Linear Regression

Machine Learning

(Largely based on slides from Andrew Ng)

Prof. Sandra Avila

Institute of Computing (IC/Unicamp)

@sandraavilabr

MC886/MO444, August 30, 2022

House Price Prediction

$ 70 000

https://www.youtube.com/watch?v=IpGxLWOIZy4

House Price Prediction

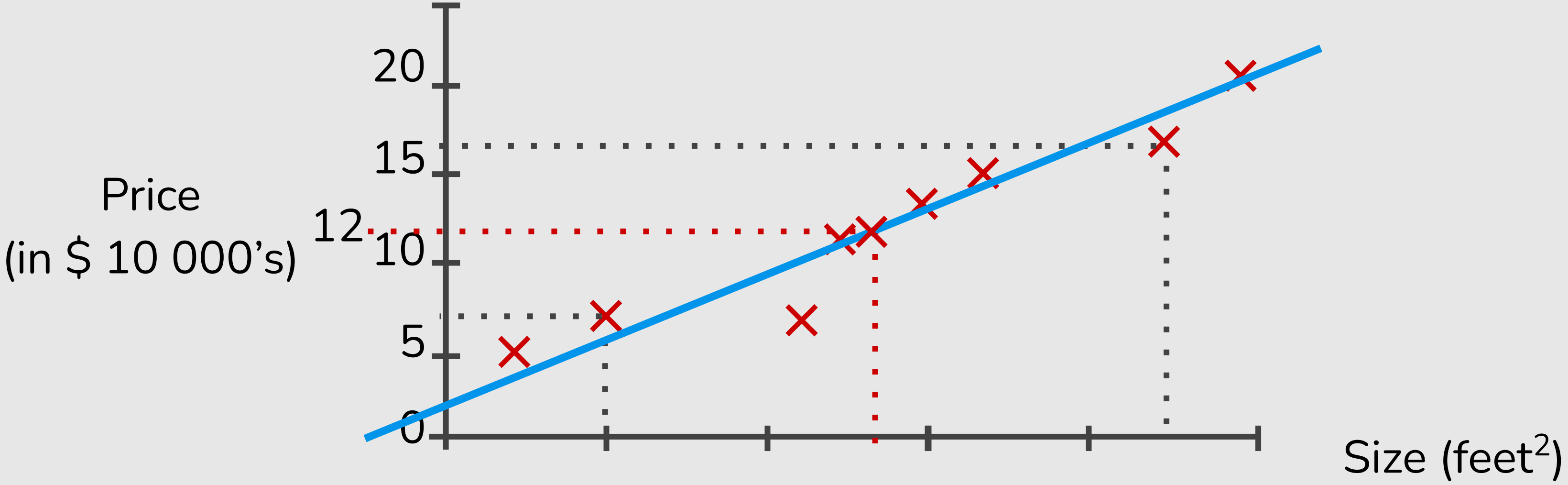
$ 160 000

House Price Prediction

???

House Price Prediction

Linear Regression





kaggle

Search

+ Create

- Home
- Competitions
- Datasets
- Code
- Discussions
- Courses
- More

GettingStarted Prediction Competition

House Prices - Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting

Kaggle · 3,840 te ms · Ongoing

Overview | Data | Code | Discussion | Leaderboard | Rules | Join Competition

Overview

- Description
- Evaluation
- Tutorials
- Frequently Asked Questions

Start here if...

You have some experience with R or Python and machine learning basics. This is a perfect competition for data science students who have completed an online course in machine learning and are looking to expand their skill set before trying a featured competition.

Competition Description

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview kaggle

Search

Create

Home

Competitions

Datasets

Code

Discussions

Courses

More

GettingStarted Prediction Competition

House Prices - Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting

Kaggle · 3,840 teams · Ongoing

Overview Data Code Discussion Leaderboard Rules

Join Competition

Leaderboard

Raw Data Refresh

Search leaderboard

This leaderboard is calculated with all of the test data.

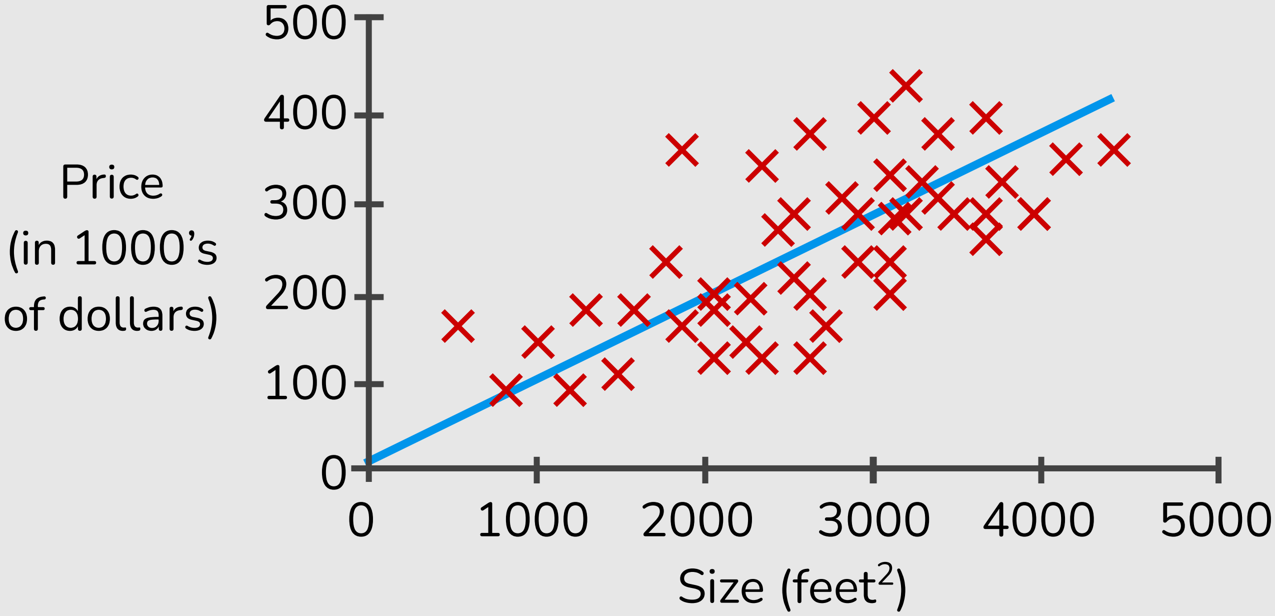
#	Team	Members	Score	Entries	Last	Code	Join
1	fedesoriano		0.00000	2	23d		
2	Lev1nLee		0.00000	1	2d		
3	Moshi Wei		0.00044	3	2mo		
4	Jewel Liu		0.00044	1	2mo		
5	YIYANG HAO		0.00044	13	6d		
6	Bing Guo		0.00044	4	2mo		

Today's Agenda

- Linear Regression with One Variable
  - Model Representation
  - Cost Function
  - Gradient Descent
- Linear Regression with Multiple Variables
  - Gradient Descent for Multiple Variables
  - Feature Scaling
  - Learning Rate
  - Features and Polynomial Regression
  - Normal Equation

Model Representation

Housing Prices



Supervised Learning

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

Regression Problem

Predict real-valued output

Training set of housing prices

Size in feet^2 (x)	**Price ($\$$) in 1000's (y)**
2104	460
1416	232
1534	315
852	178
...	...

