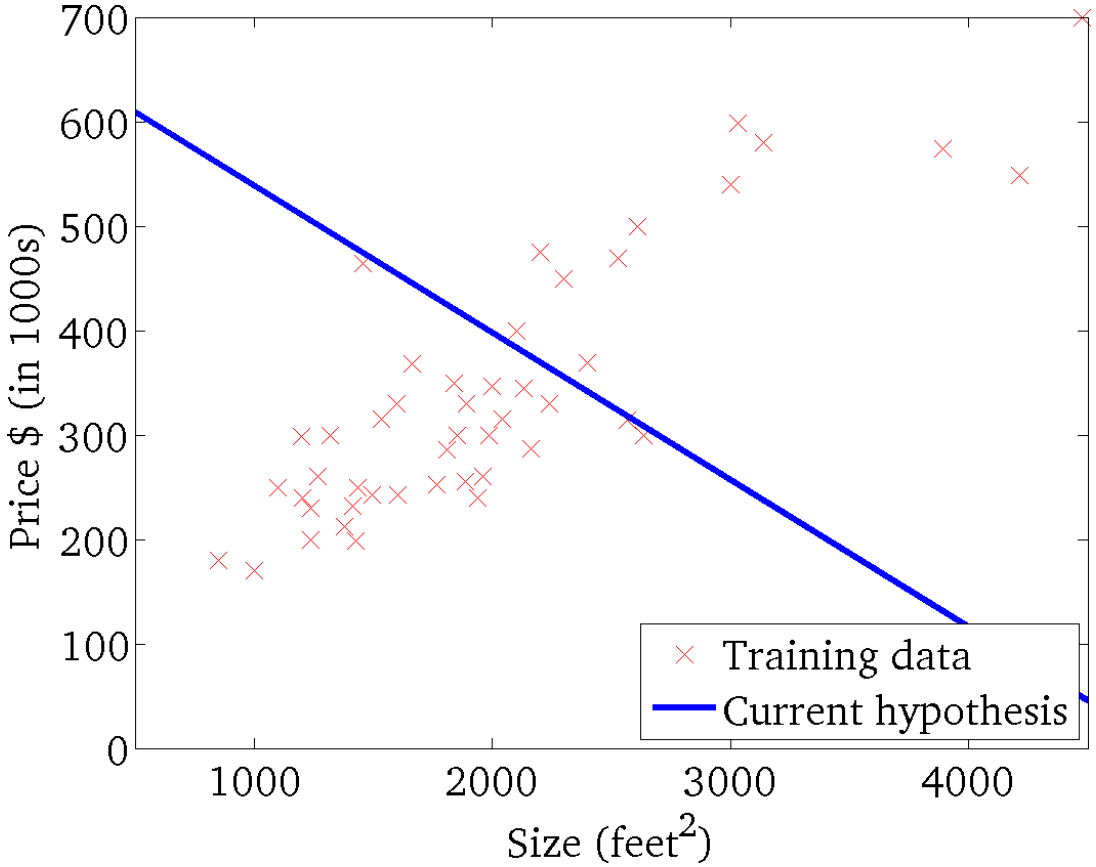
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



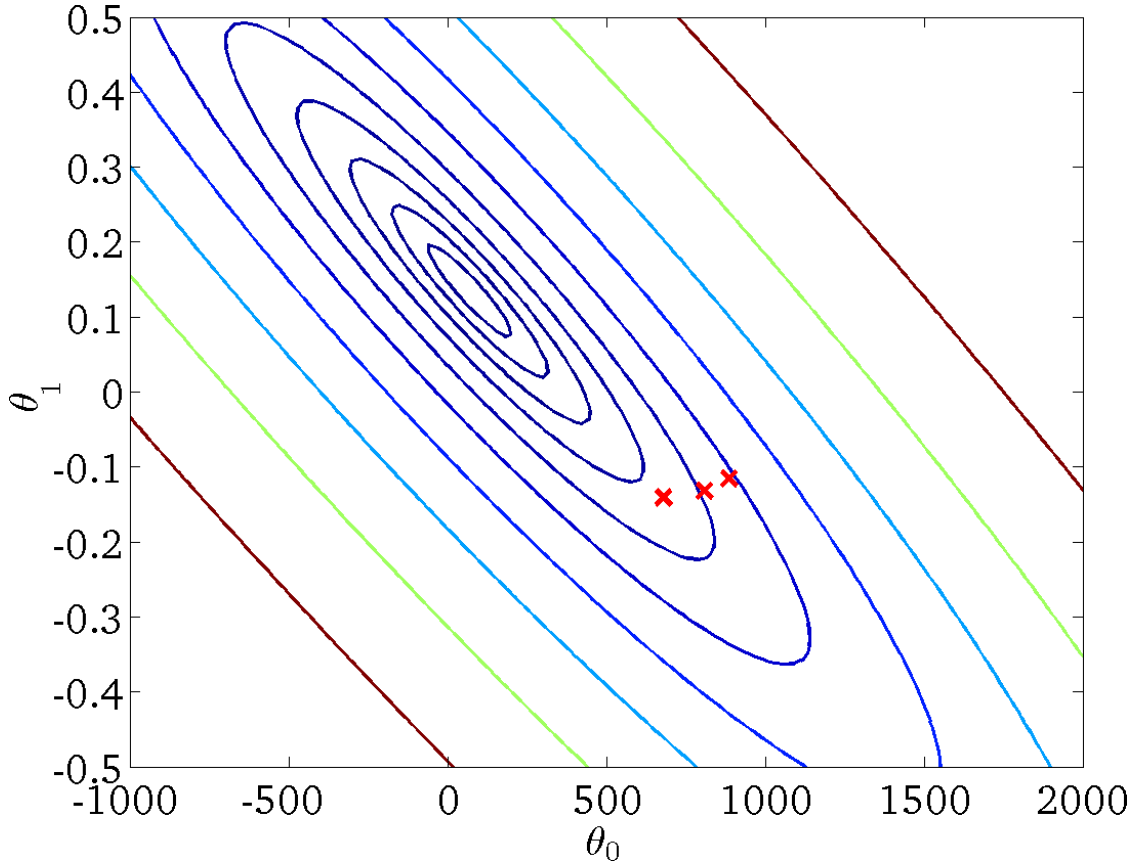
$h_\theta(x)$

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



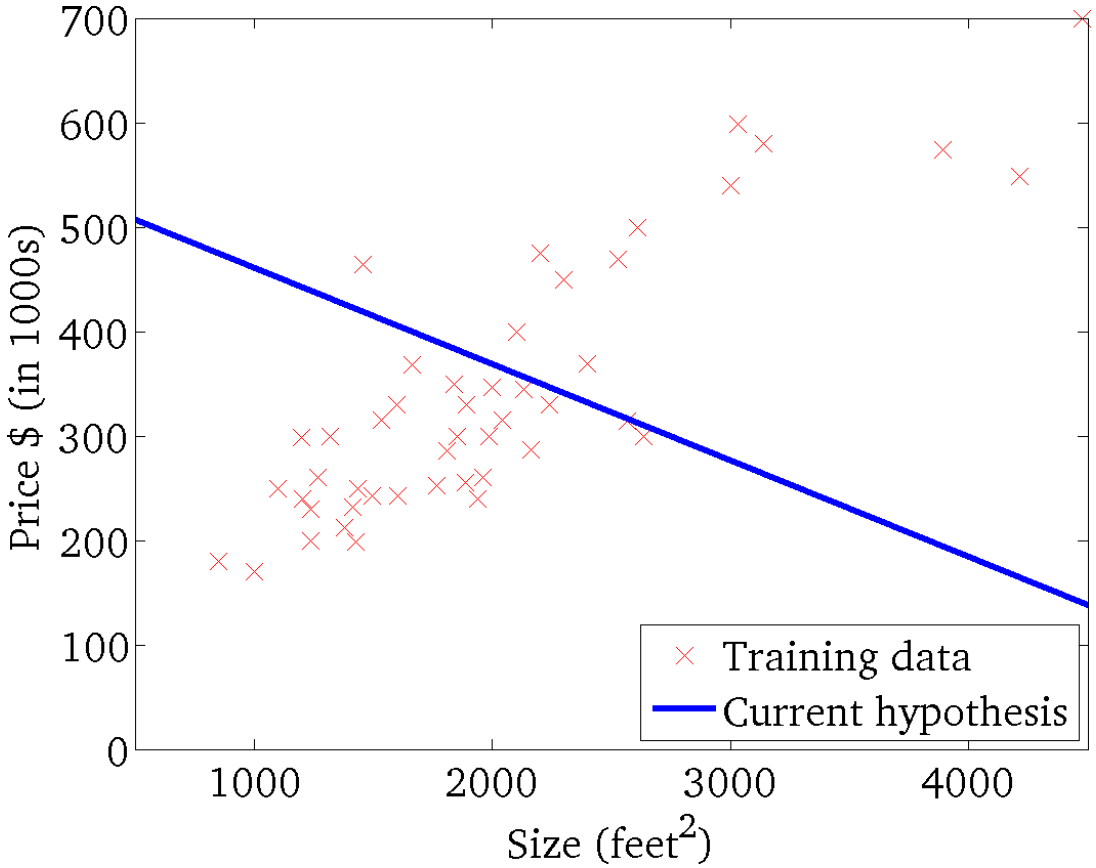
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



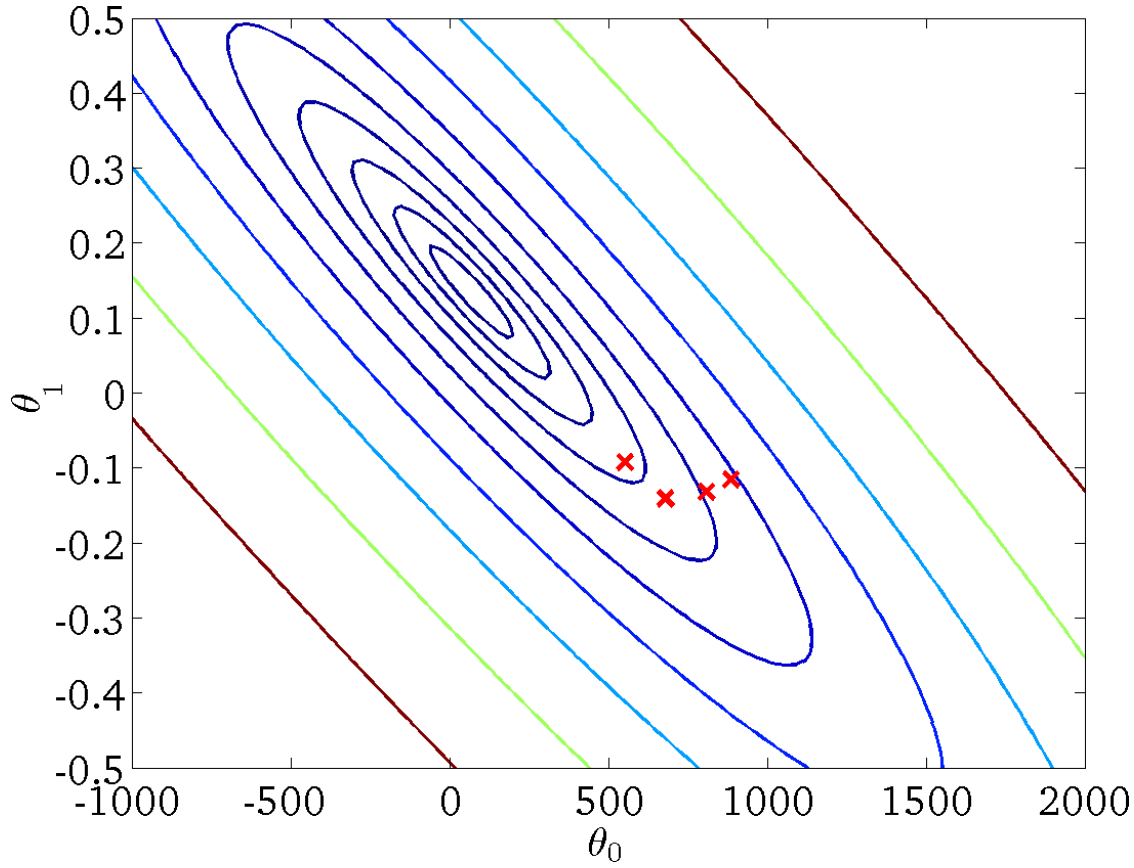
$h_\theta(x)$

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



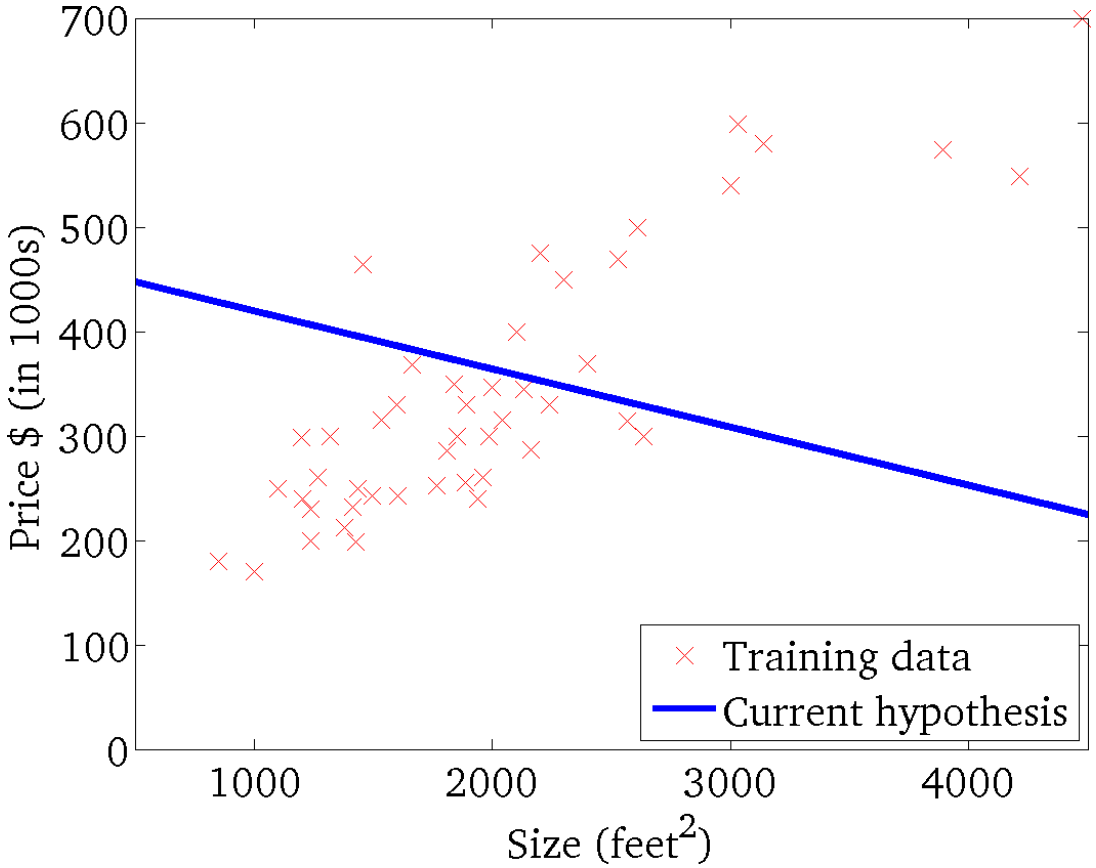
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