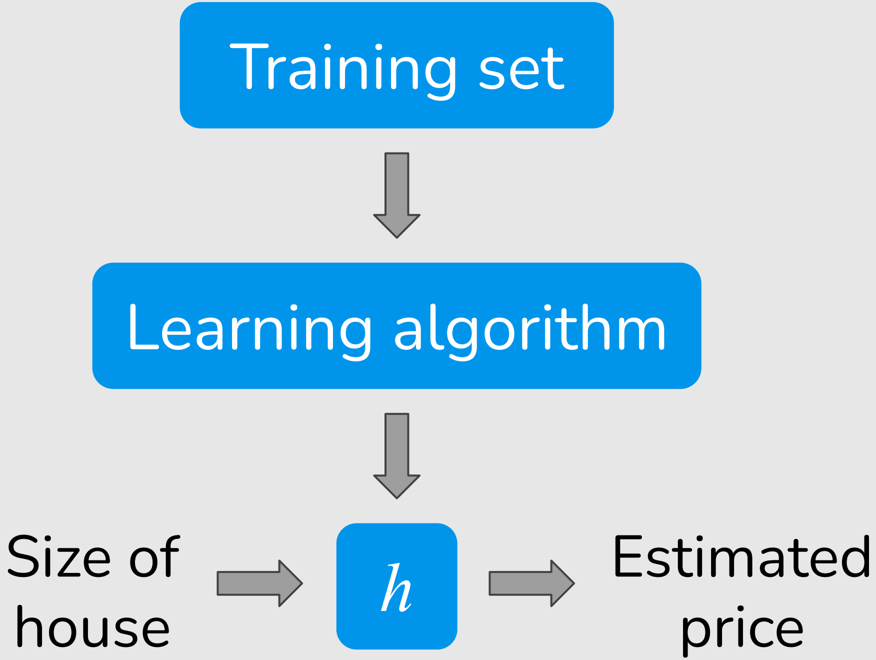
Notation:

m = Number of training examples

x's = "input" variable / features

y's = "output" variable / "target" variable

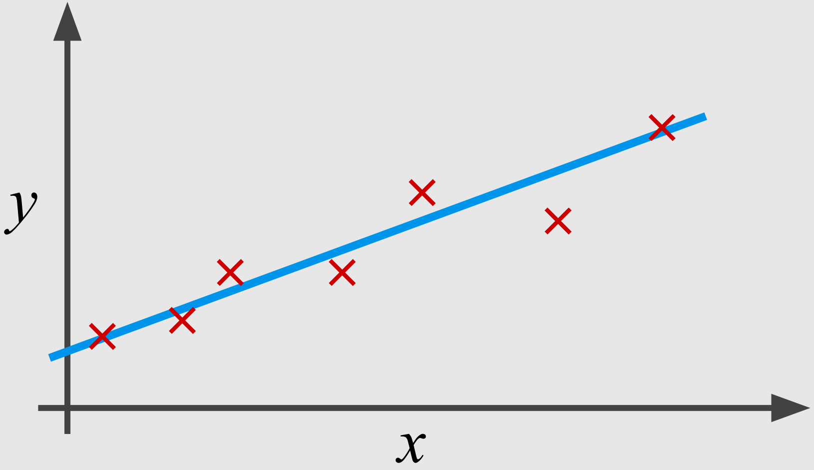
How do we represent h ?



(hypothesis)

h maps x's to y's

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



Linear regression with one variable.
Univariate linear regression.

Cost Function

Training Set

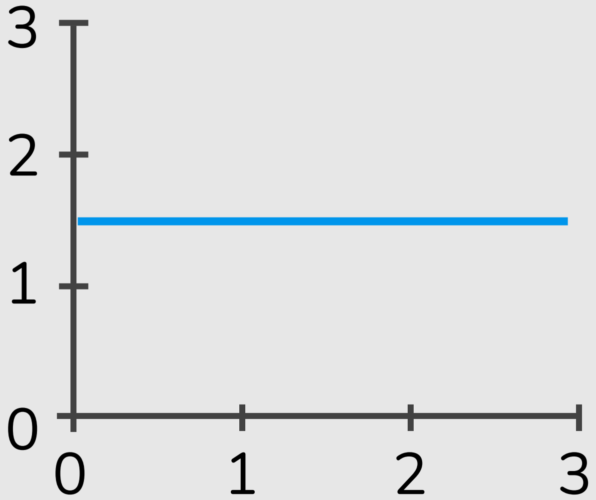
Size in feet^2 (x)	**Price (\$) in 1000's (y)**
2104	460
1416	232
1534	315
852	178
...	...

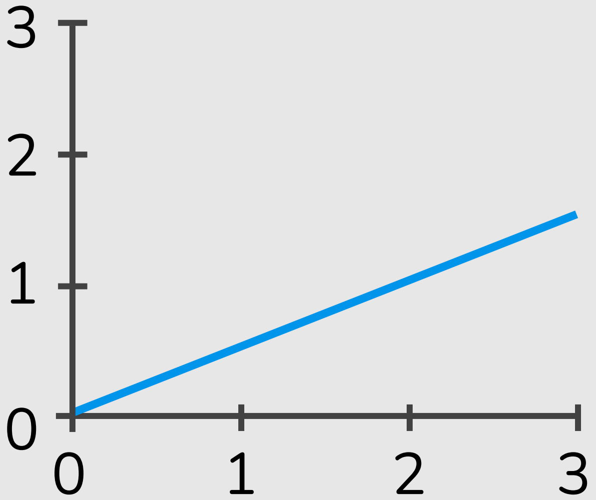
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

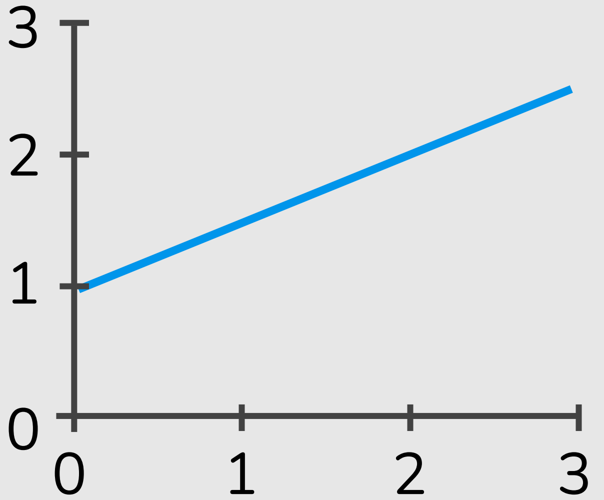
θi's: Parameters

How to choose θi's ?

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



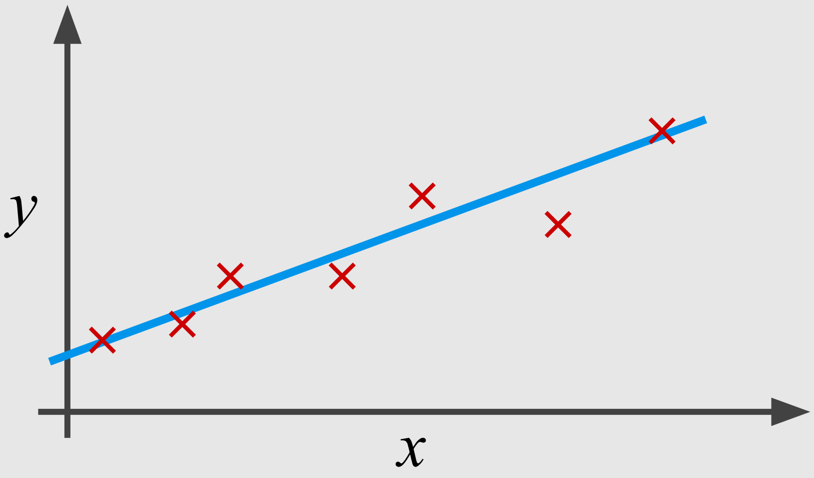




$\theta_0 = 1.5$
$\theta_1 = 0$

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

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



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

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

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

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

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

Cost function
(Squared error function)

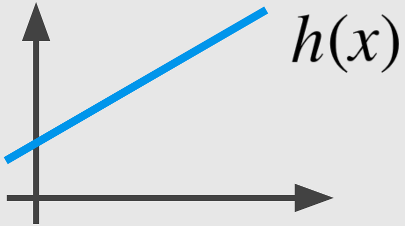
Cost Function Intuition I

Hypothesis:

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

Parameters:

$$\theta_0 , \theta_1$$



Cost Function:

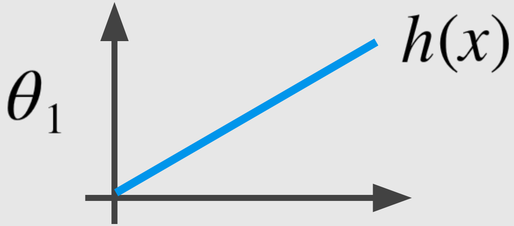
$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Goal:

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

Simplified

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



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

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

$$h_\theta(x)$$

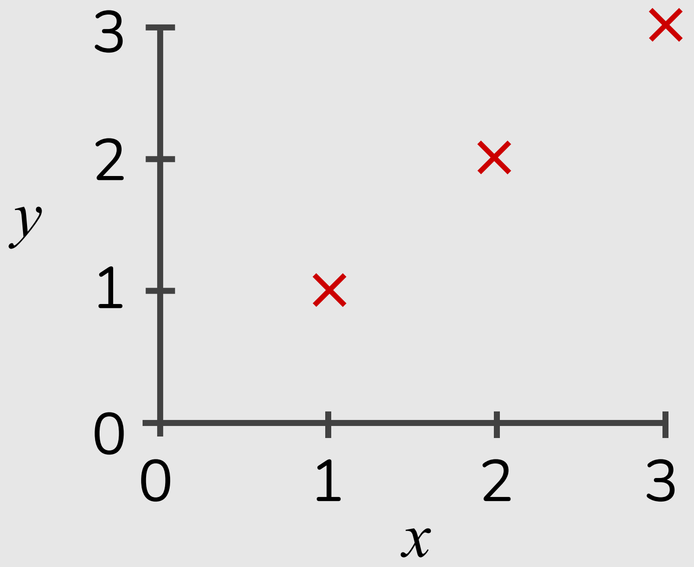
(for fixed θ_1, this is a function of x)

$$J(\theta_1)$$

(function of the parameters θ_1)

$h_\theta(x)$

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

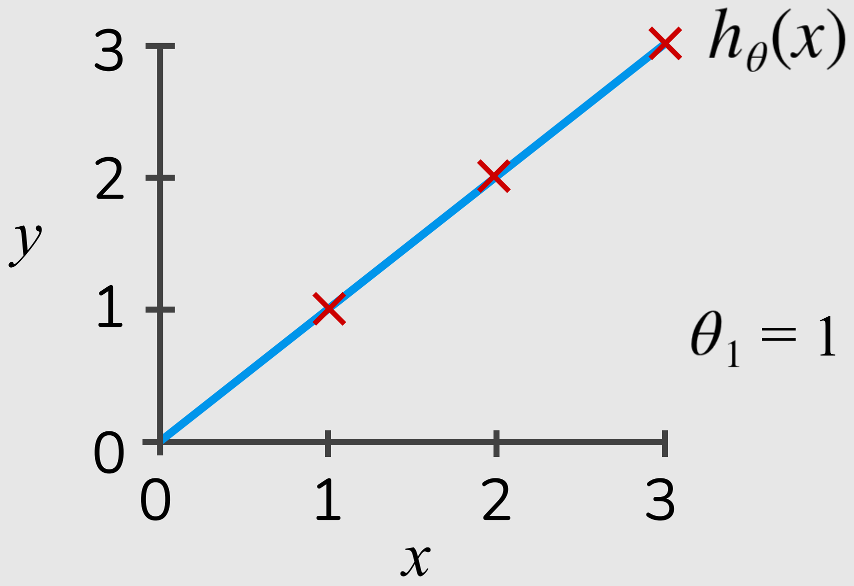


$J(\theta_1)$

(function of the parameters θ_1)

$h_\theta(x)$

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



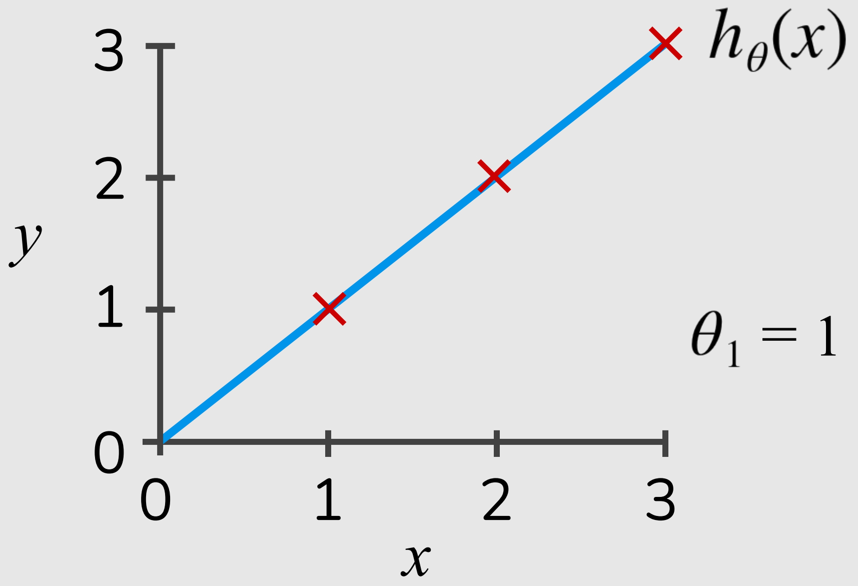
$J(\theta_1) = J(1) = ?$

$J(\theta_1)$

(function of the parameters θ_1)

$h_\theta(x)$

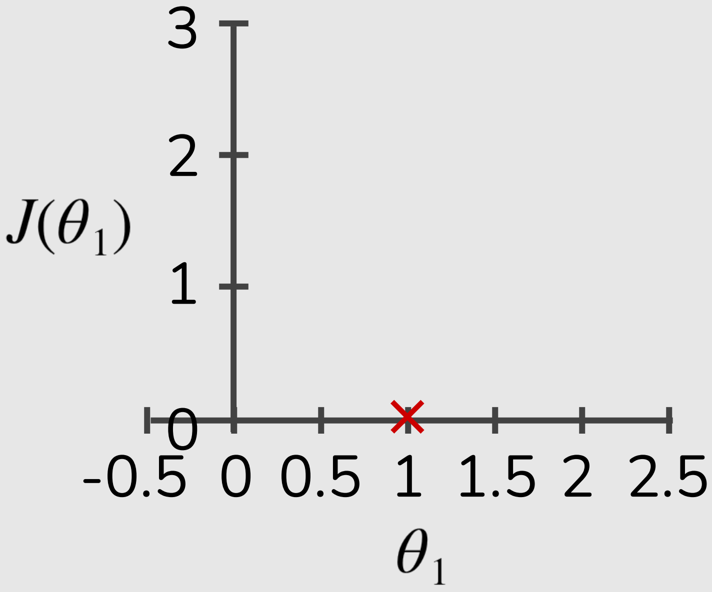
(for fixed θ_1, this is a function of x)



$J(\theta_1) = J(1) = 0$

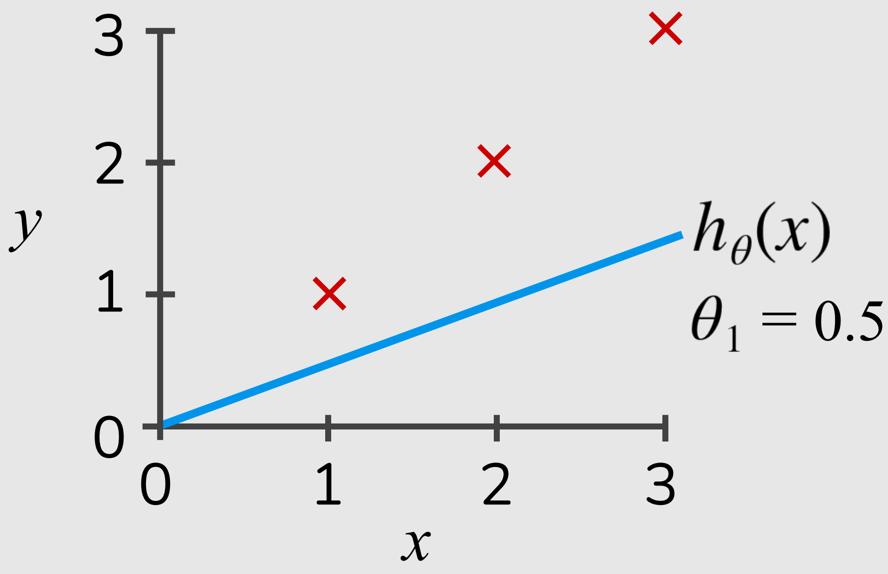
$J(\theta_1)$

(function of the parameters θ_1)



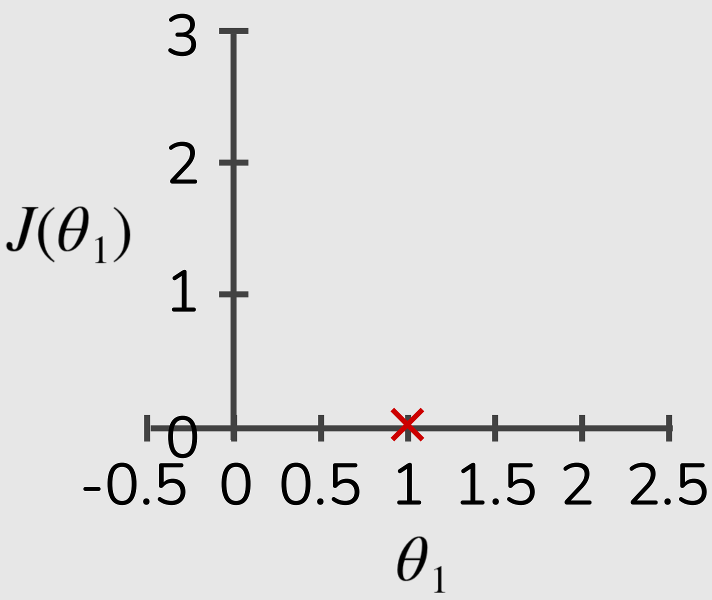
$h_\theta(x)$

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



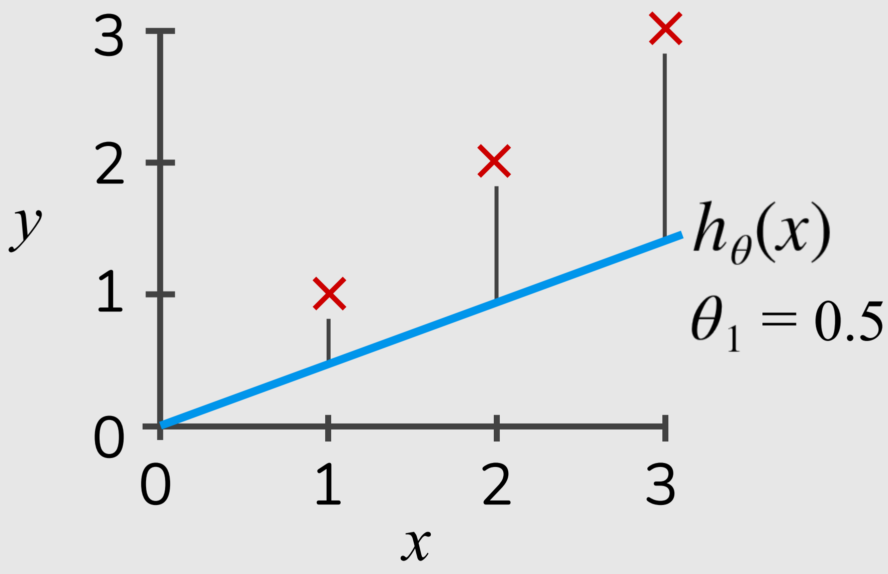
$J(\theta_1)$

(function of the parameters θ_1)



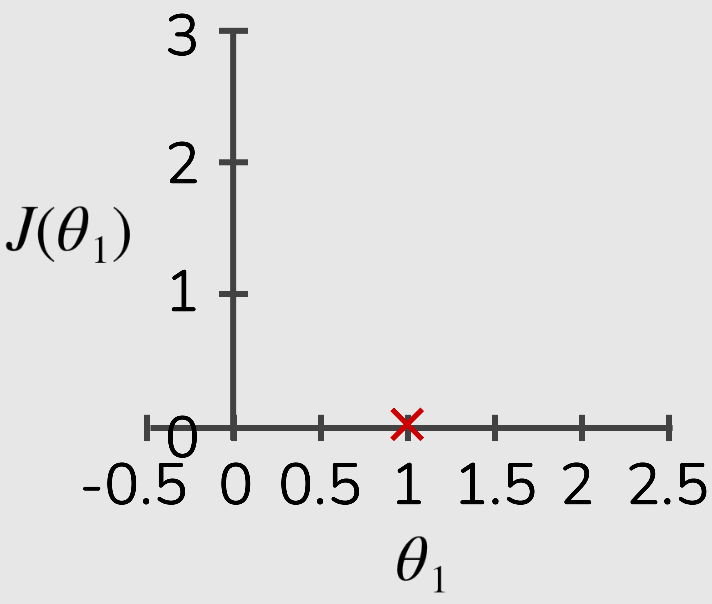
$h_\theta(x)$

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



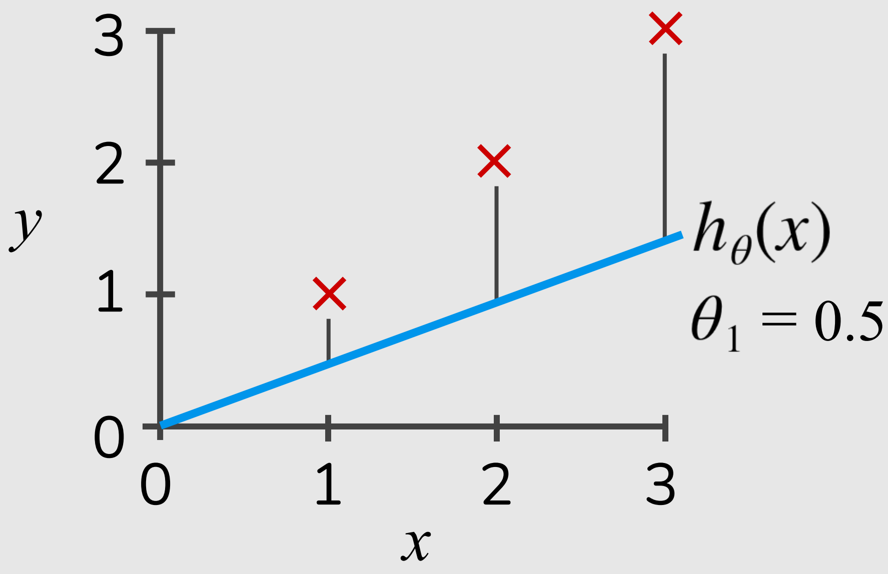
$J(\theta_1)$

(function of the parameters θ_1)



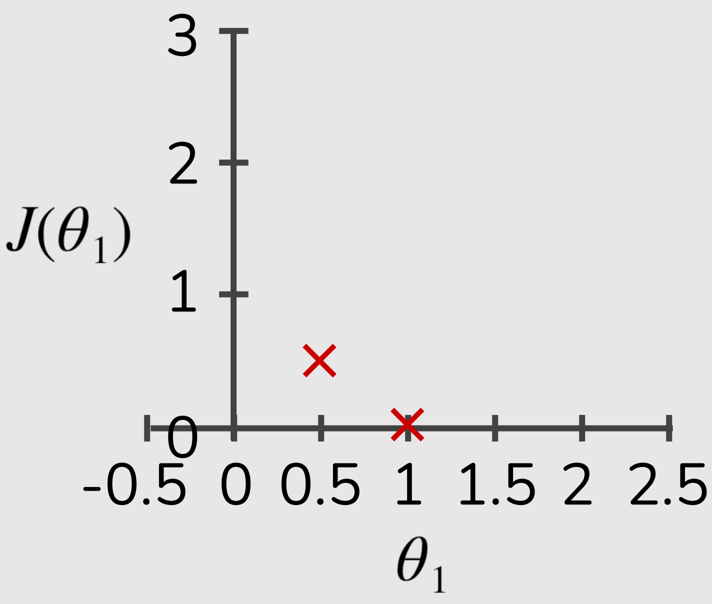
$h_\theta(x)$

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



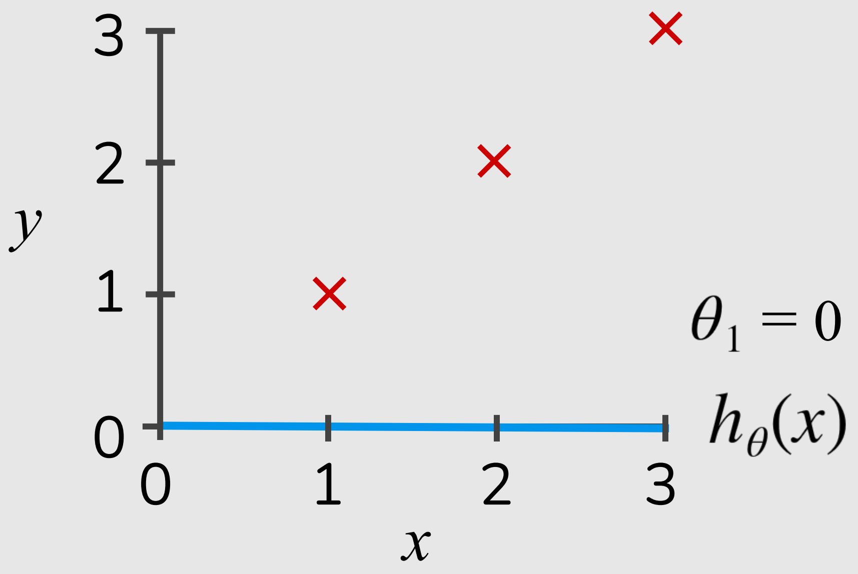
$J(\theta_1)$

(function of the parameters θ_1)