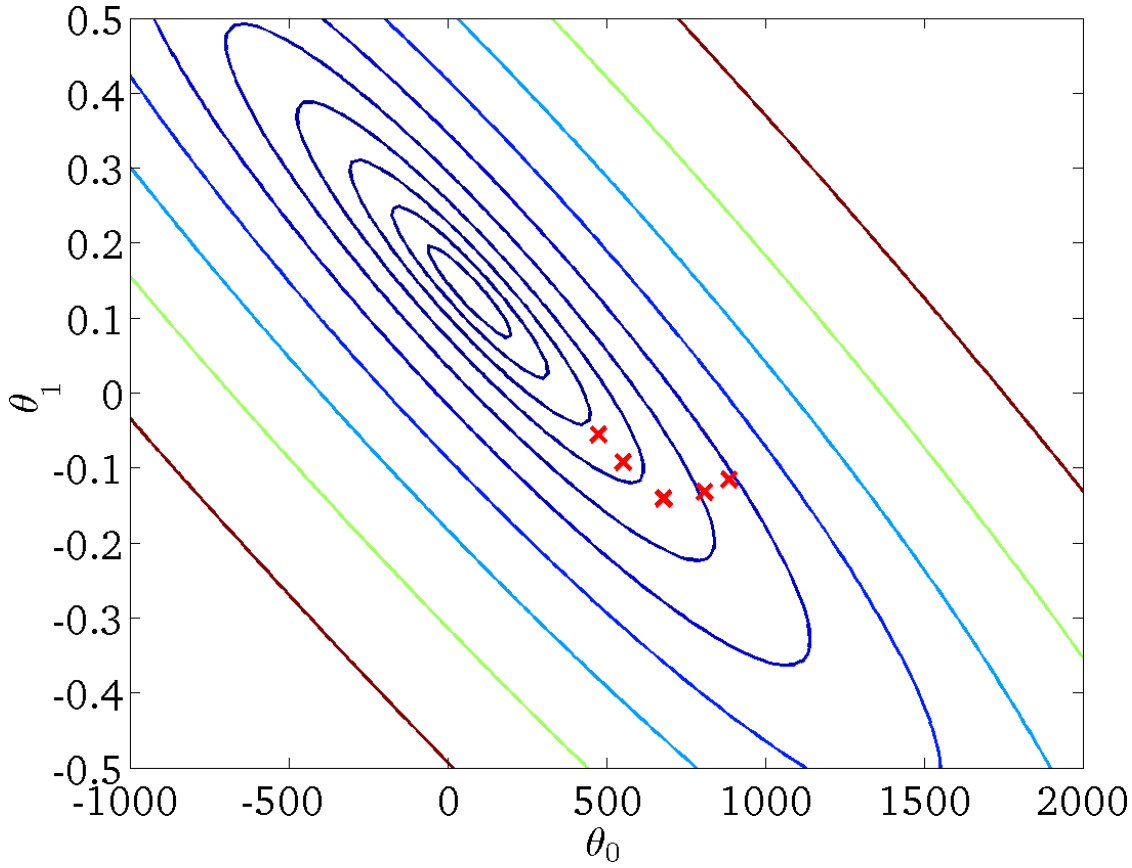
$h_\theta(x)$

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



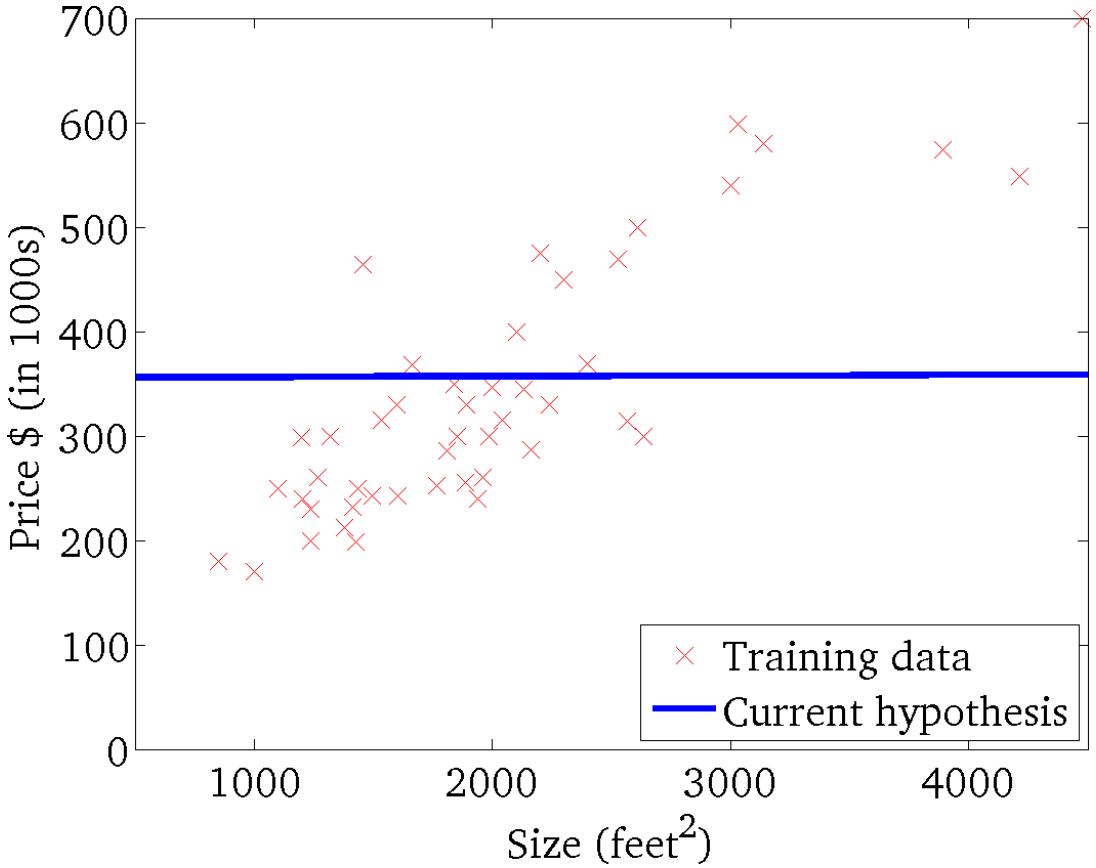
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



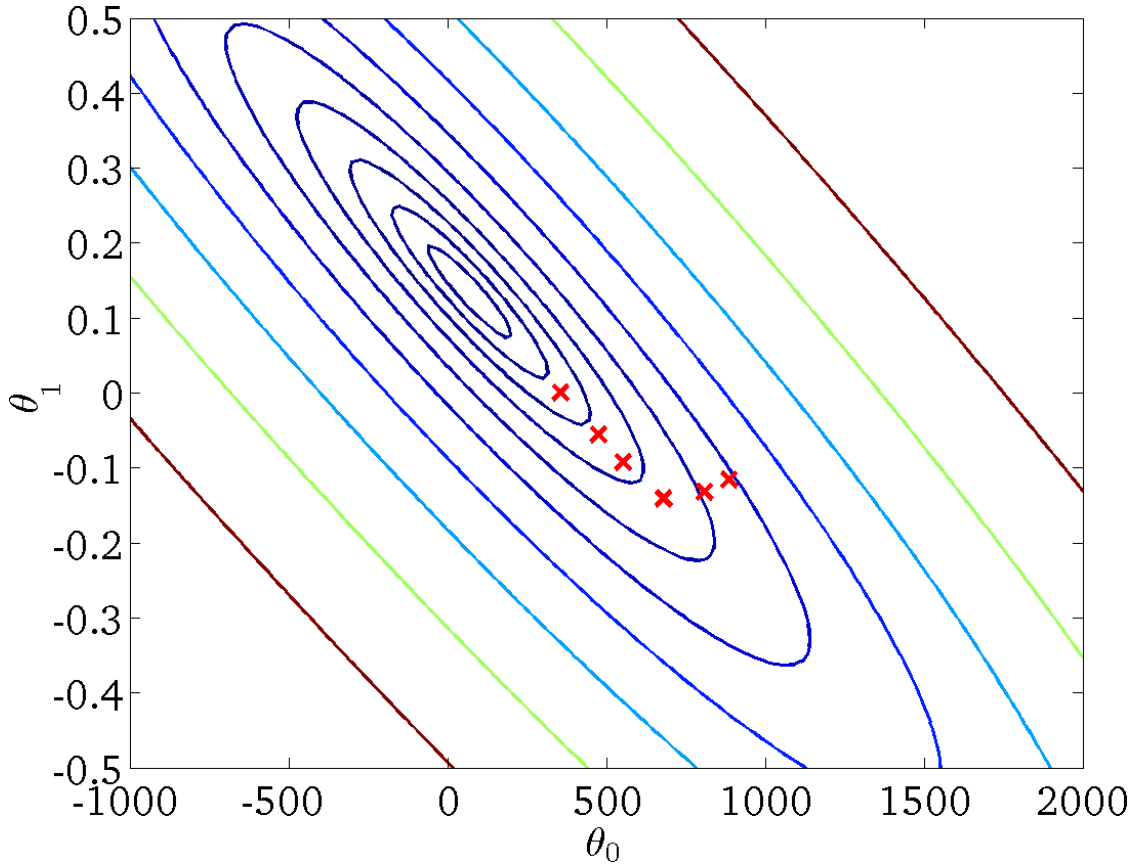
$h_\theta(x)$

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



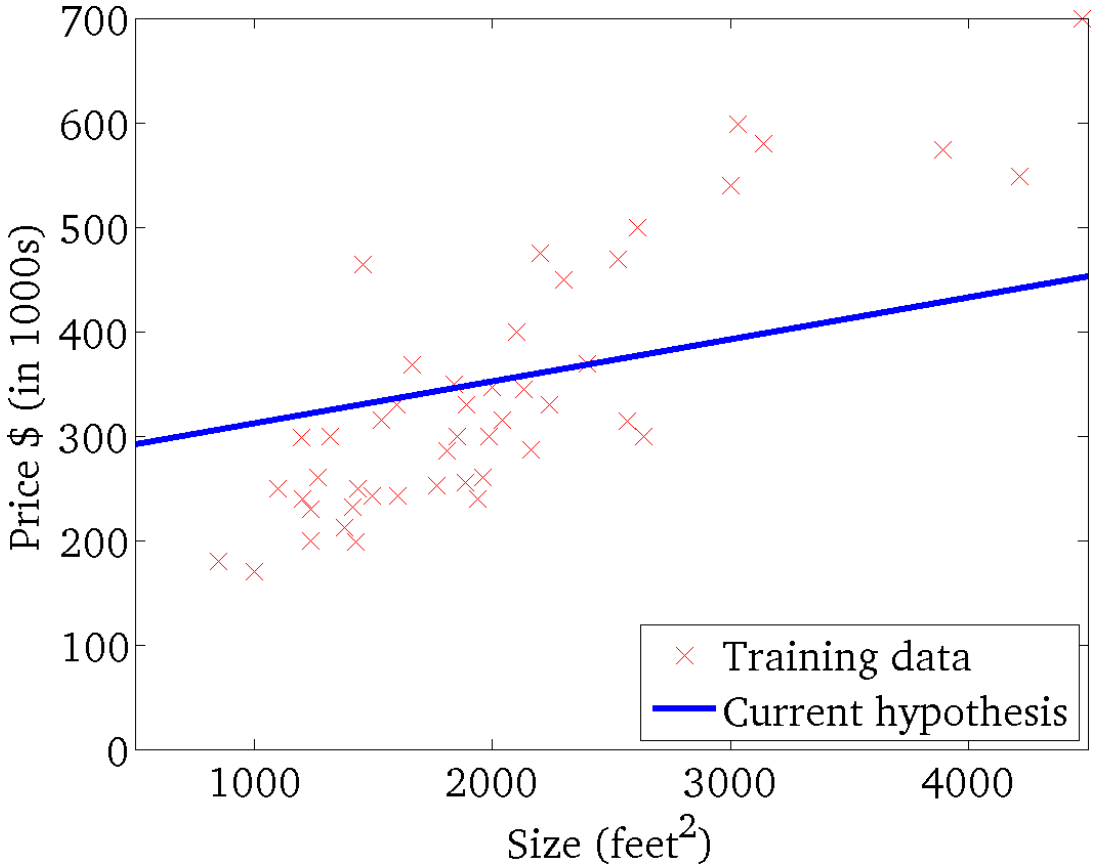
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