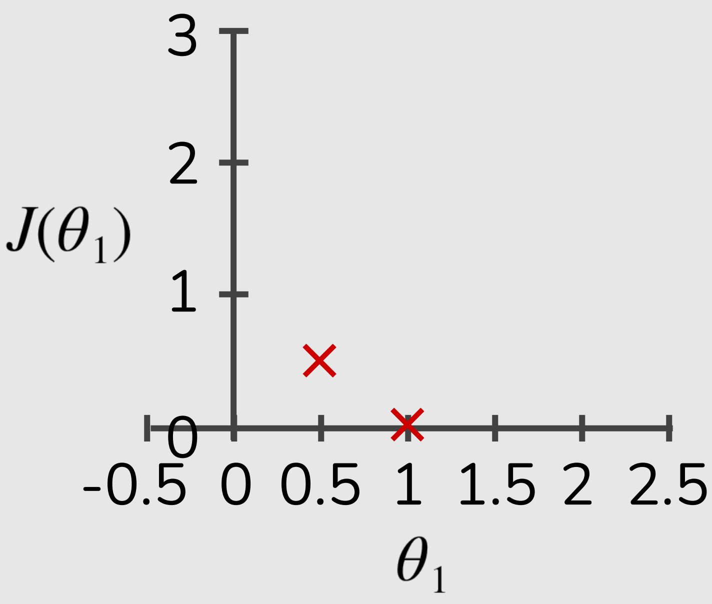
$h_\theta(x)$

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



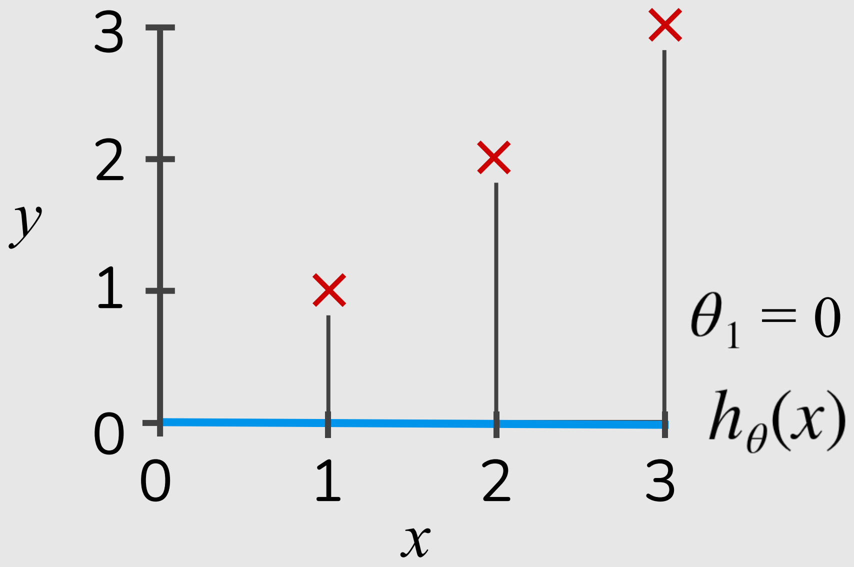
$J(\theta_1)$

(function of the parameters θ_1)



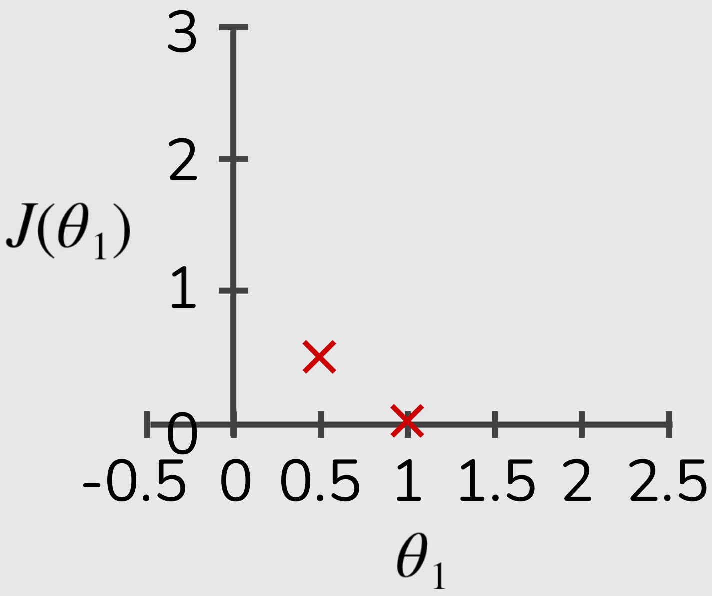
$h_\theta(x)$

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



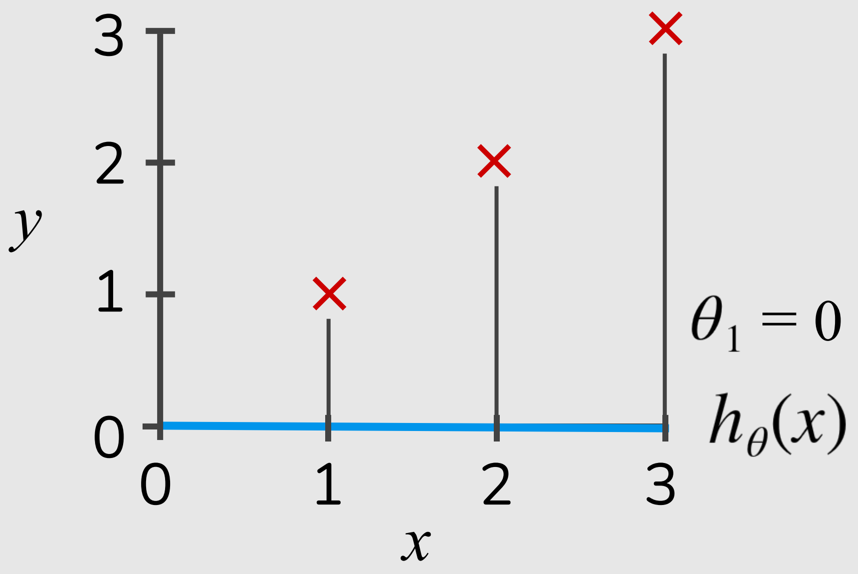
$J(\theta_1)$

(function of the parameters θ_1)



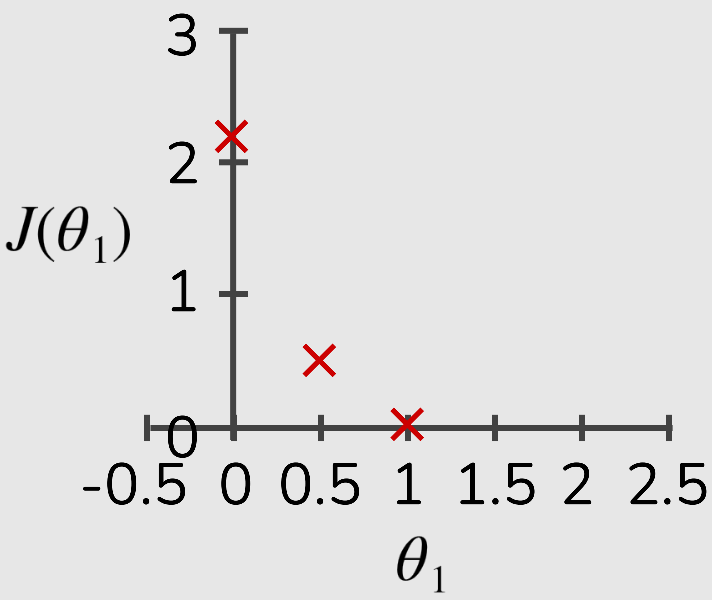
$h_\theta(x)$

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



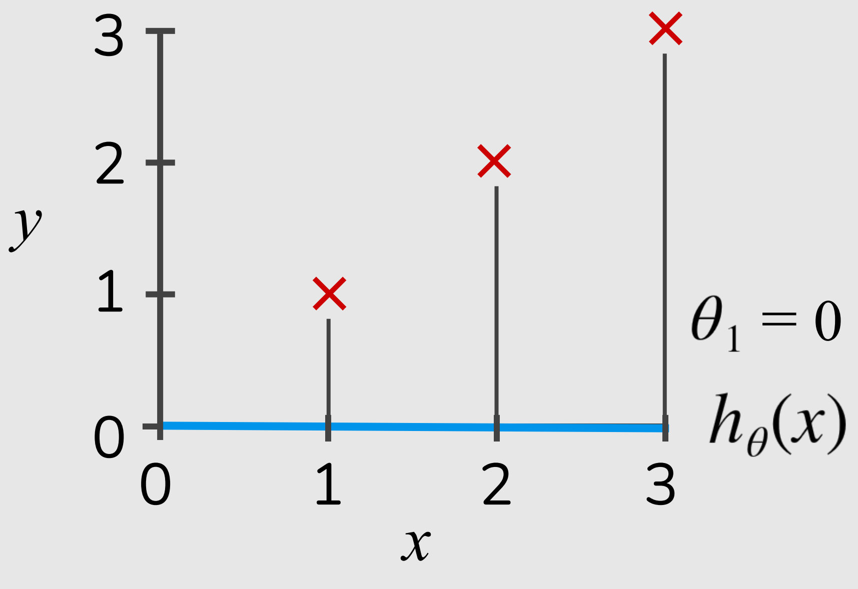
$J(\theta_1)$

(function of the parameters θ_1)