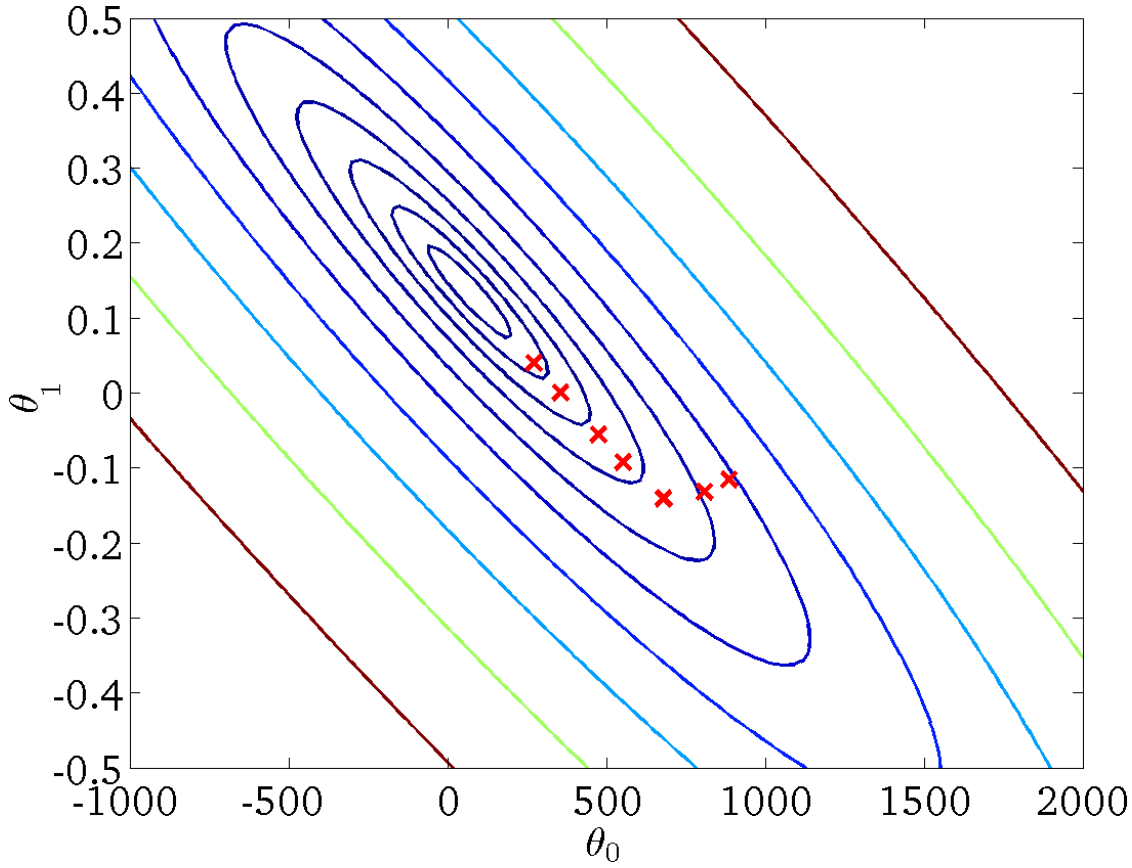
$h_\theta(x)$

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



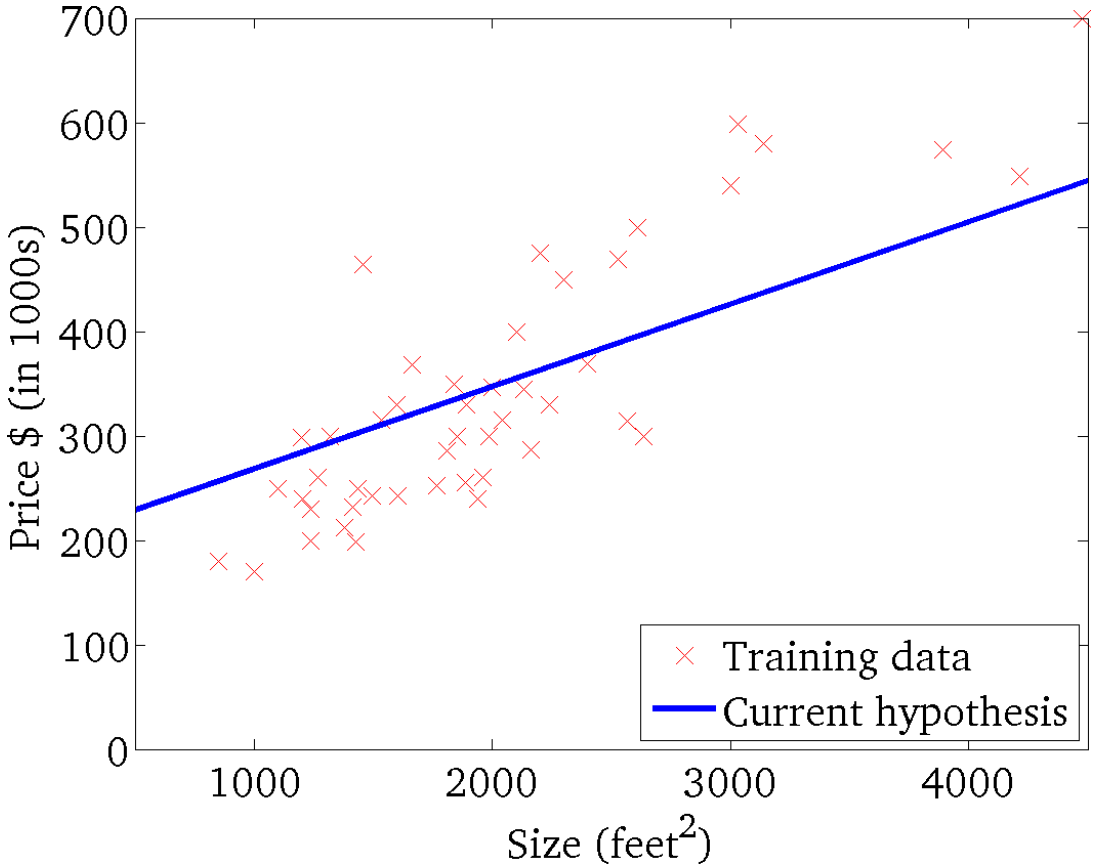
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



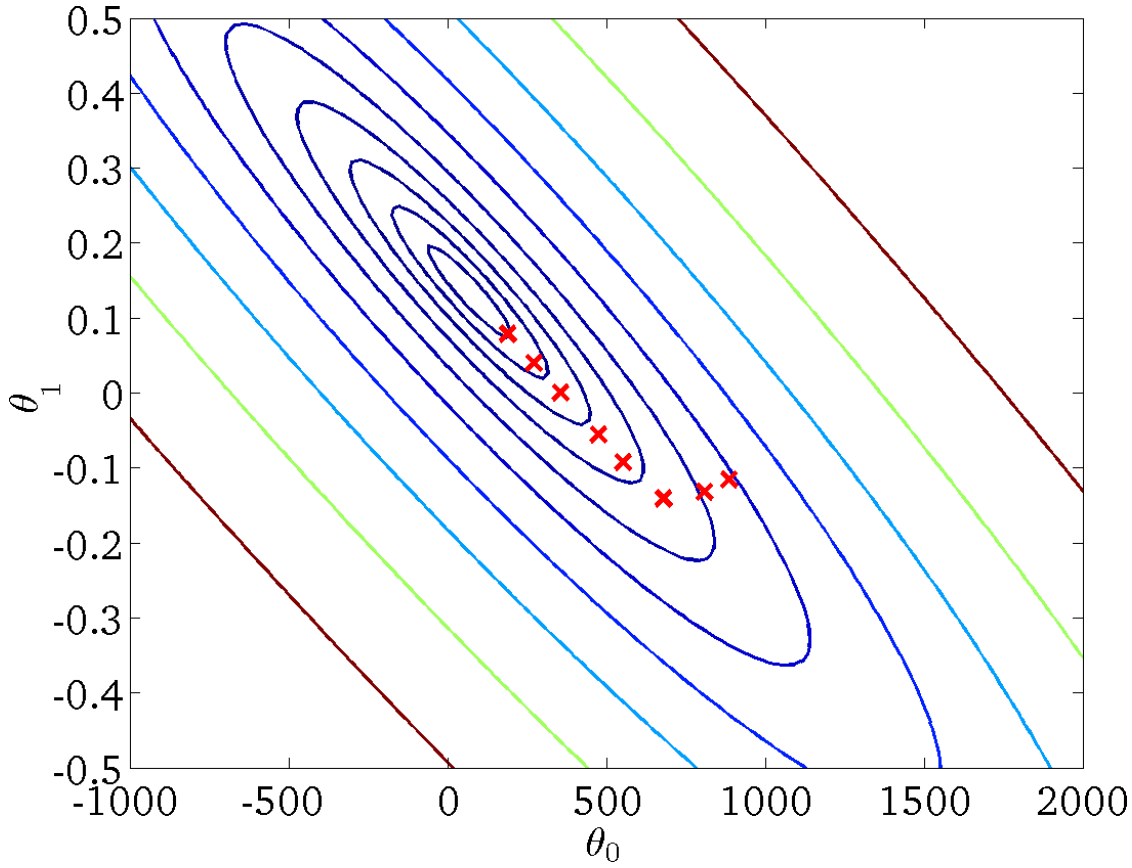
$h_\theta(x)$

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



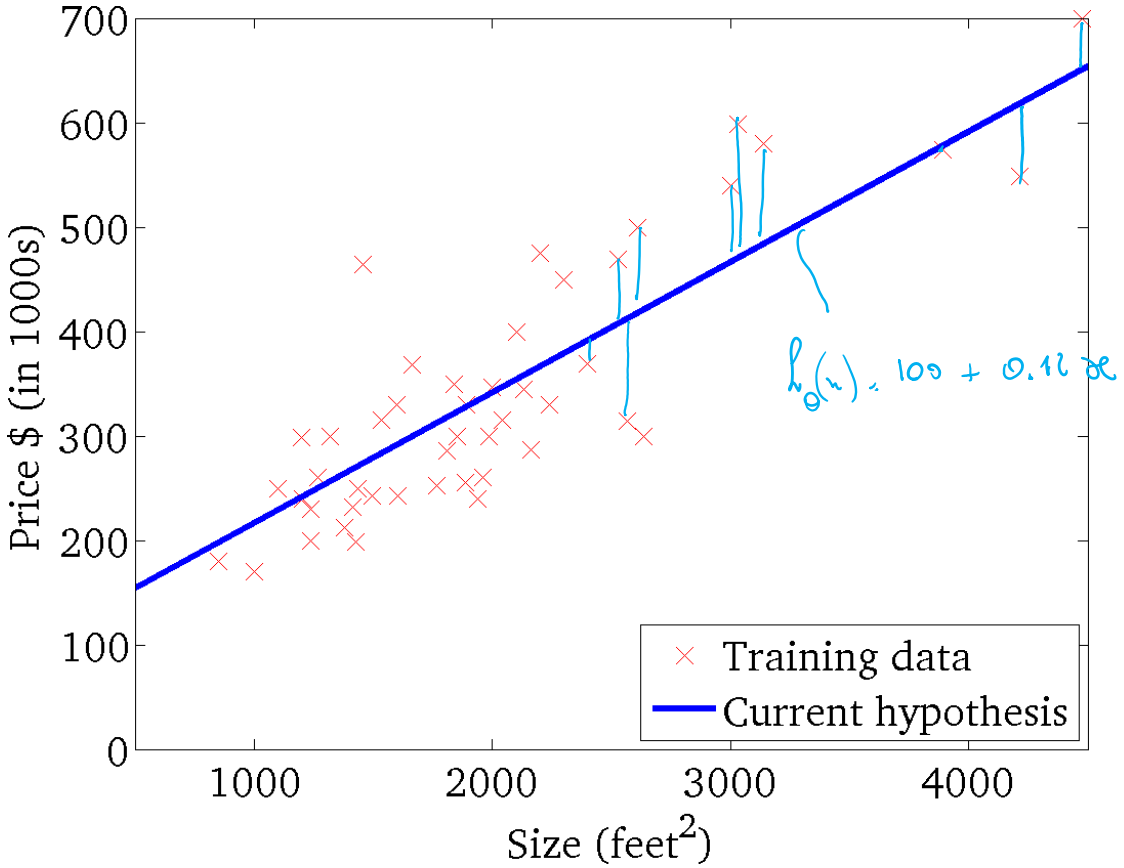
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



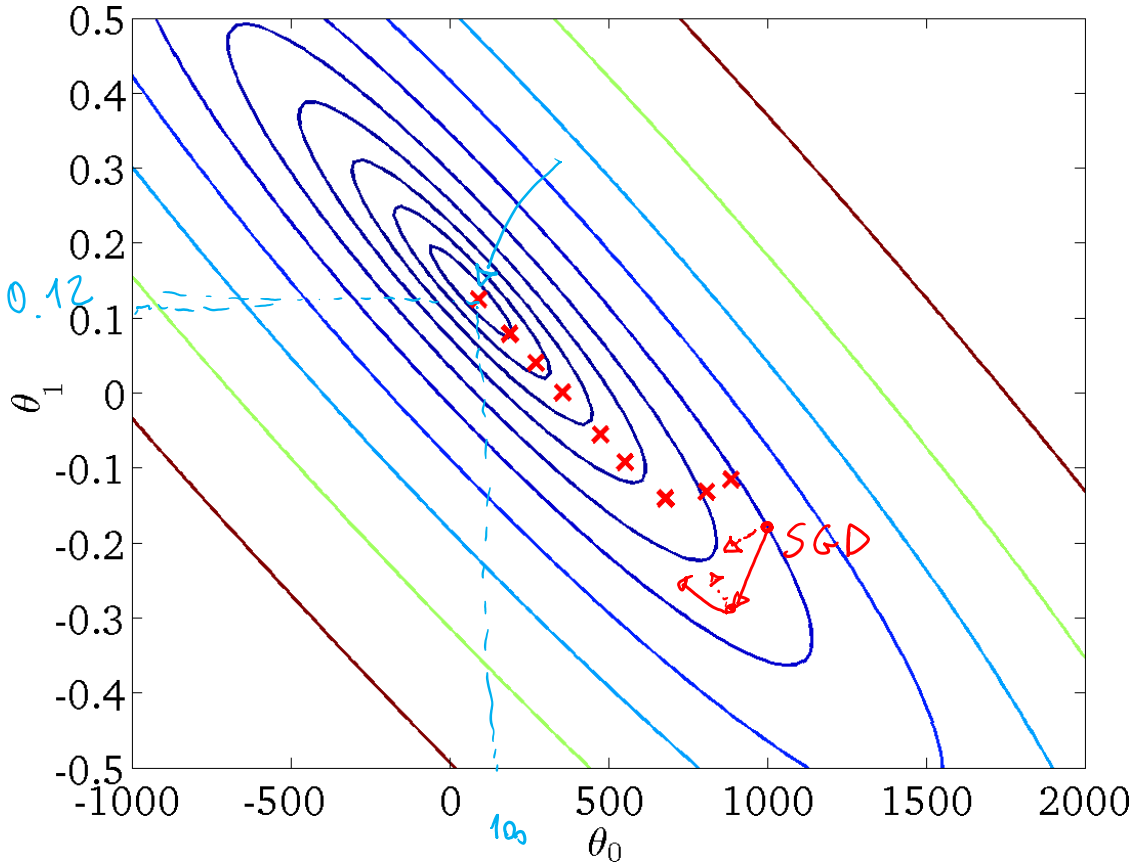
$h_\theta(x)$

(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)



“Batch” Gradient Descent

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

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

Gradient Descent

- The gradient is the result of considering all dataset samples
- This may be very slow
- Can we do something more efficient?

Stochastic Gradient Descent (SGD)

- For large training sets, the evaluation the gradient over all samples may be expensive
- **Stochastic** or "**online**" gradient descent approximates the true gradient by a gradient at a single example
  - Pseudocode:
    - Choose an initial vector of parameters $\boldsymbol{\theta_0}$ and learning rate α
    - Repeat until convergence:
      - Randomly shuffle examples in the training set
      - For $k = 1,2, \dots m$, do:
        - $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \alpha \nabla J^{(k)}(\boldsymbol{\theta}^{(i)})$

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (\dots)$$

Stochastic Gradient Descent (SGD)

- For large training sets, the evaluation the gradient over all samples may be expensive
- **Stochastic** or "**online**" gradient descent approximates the true gradient by a gradient at a single example
  - Pseudocode:
    - Choose an initial vector of parameters $\boldsymbol{\theta_0}$ and learning rate α
    - Repeat until convergence:
      - Randomly shuffle examples in the training set
      - For $k = 1,2, \ldots m$, do:
        - $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \alpha \nabla J^{(k)}(\boldsymbol{\theta}^{(i)})$
- Normally preferable: **mini-batch** gradient descent
  - Consider a mini-batch at each step
  - This normally results in smoother convergence
  - This normally is faster, thanks to vectorization libraries

- SHUFFLE (PERMUTE ORDER)
- SELECT MINI-BATCHES
- ITERATE UP TO AN EPOCH
- REPEAT

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:

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

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

Previously (n=1):

Repeat $\{$

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

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

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

$x_0^{(i)} = 1$

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

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

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

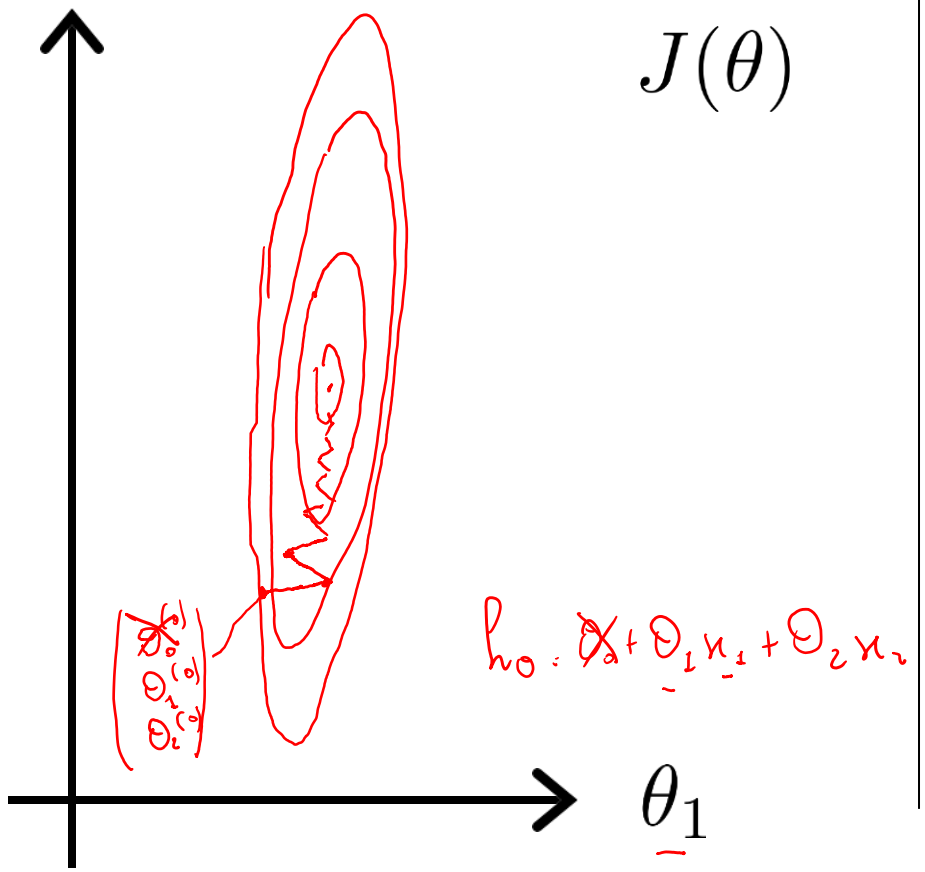
$\ldots$

Feature Scaling

Idea: Make sure features are on a similar scale.

E.g. x_1 = size (0–2000 feet^2)

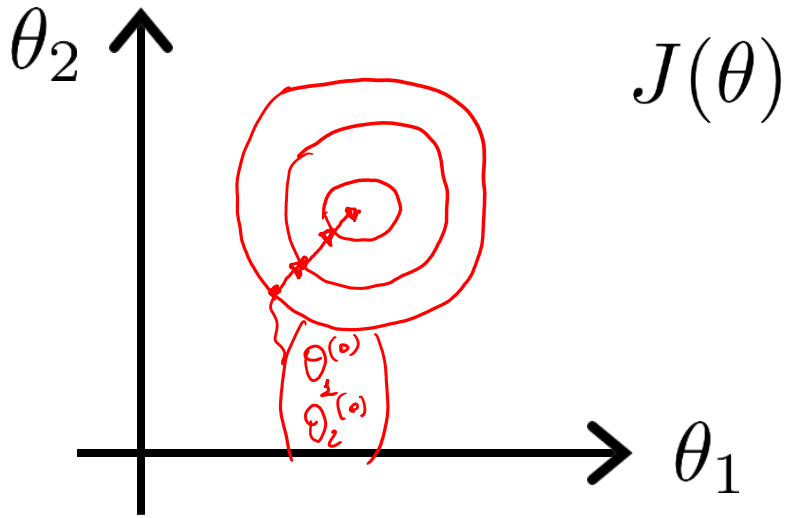
x_2 = number of bedrooms (1–5)



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

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

$-1 \leq x_j \leq 1$



Feature Scaling

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

$x_0 = 1$

$0 \leq x_1 \leq 3$ ✓

$-2 \leq x_2 \leq 0.5$ ✓

$-100 \leq x_3 \leq 500$ ✗

$-0.0001 \leq x_4 \leq 0.0006$ ✗

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. $x_1 = \frac{size - 1000}{2000}$ — $\mu_1 = 1000$ AVERAGE SIZE IN FEET^2 — RANGE

$x_2 = \frac{\#bedrooms - 2}{5 \;\; 4}$ — $\mu_2 = 2$ — 1 - 5

$-0.5 \leq x_1 \leq 0.5, \; -0.5 \leq x_2 \leq 0.5$

$x_j \leftarrow \frac{x_j - \mu_j}{s_j}$ — μ_j: AVERAGE VALUE OF x_j IN TRAINING SET

s_j: RANGE (MAX - MIN) (OR STANDARD DEVIATION)

$j = 1 \ldots n$

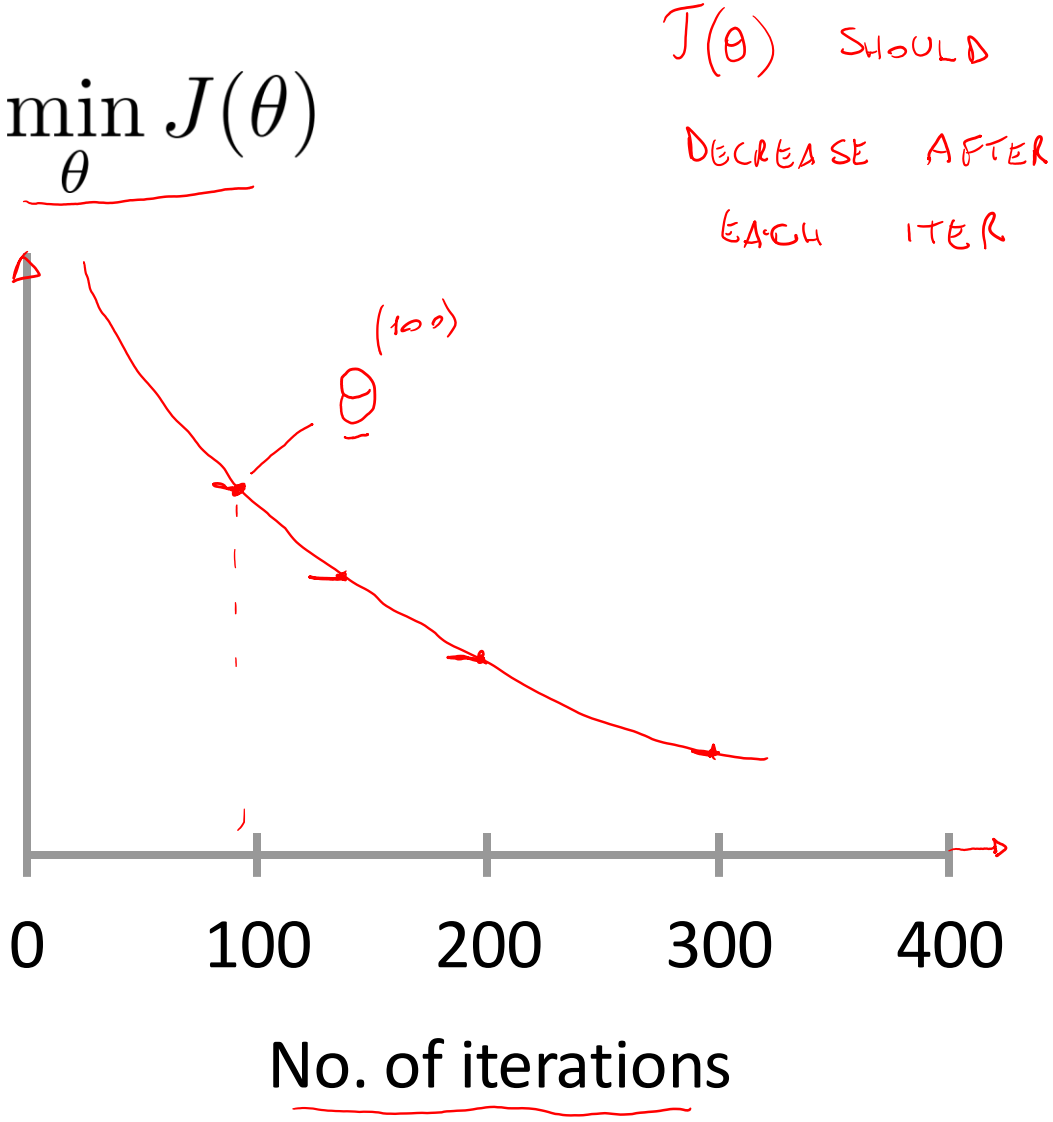
Gradient Descent: Learning Rate

Gradient descent

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

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

Making sure gradient descent is working correctly.