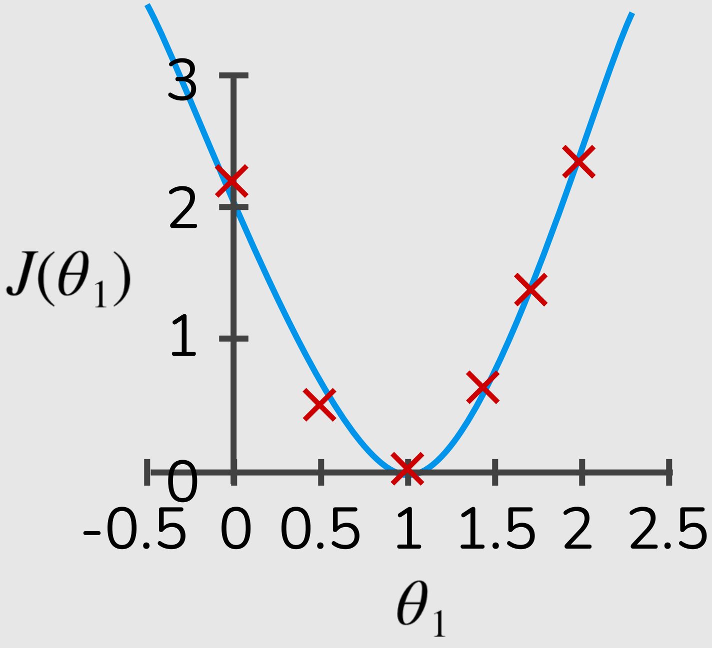
$h_\theta(x)$

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



$J(\theta_1)$

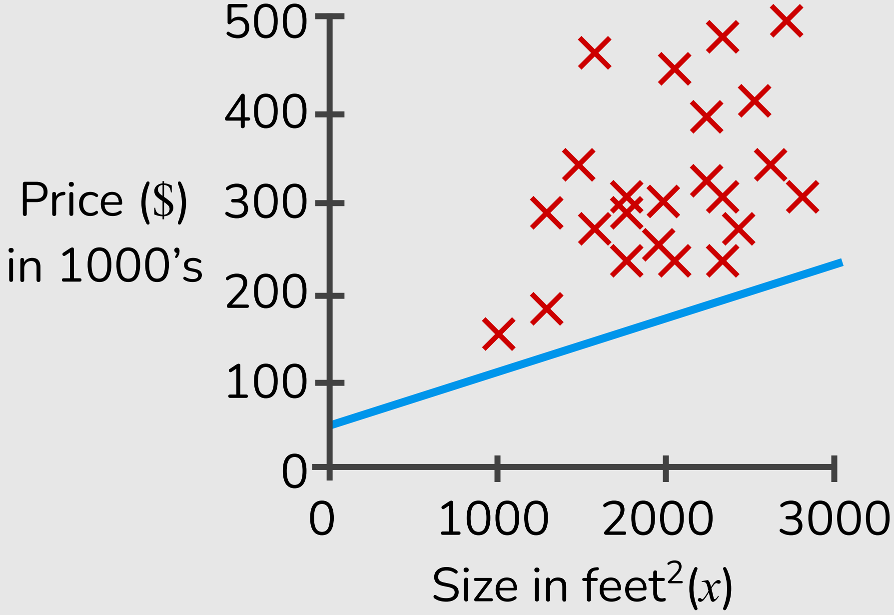
(function of the parameters θ_1)



Cost Function Intuition II

$h_\theta(x)$

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



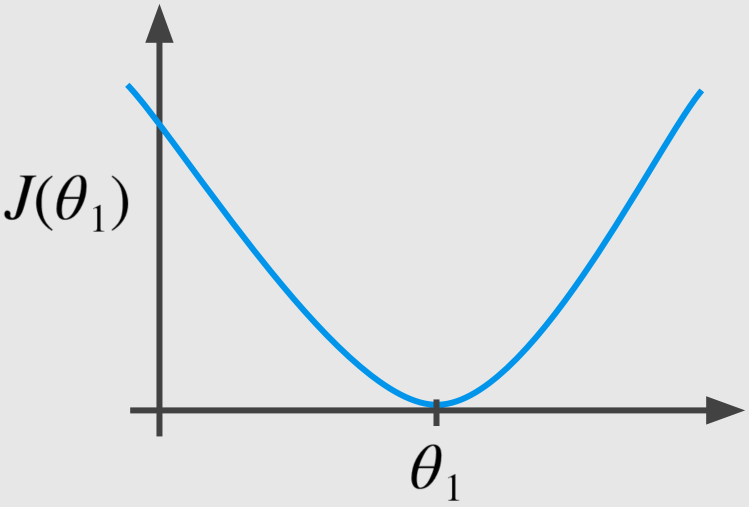
$h_\theta(x) = 50 + 0.06x$

$\theta_0 = 50$

$\theta_1 = 0.06$

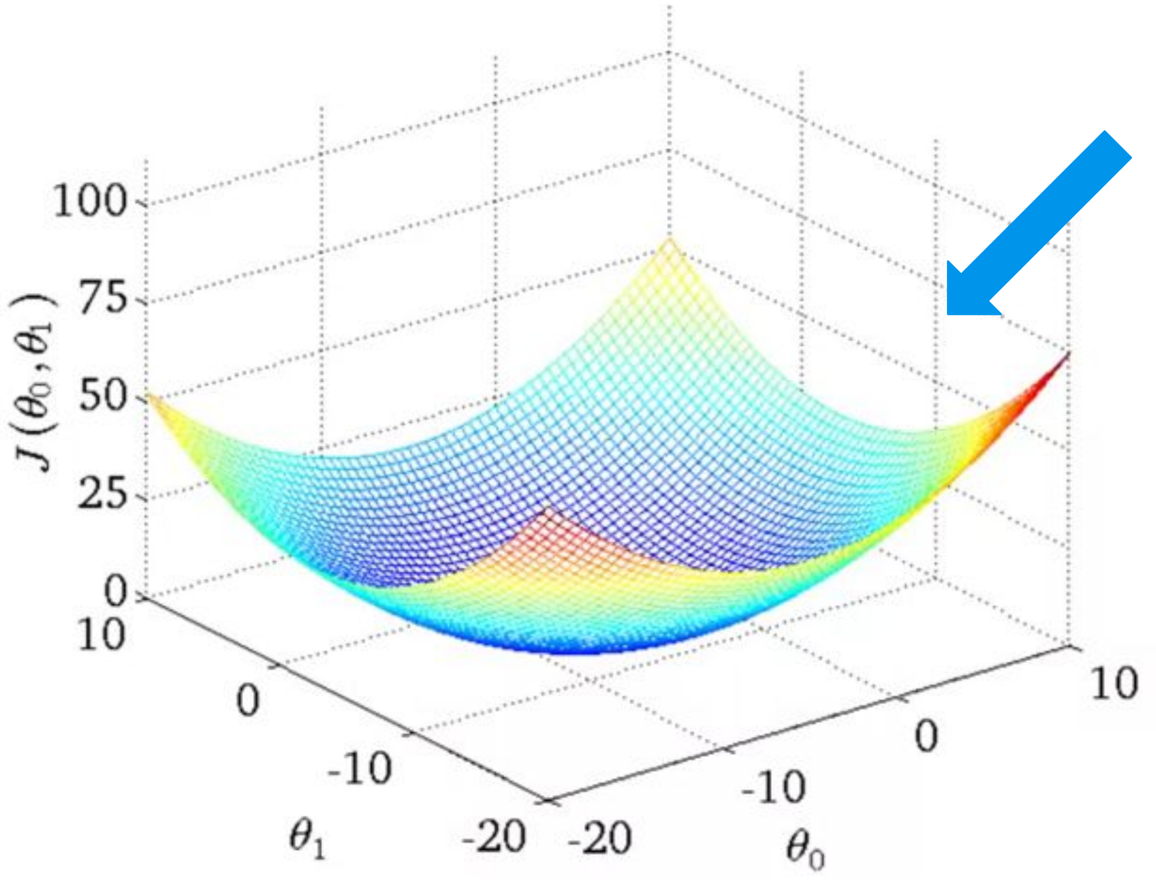
$J(\theta_0,\theta_1)$

(function of the parameters θ_0, θ_1)