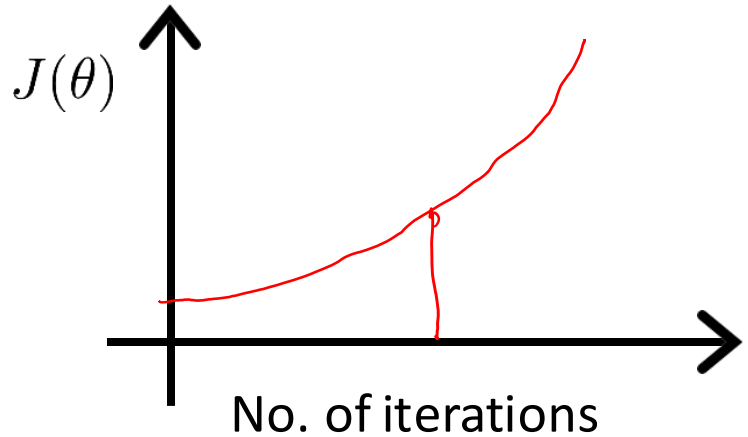


Example automatic convergence test:

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

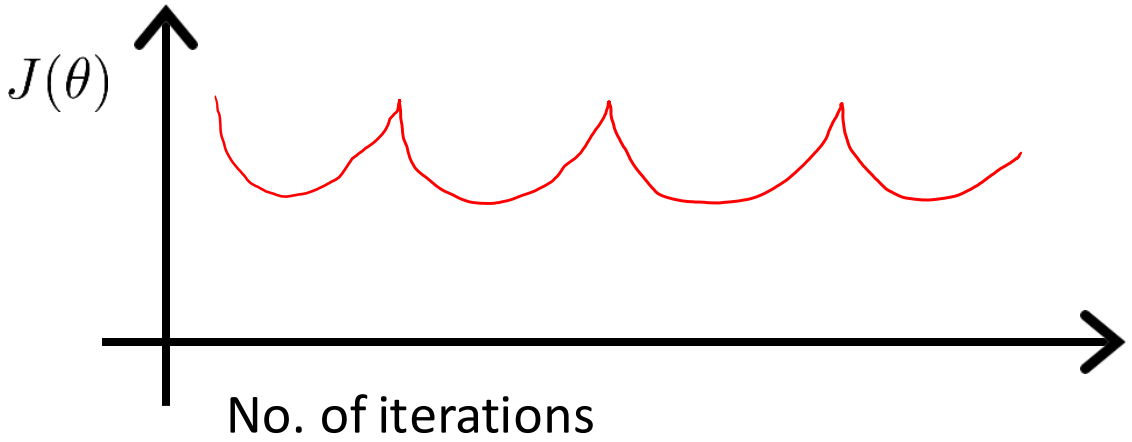
ϵ

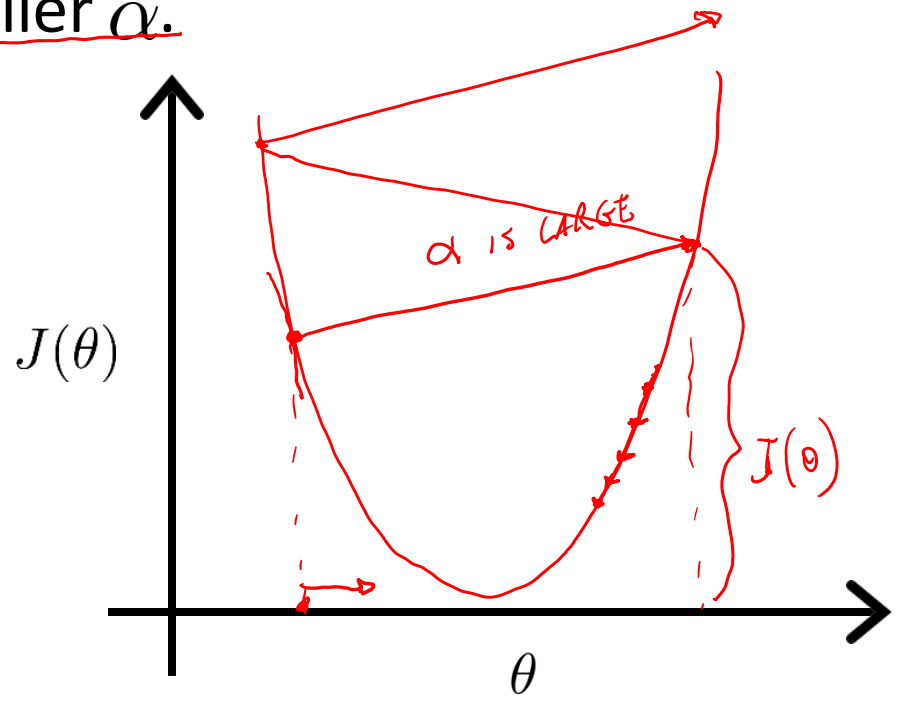
Making sure gradient descent is working correctly.



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.

Summary:

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

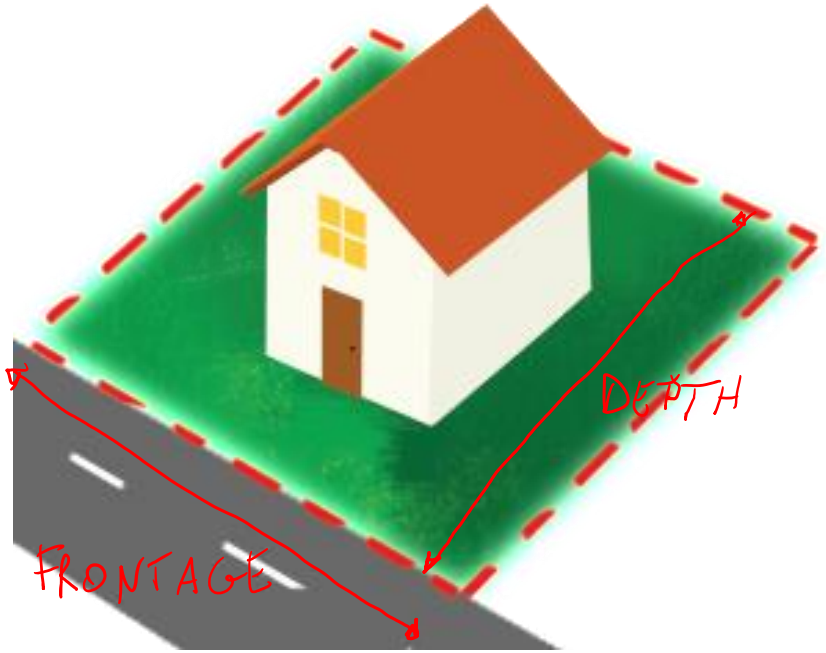
To choose α, try

$$\ldots, 0.001, \quad 0.003 \quad, 0.01, \quad 0.03 \quad, 0.1, \qquad\qquad , 1, \ldots$$

Linear Regression:
Features and Polynomial regression

Housing prices prediction

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



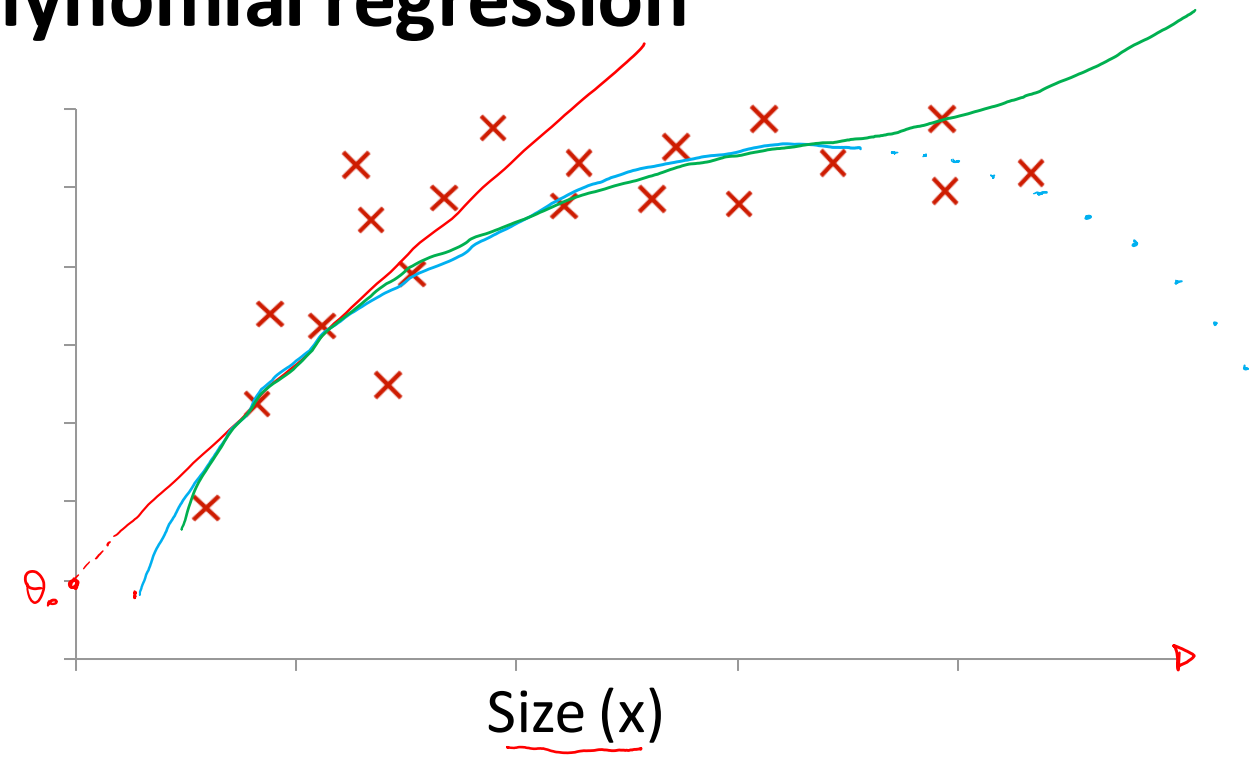
AREA

x : FRONTAGE x DEPTH

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

x : LAND AREA

Polynomial regression



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

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

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

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

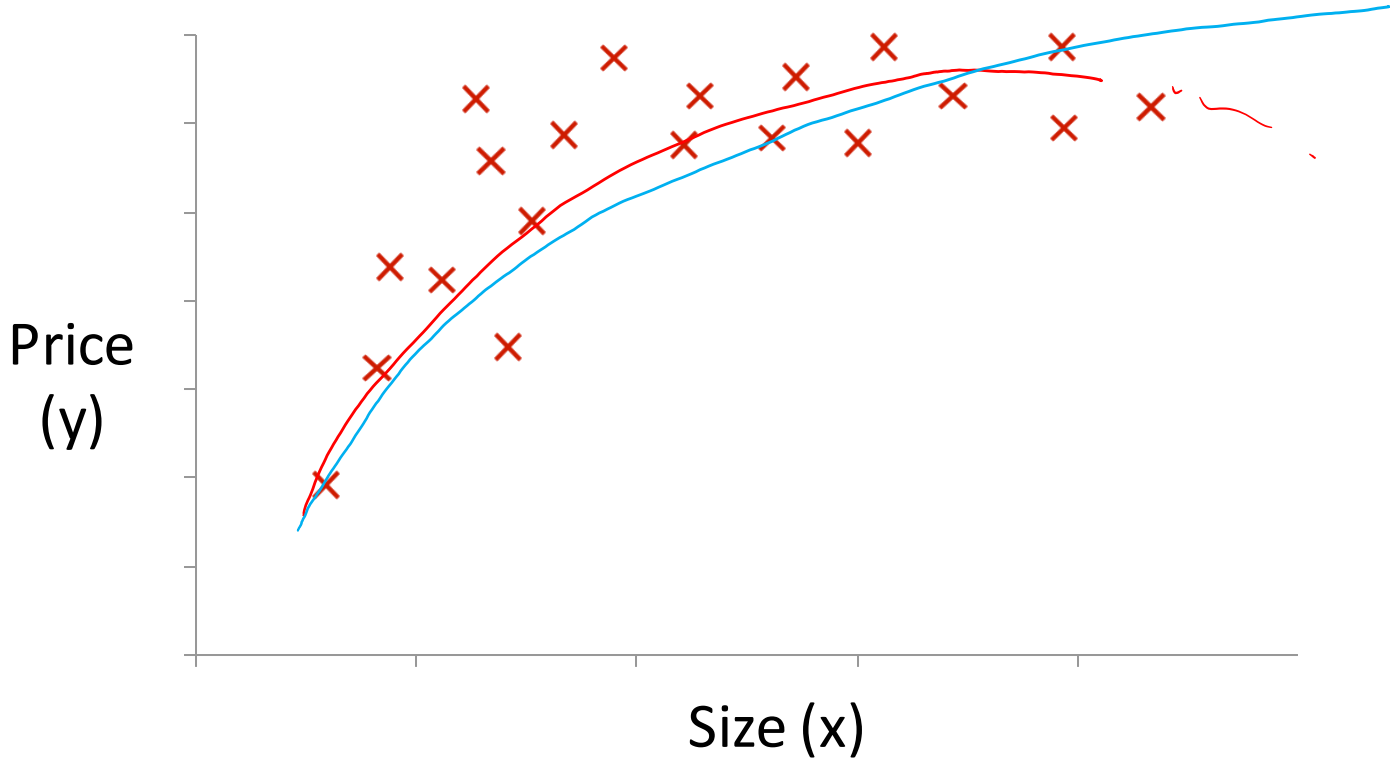
SIZE: 1 ~ 1000

$SIZE^2$: 1 ~ 1,000,000

$SIZE^3$: 1 ~ 10^9

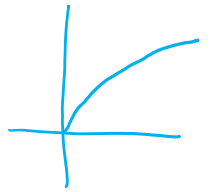
NEED TO SCALE THE FEATURES

Choice of features



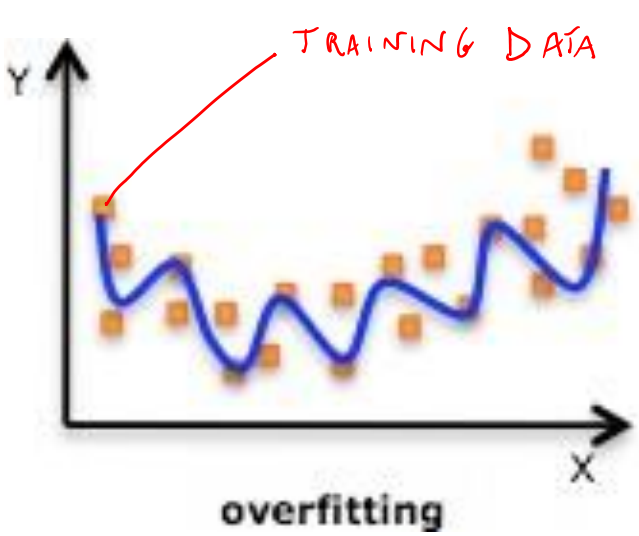
$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

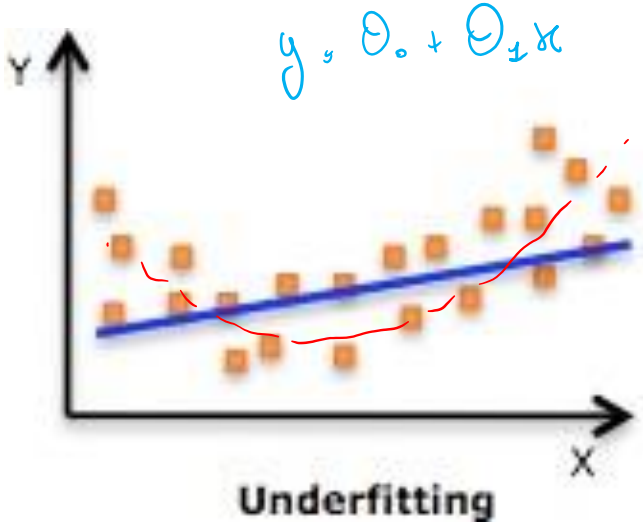
$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$$

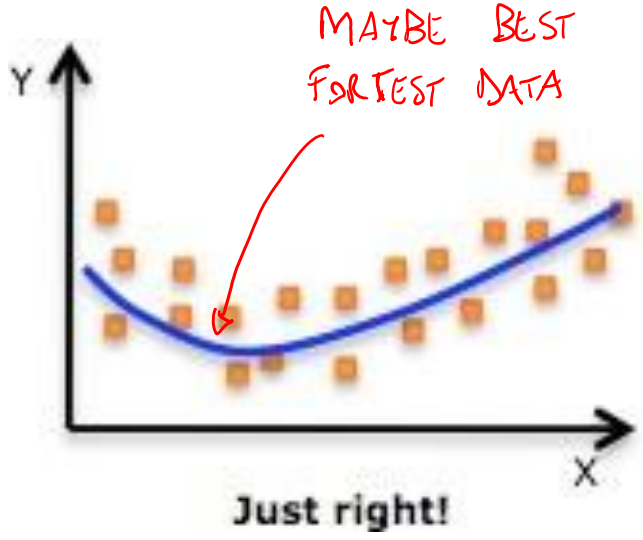


Dangers of (Polynomial) Regression

Overfitting and Underfitting







Outline

- Linear regression
  - One or multiple variables
  - Cost function
- Gradient descent, incl. batch/stochastic GD
- Normal equation
- MSE and Correlation
- Locally-Weighted Regression
- Probabilistic Interpretation of Least Squares

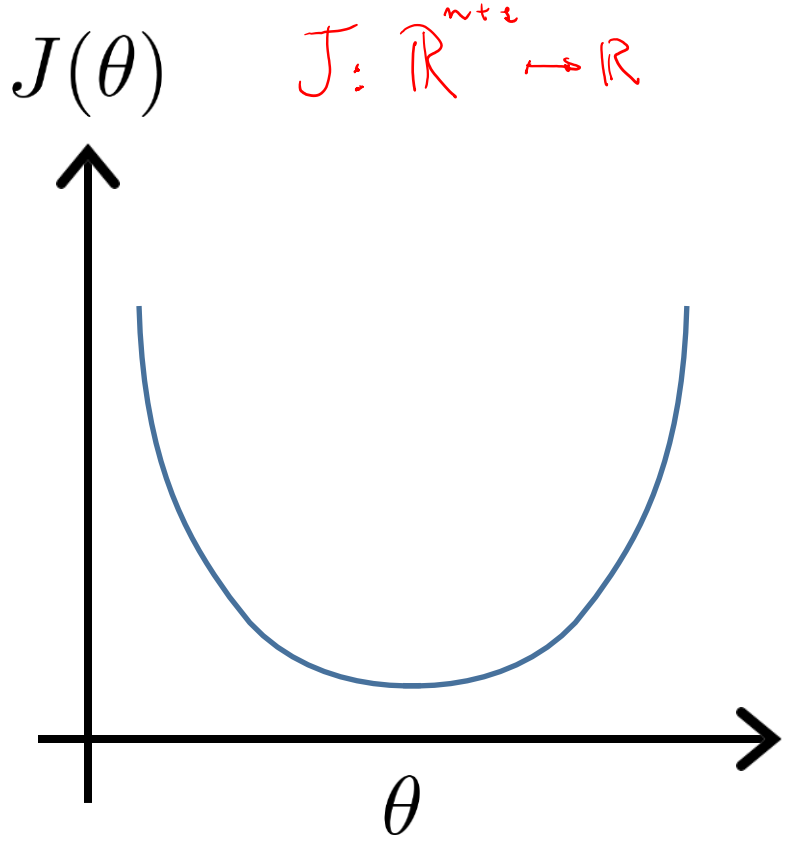
Normal Equation

$$\nabla f(p) \overset{!}{=} \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$

NABLA

$$f : \mathbb{R}^n \longmapsto \mathbb{R}$$

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$

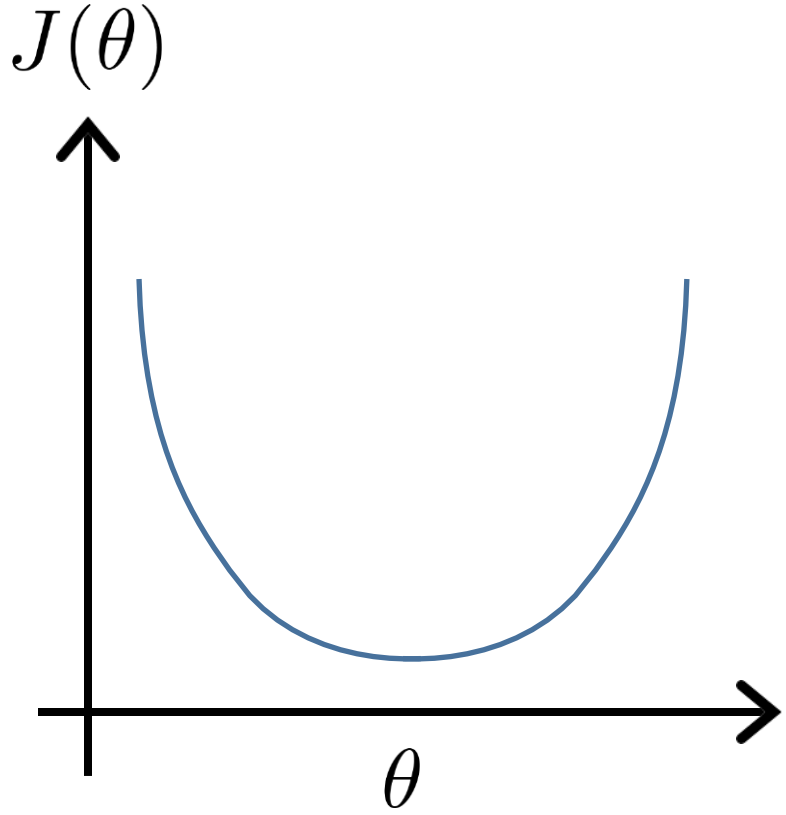


Gradient Descent

$$\Theta := \Theta - \alpha \nabla_\Theta J$$

$$\nabla_\Theta J = \begin{bmatrix} \frac{\delta J}{\delta \Theta_0} \\ \vdots \\ \frac{\delta J}{\delta \Theta_n} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$



Gradient Descent

$$\Theta := \Theta - \alpha \nabla_\Theta J \qquad \nabla_\Theta J = \begin{bmatrix} \frac{\delta J}{\delta \Theta_0} \\ \vdots \\ \frac{\delta J}{\delta \Theta_n} \end{bmatrix} \in \mathbb{R}^{n+1}$$

Normal equation: what about solving for Θ analytically?

Useful notation

- For $f: \mathbb{R}^{mxn} \mapsto \mathbb{R}$, define:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial}{\partial A_{11}} f & \cdots & \frac{\partial}{\partial A_{1n}} f \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial A_{m1}} f & \cdots & \frac{\partial}{\partial A_{mn}} f \end{bmatrix}$$

GRADIENT OF $f(A)$ W.R.T. A

$f(A) \in \mathbb{R}$, $A \in \mathbb{R}^{mxn}$

$\nabla_A f(A) \in \mathbb{R}^{m \times n}$

- Trace:
  - If $A \in \mathbb{R}^{nxn}$ $\quad \operatorname{tr} A = \sum_{i=1}^{n} A_{ii} \in \mathbb{R}$

Facts

- Some facts of matrix derivatives (without proof)

$\text{tr}\ AB = \text{tr}\ BA$

$\text{tr}\ ABC = \text{tr}\ CAB = \text{tr}\ BCA$

$f(A) = \text{tr}\ BA \in \mathbb{R} \qquad \nabla_A \text{tr}\ AB = B^T$

$\text{tr}\ A = \text{tr}\ A^T$

If $a \in \mathbb{R} \qquad \text{tr}\ a = a$

$\nabla_A \text{tr}\ ABA^T C = CAB + C^T AB^T$

m examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$; n features.

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

X = DESIGN MATRIX

$$= \begin{bmatrix} \text{——} \left(x^{(1)}\right)^T \text{——} \\ \text{——} \left(x^{(2)}\right)^T \text{——} \\ \vdots \\ \text{——} \left(x^{(m)}\right)^T \text{——} \end{bmatrix}$$

$m \times (n+1)$

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

Cost function

$$X\Theta - y = \begin{bmatrix} h\left(x^{(1)}\right) - y^{(1)} \\ \vdots \\ h\left(x^{(m)}\right) - y^{(m)} \end{bmatrix}$$

DESIGN MATRIX

where $\Theta^T x^{(i)} = \sum_{j=0}^{n} \Theta_j x_j^{(i)}$

Recall $z^T z = \sum_i z_i^2$

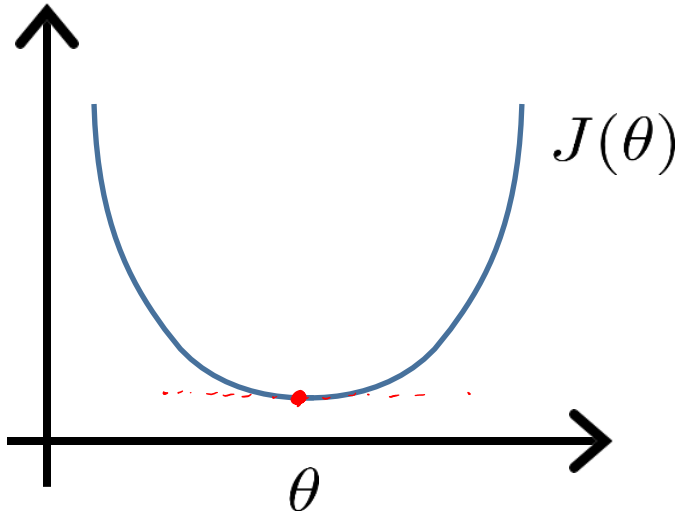
$$\frac{1}{2}(X\Theta - y)^T(X\Theta - y) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = J(\Theta) \qquad \theta \in \mathbb{R}^{n+1}$$

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

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

$\frac{d}{d\theta} J(\theta) \overset{\text{SET}}{=} 0$

Solve for θ



Solve for Θ analytically

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \quad \text{attained when} \quad \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$$

Expanding $J(\Theta)$:

$$\nabla_\Theta \frac{1}{2} (X\Theta - y)^T (X\Theta - y) \qquad tr\, a = a \qquad a^T = a$$

$$= \frac{1}{2} \nabla_\Theta tr \left(\Theta^T X^T X \Theta - \underbrace{\Theta^T X^T y}_{\text{SCALAR}} - y^T X \Theta + \underline{y^T y} \right)$$

$$= \frac{1}{2} \left[\nabla_\Theta tr\, \Theta \Theta^T X^T X - \nabla_\Theta tr\, y^T X \Theta - \nabla_\Theta tr\, y^T X \Theta \right]$$

Solve for Θ analytically

$$\frac{1}{2}\left[\nabla_\Theta \operatorname{tr}\, \Theta\Theta^T X^T X - \nabla_\Theta \operatorname{tr}\, y^T X\Theta - \nabla_\Theta \operatorname{tr}\, y^T X\Theta\right]$$

Recall $\nabla_A \operatorname{tr}\, ABA^T C = CAB + C^T AB^T$ $\nabla_A \operatorname{tr}\, AB = B^T$ $\operatorname{tr}\, AB = \operatorname{tr}\, BA$

$$\nabla_\Theta \operatorname{tr}\, \underbrace{\Theta}_{A} \underbrace{I}_{B} \underbrace{\Theta^T}_{A^T} \underbrace{X^T X}_{C} = \underbrace{X^T X}_{C} \underbrace{\Theta}_{A} \underbrace{I}_{B} + \underbrace{X^T X}_{C^T} \underbrace{\Theta}_{A} \underbrace{I}_{B^T}$$

$$\nabla_\Theta \operatorname{tr}\, \underbrace{y^T X\Theta}_{A} = X^T y$$

Solve for Θ analytically

$$\nabla_\theta J = \frac{1}{2}\left(X^T X\theta + X^T X\theta - 2X^T \vec{y}\right)$$
$$= X^T X\theta - X^T \vec{y} \stackrel{!}{=} 0$$

$$X^T X\theta = X^T \vec{y}$$ Normal equation

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

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

m

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \in \mathbb{R}^{m \times (n+1)}$$

DESIGN MATRIX

$n+1$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \in \mathbb{R}^{m}$$

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

m **training examples,** n **features.**

Gradient Descent

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

$n = 10^6$

Normal Equation

- No need to choose α.
- Don't need to iterate.
- Need to compute $(X^T X)^{-1}$ $n \times n$ $O(n^3)$
- Slow if n is very large.

$n = 100$ .. $n = 10{,}000$

Normal equation

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

- What if $X^T X$ is non--invertible? (singular/ degenerate)

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

- Redundant features (linearly dependent).
  E.g. x_1 =size in feet^2
  ~~x_2 =size in m^2~~

  $x_1 = (3.28)^2 x_2$

  1 m = 3,28 FEET

- Too many features (e.g. $m \leq n$).
  - Delete some features, or use regularization.