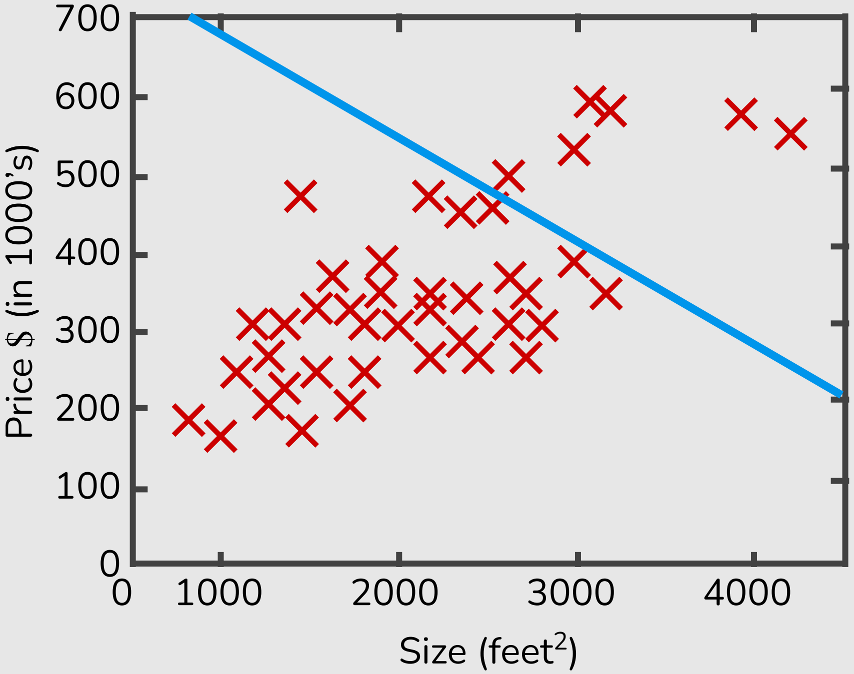
θ_0 and θ_1?

Convex Function



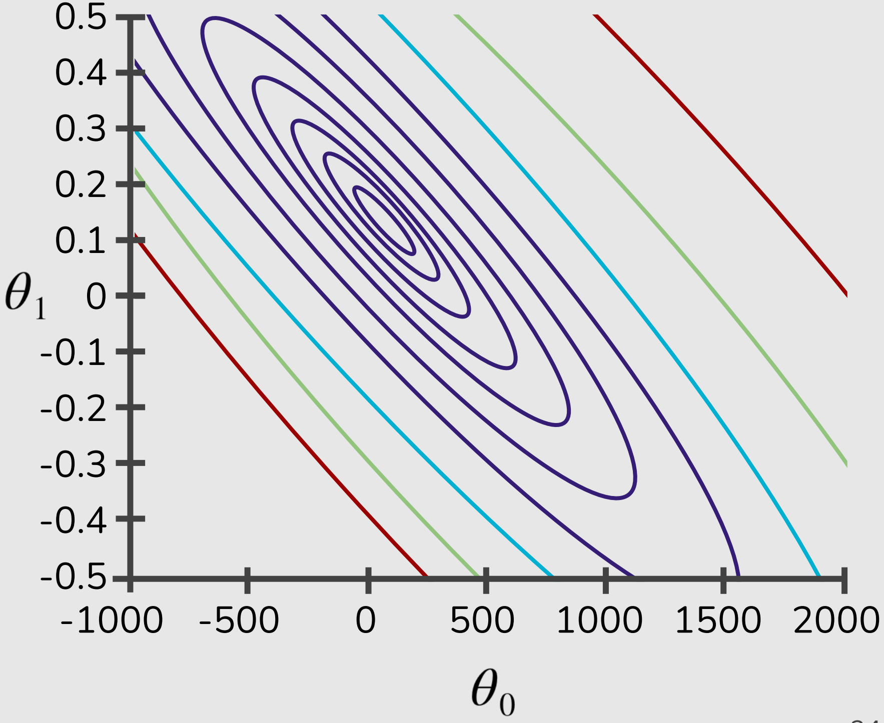
$h_\theta(x)$

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



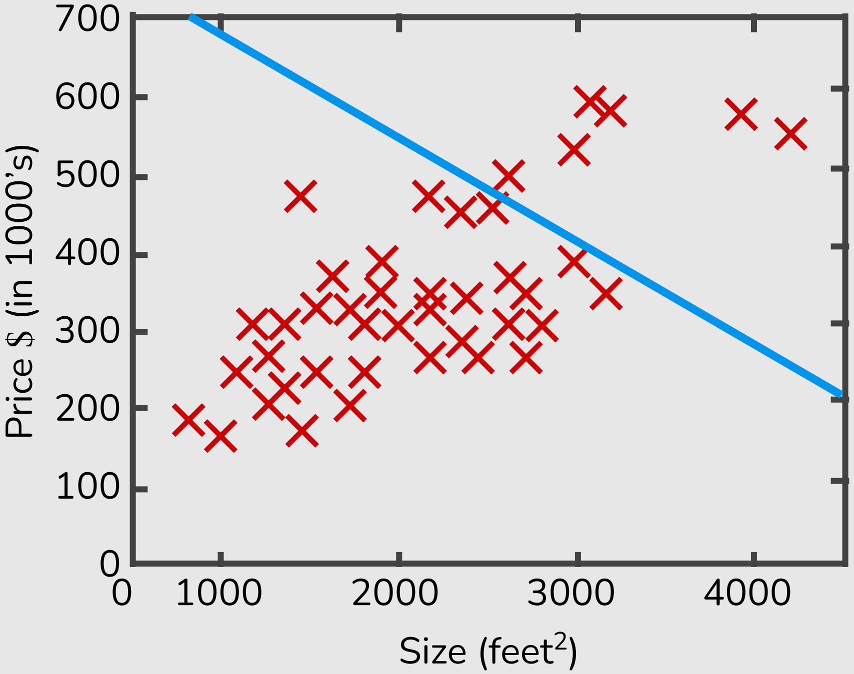
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