Outline

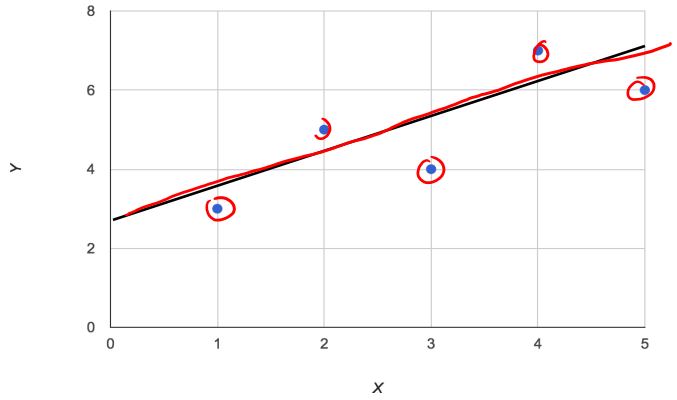
- Linear regression
  - One or multiple variables
  - Cost function
- Gradient descent, incl. batch/stochastic GD
- Normal equation
- MSE and Correlation
- Locally-Weighted Regression
- Probabilistic Interpretation of Least Squares

Linear Regression and Correlation

Summary So Far

- Given: Set of known (x,y) points
- Find: function $f(x)=ax+b$ that "best fits" the known points, i.e., $f(x)$ is close to y
- Use function to predict y values for new x's

➢ Also can be used to test correlation



Correlation and Causation

Correlation – Values track each other

- Height and Shoe Size
- Grades and Entrance Exam Scores

Causation – One value directly influences another

- Education Level → Starting Salary
- Temperature → Cold Drink Sales

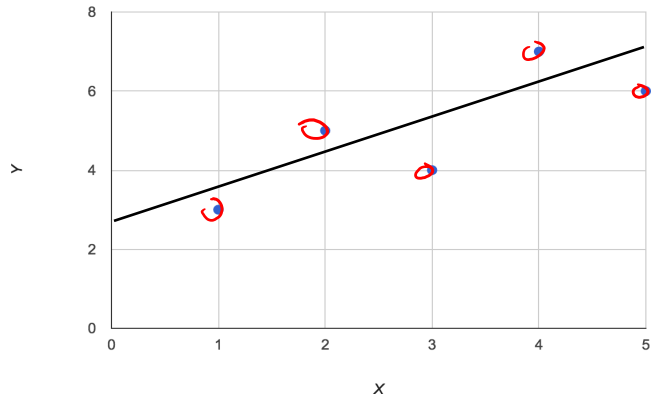
Correlation and Causation (from Overview)

Correlation – Values track each other

- Height and Shoe Size
- Grades and Entrance Exam Scores

Find: function $f(x)=ax+b$ that "best fits" the known points, i.e., $f(x)$ is close to y

The better the function fits the points, the more correlated x and y are



Regression and Correlation

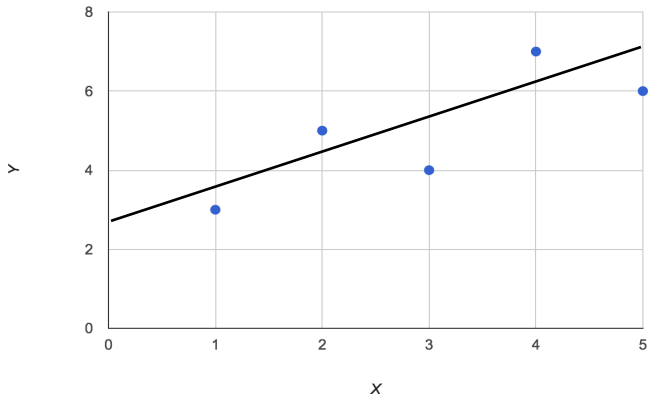
The better the function fits the points,
the more correlated *x* and *y* are

- Linear functions only
- Correlation – Values track each other

  Positively – when one goes up the other goes up
- Also negative correlation

  When one goes up the other goes down

  - Latitude versus temperature
  - Car weight versus gas mileage
  - Class absences versus final grade



Calculating Simple Linear Regression

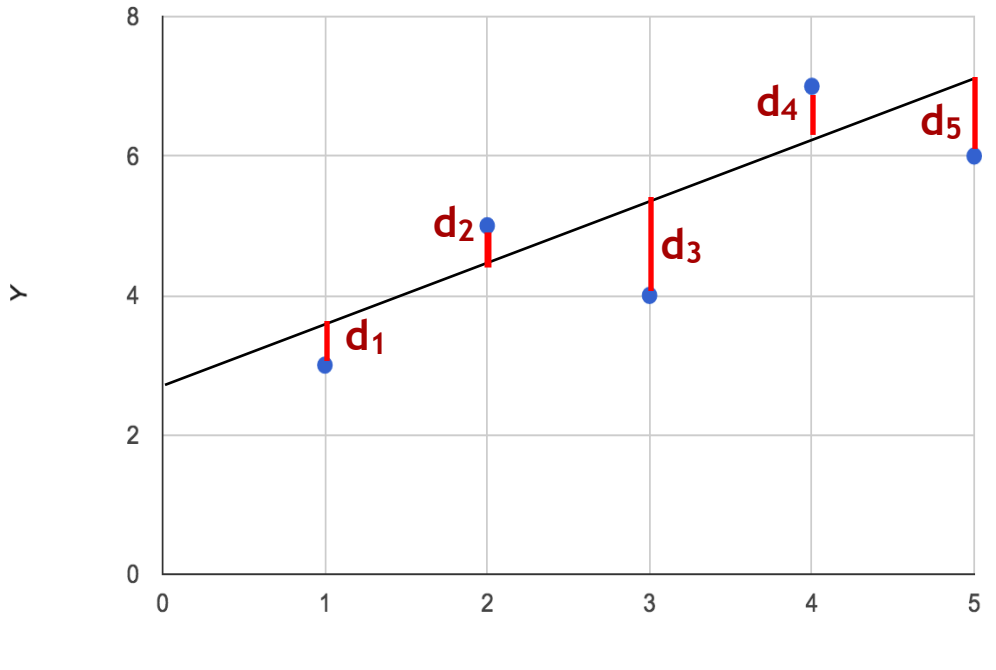
Method of least squares

- Given a point and a line, the error for the point is its vertical distance d from the line, and the squared error is d^2
- Given a set of points and a line, the sum of squared error (SSE) is the sum of the squared errors for all the points
- Goal: Given a set of points, find the line that minimizes the SSE

J

Calculating Simple Linear Regression

Method of least squares

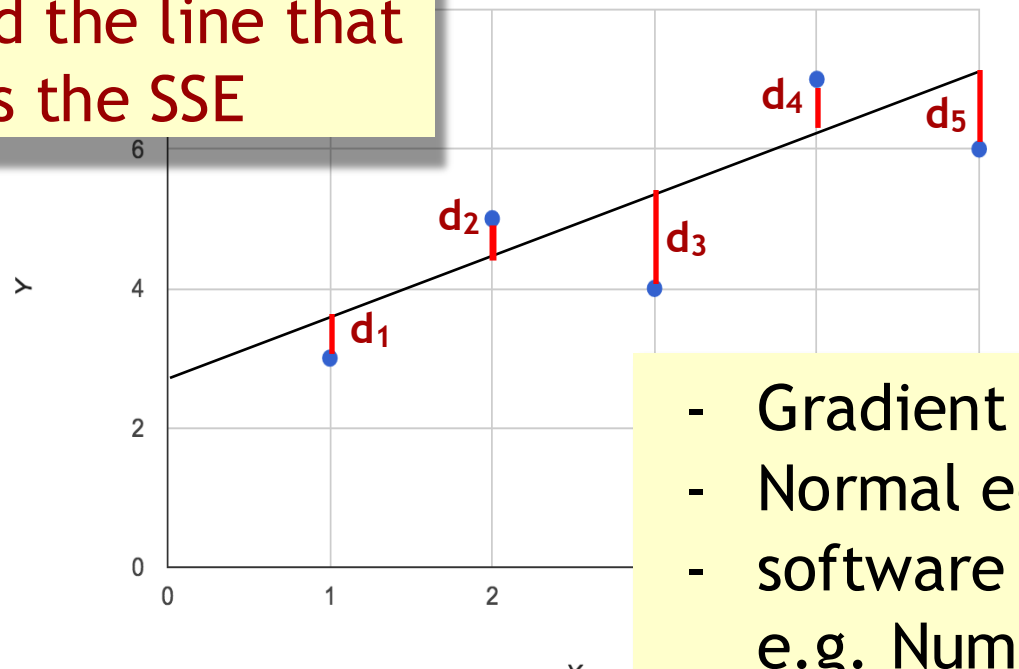


$$SSE = d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2$$

Calculating Simple Linear Regression

Method of least squares

Goal: Find the line that minimizes the SSE



- Gradient Descent
- Normal equation
- software packages, e.g. Numpy polyfit

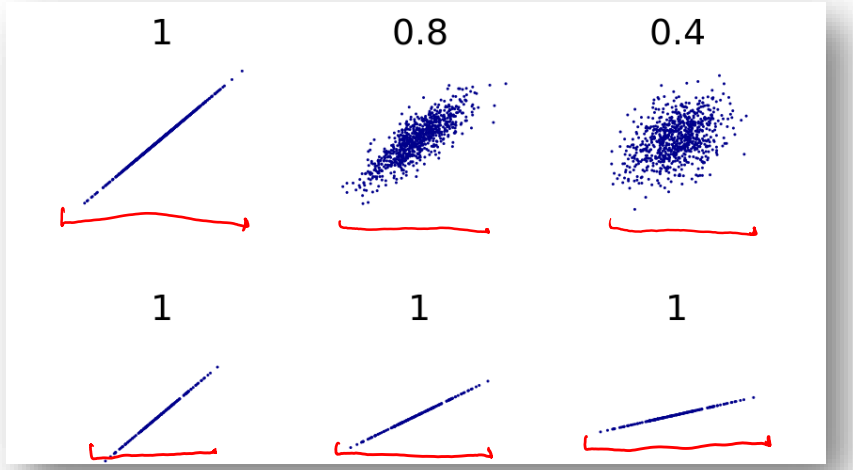
$$SSE = d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2$$

Measuring Correlation

More help from software packages…

Pearson's Product Moment Correlation (PPMC)

- "Pearson coefficient", "correlation coefficient"
- Value **r** between 1 and -1
  - 1 maximum positive correlation
  - 0 no correlation
  - -1 maximum negative correlation



Swapping x and y axes yields same values

Measuring Correlation

More help from software packages…

Pearson's Product Moment Correlation (PPMC)

- "Pearson coefficient", "correlation coefficient"
- Value **r** between 1 and -1
  - 1 maximum positive correlation
  - 0 no correlation
  - -1 maximum negative correlation

Coefficient of determination

"The better the function fits the points, the more correlated *x* and *y* are"

- r^2, R^2, "R squared"
- Measures fit of any line/curve to set of points
- Usually between 0 and 1
- For simple linear regression R^2 = Pearson^2

Correlation Game

http://aionet.eu/corguess (*)

Try to get:

Right answers ≥ 10, Guesses ≤ Right answers × 2

Anti-cheating: Pictures = Right answers + 1

(*) Improved version of “Wilderdom correlation guessing game” thanks to Poland participant Marcin Piotrowski

Other correlation games:

http://guessthecorrelation.com/

http://www.rossmanchance.com/applets/GuessCorrelation.html

http://www.istics.net/Correlations/

Outline

- Linear regression
  - One or multiple variables
  - Cost function
- Gradient descent, incl. batch/stochastic GD
- Normal equation
- MSE and Correlation
- Locally-Weighted Regression
- Probabilistic Interpretation of Least Squares

Locally-Weighted Regression

Recap

m **examples** $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ **;** n **features.**

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

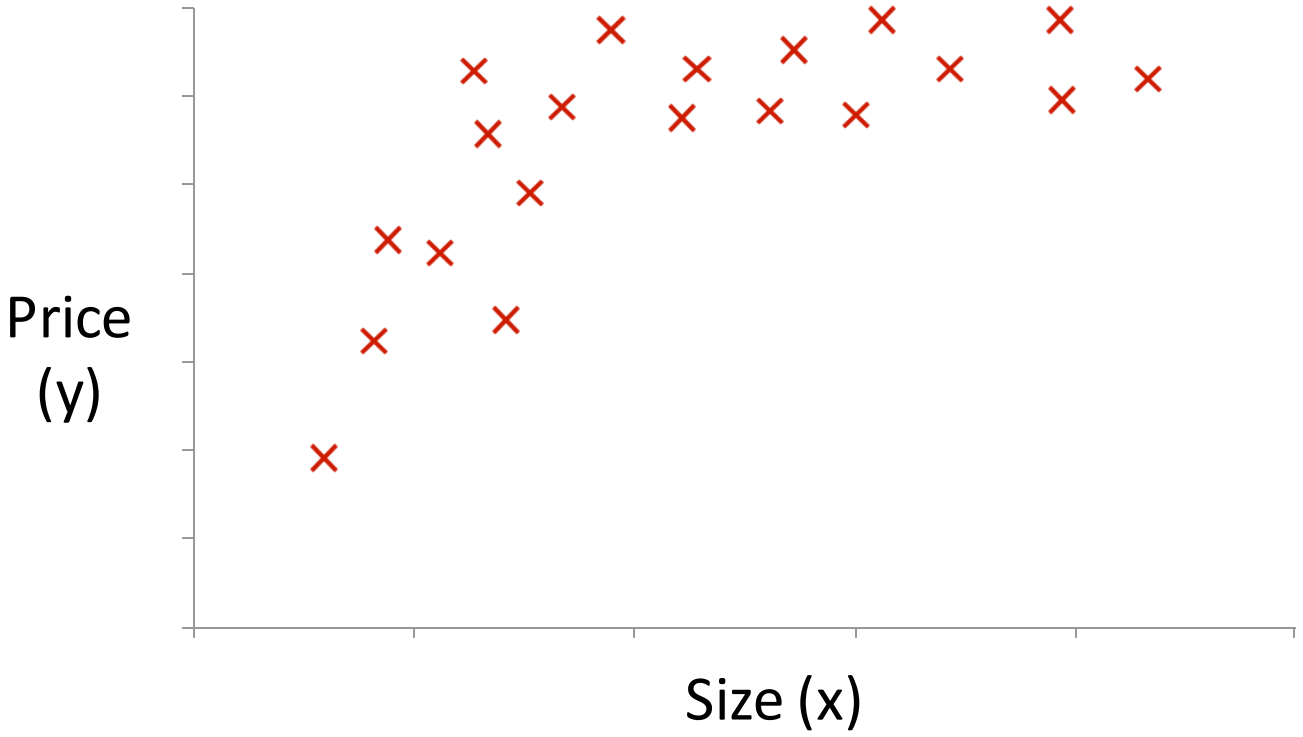
$x^{(i)}$ FEATURES FOR i-th TRAINING EXAMPLE

$h_\theta(x^{(i)})$ PREDICTION FOR i-th TRAINING EXAMPLE

x NEW EXAMPLE (TEST)

$$h_\Theta(x) = \sum_{j=0}^{n} \Theta_j x_j = \Theta^T x \qquad J(\Theta) = \frac{1}{2} \sum_{i=1}^{m} \left(h_\Theta(x^{(i)}) - y^{(i)} \right)^2$$

Recap: choice of features



$$\Theta_0 + \Theta_1 x_1$$

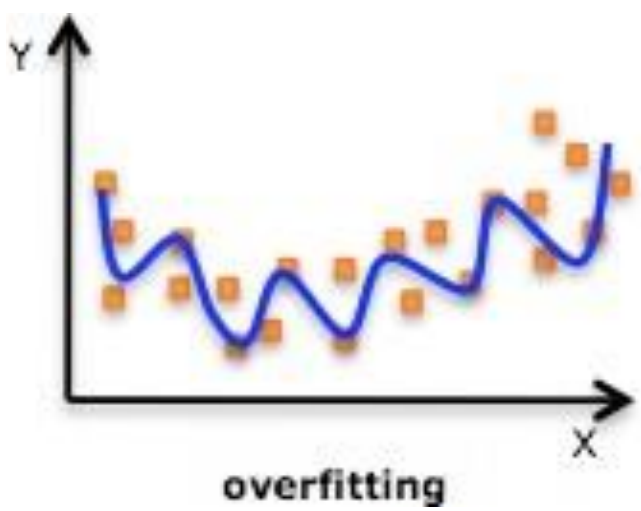
$$\Theta_0 + \Theta_1 x + \Theta_2 x^2$$

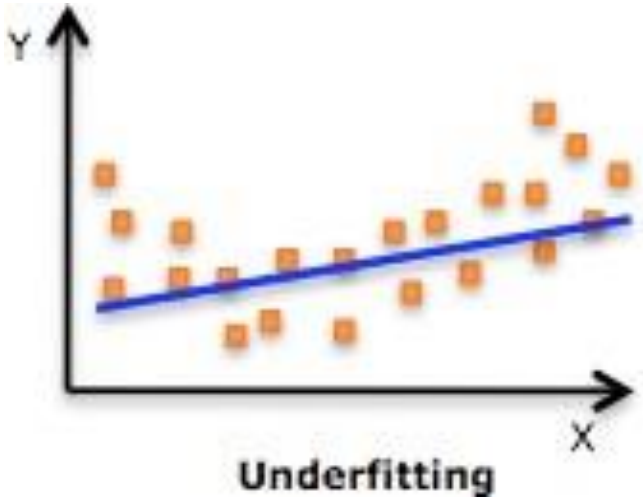
$$\Theta_0 + \Theta_1 x + \Theta_2 \sqrt{x} + \Theta_3 \log(x)$$

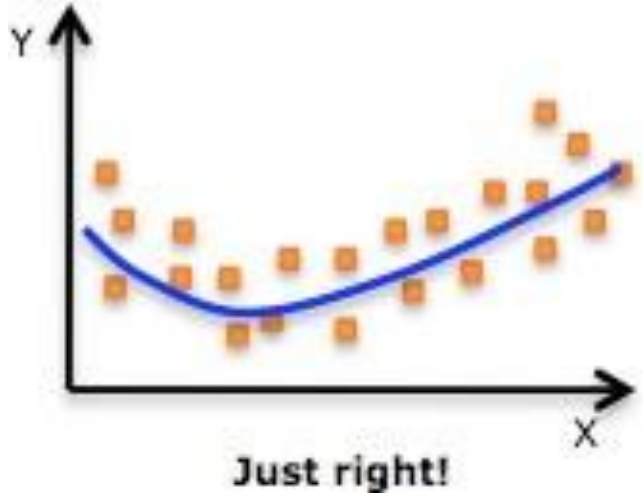
x_1 x_2 x_3

Recap: dangers of polynomial regression

Overfitting and Underfitting



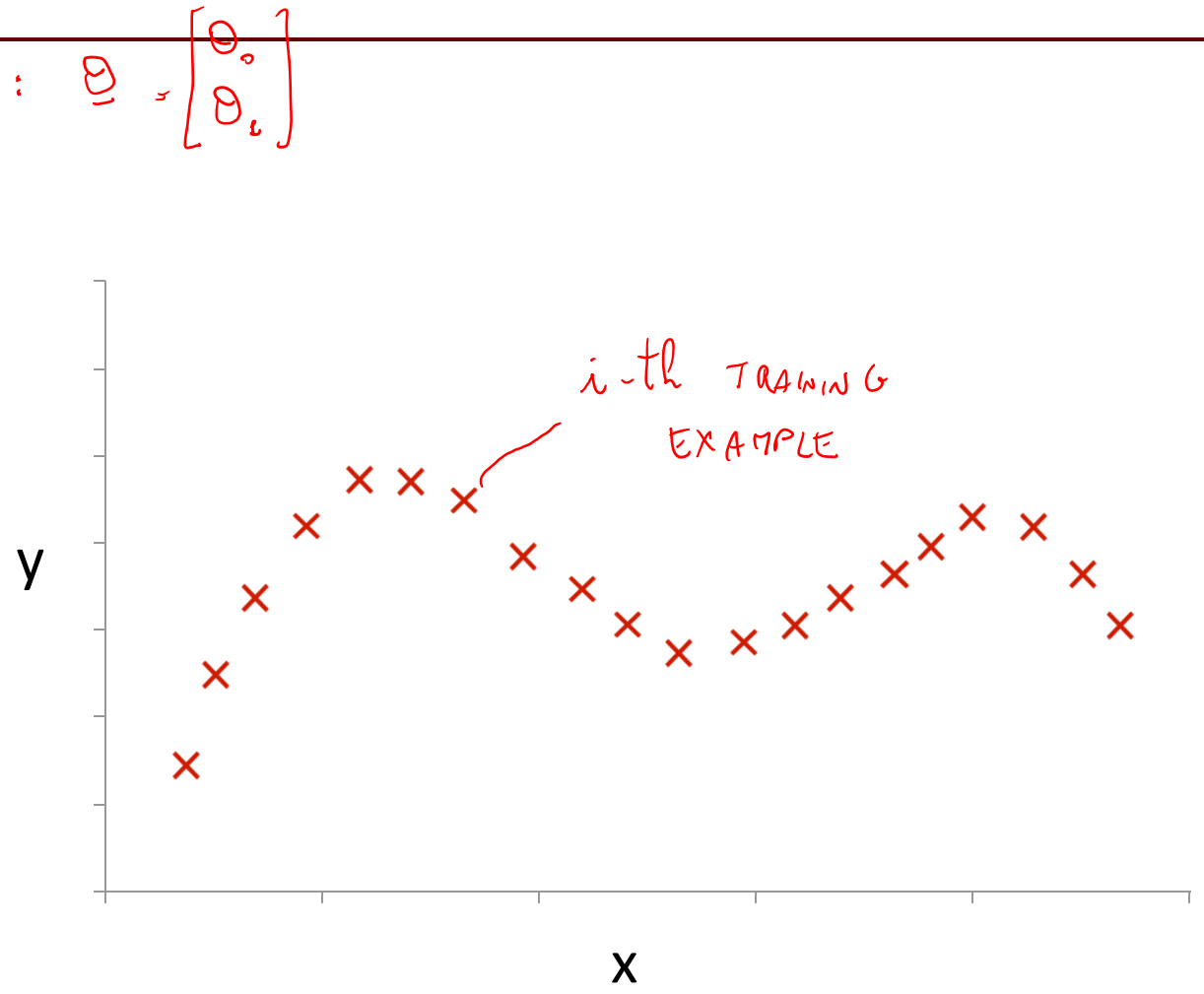




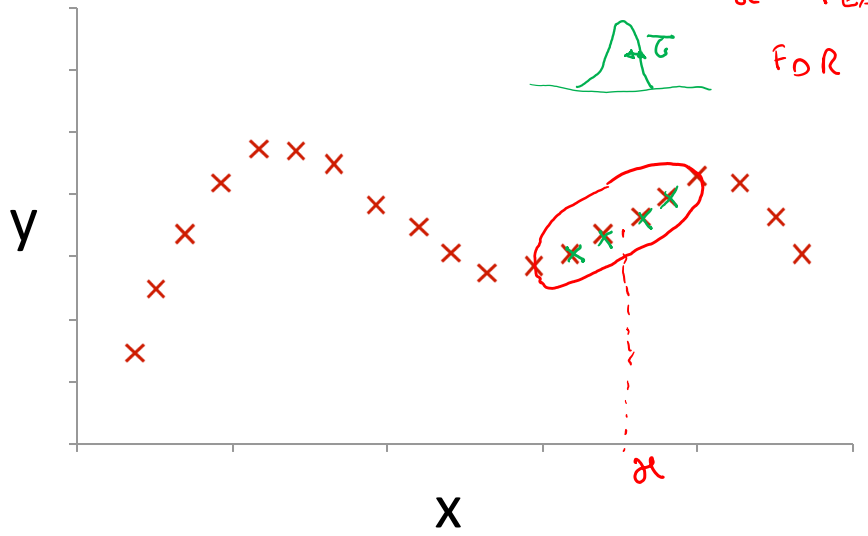
Locally-weighted regression

- "Parametric learning algorithm": fixed set of parameters
- "Non-parametric learning algorithm": parameters grow with the data (more data to keep in memory)

- Locally-weighted regression
  - also named Loess or Lowess



Locally-weighted regression



x FEATURE VECTOR FOR NEW EXAMPLE

τ

LR:

TO EVALUATE h AT CERTAIN x

FIT θ TO MINIMIZE

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

RETURN $\theta^T x$

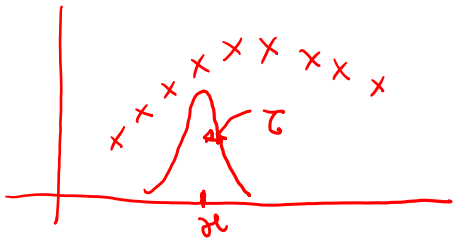
LOCALLY WEIGHTED REGRESSION

FIT θ TO MINIMIZE

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

$$w^{(i)} = \exp\left(-\frac{\|x^{(i)}-x\|^2}{2\tau^2}\right)$$

τ = BANDWIDTH



IF $\|x^{(i)}-x\|$ SMALL $w^{(i)} = 1$

$\|x^{(i)}-x\|$ LARGE $w^{(i)} \approx 0$

Outline

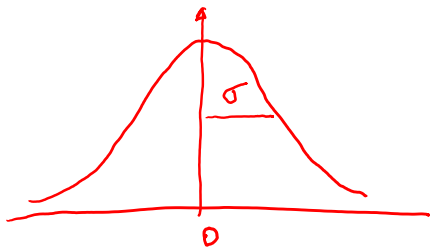
- Linear regression
  - One or multiple variables
  - Cost function
- Gradient descent, incl. batch/stochastic GD
- Normal equation
- MSE and Correlation
- Locally-Weighted Regression
- Probabilistic Interpretation of Least Squares

Probabilistic Interpretation of Least Squares

Why Least Squares

- Assume $y^{(i)} = \Theta^T x^{(i)} + \epsilon^{(i)}$ ← "error": UNMODELLED EFFECTS, RANDOM NOISE

$$\epsilon^{(i)} \sim N(0, \sigma^2) \qquad p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(\frac{-(\epsilon^{(i)})^2}{2\sigma^2}\right)$$



ASSUMPTION: $\epsilon^{(i)}$ ARE IID

$$p(y^{(i)} \mid x^{(i)}; \Theta) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(y^{(i)} - \Theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$y^{(i)}$: RANDOM VARIABLE; $\Theta^T x^{(i)}$: MEAN

PARAMETRIZED BY

$$y^{(i)} \mid x^{(i)}; \Theta \sim N(\Theta^T x^{(i)}, \sigma^2)$$

Why Least Squares Contd

- Likelihood $L(\Theta) = P(\vec{y}|X;\Theta)$

$$\mathcal{L}(\theta) = P(\vec{y} \mid X;\theta)$$

$$= \prod_{i=1}^{m} P\left(y^{(i)} \mid x^{(i)};\theta\right)$$

$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right)$$

LOG LIKELIHOOD

$$\ell(\theta) = \log \mathcal{L}(\theta)$$

$$= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\,\sigma} \exp(\dots)$$

$$= \sum_{i=1}^{m} \left[\log \frac{1}{\sqrt{2\pi}\,\sigma} + \log \exp(\dots)\right]$$

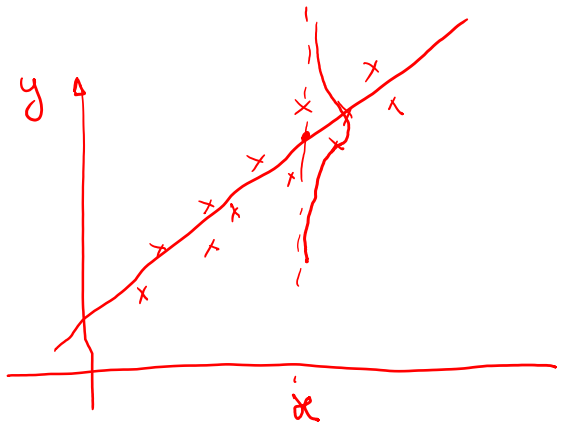
$$= m \log \frac{1}{\sqrt{2\pi}\,\sigma} + \sum_{i=1}^{m} -\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}$$

MLE : MAXIMUM LIKELIHOOD ESTIMATION

CHOOSE θ TO MAXIMIZE $\mathcal{L}(\theta)$

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

More on the Probabilistic Interpretation



$$p(y \mid x, \theta) = \frac{1}{\sqrt{2\pi}\,\sigma} \, e^{-\frac{y - \theta^T x}{2\sigma^2}}$$

References

- Chapter 3.1 in [Bishop, 2006. Pattern Recognition and Machine Learning]
- Chapter 9 in [Deisenroth Faisal Ong Book 2020 Mathematics for Machine Learning]

Thank you

Acknowledges: slides and material from Andrew Ng, Eric Xing, Matthew R. Gormley, Jessica Wu

SAPIENZA
UNIVERSITÀ DI ROMA