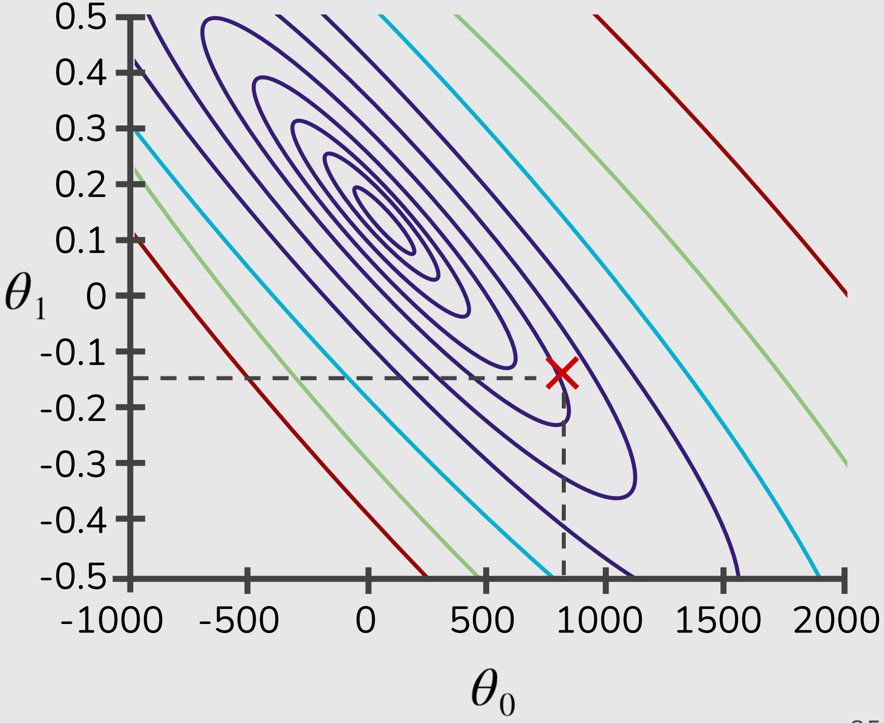
$h_\theta(x)$

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



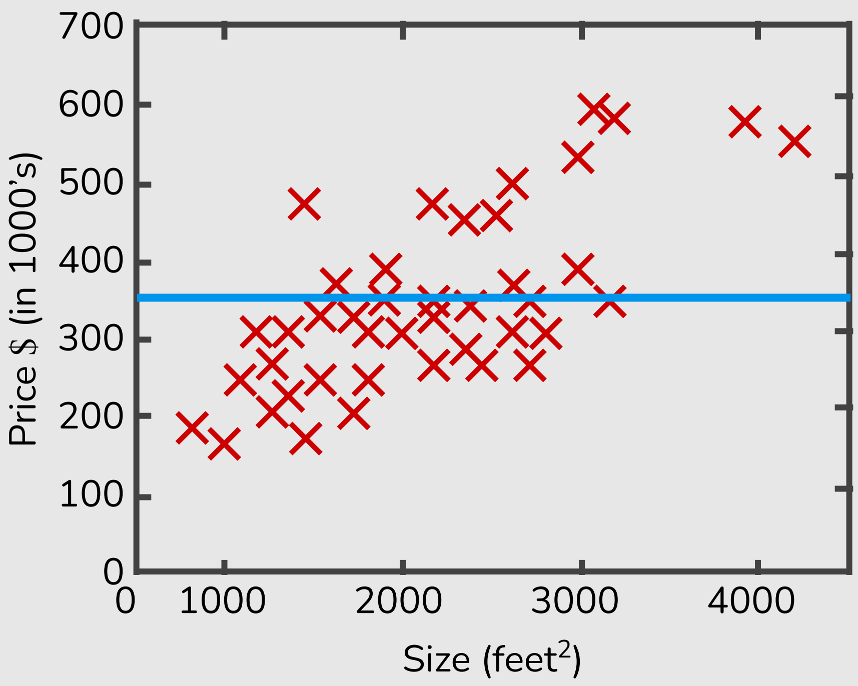
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



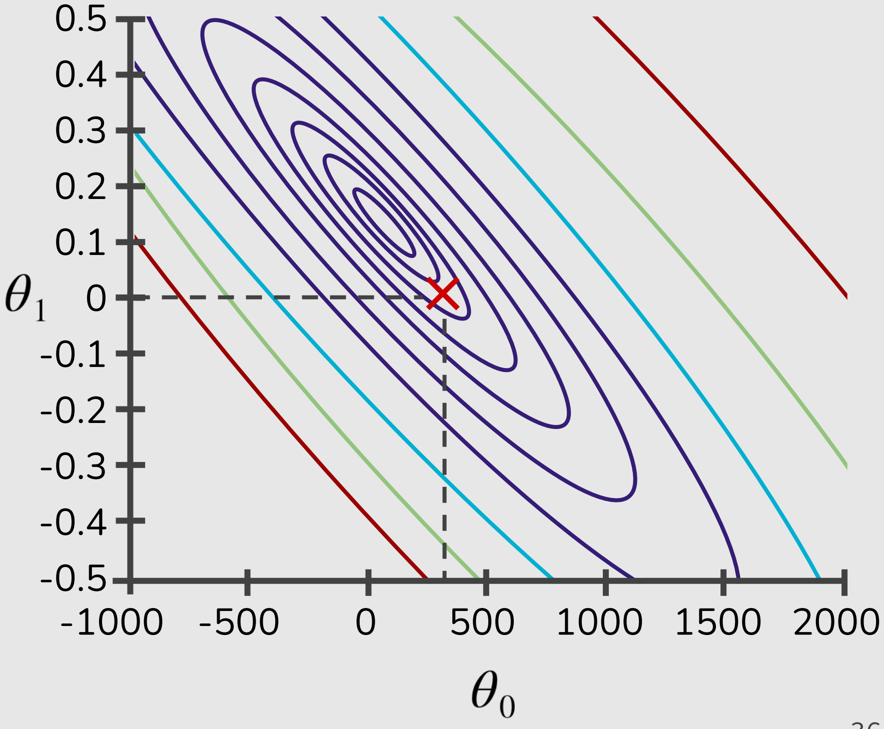
$h_\theta(x)$

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



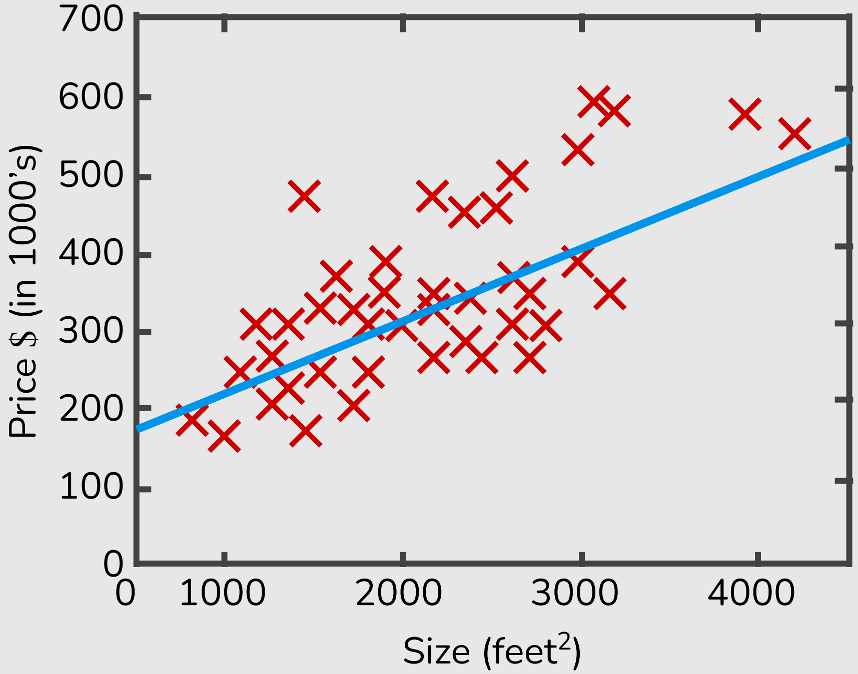
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



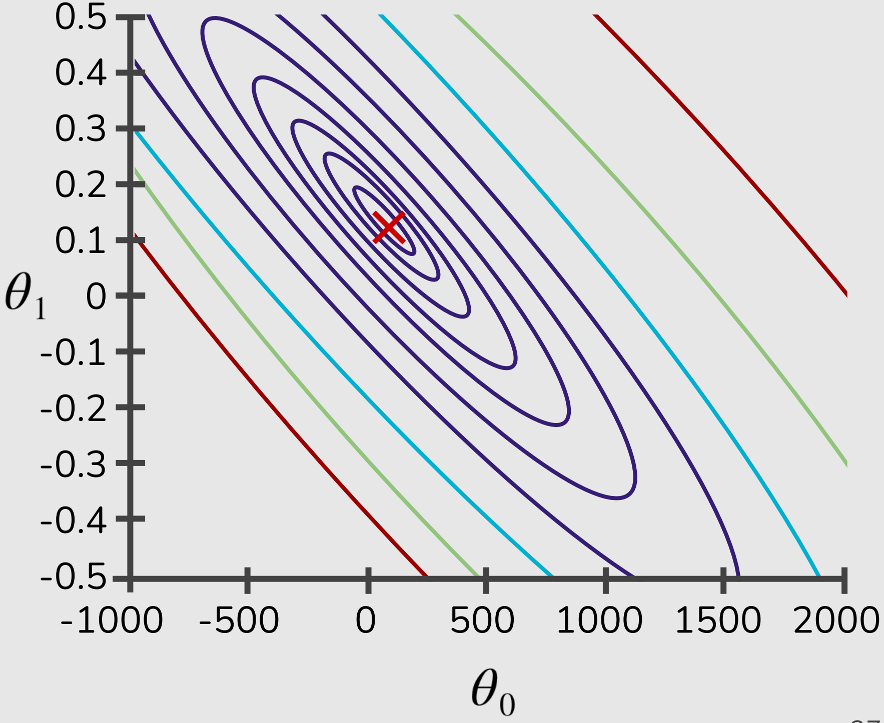
$h_\theta(x)$

(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)



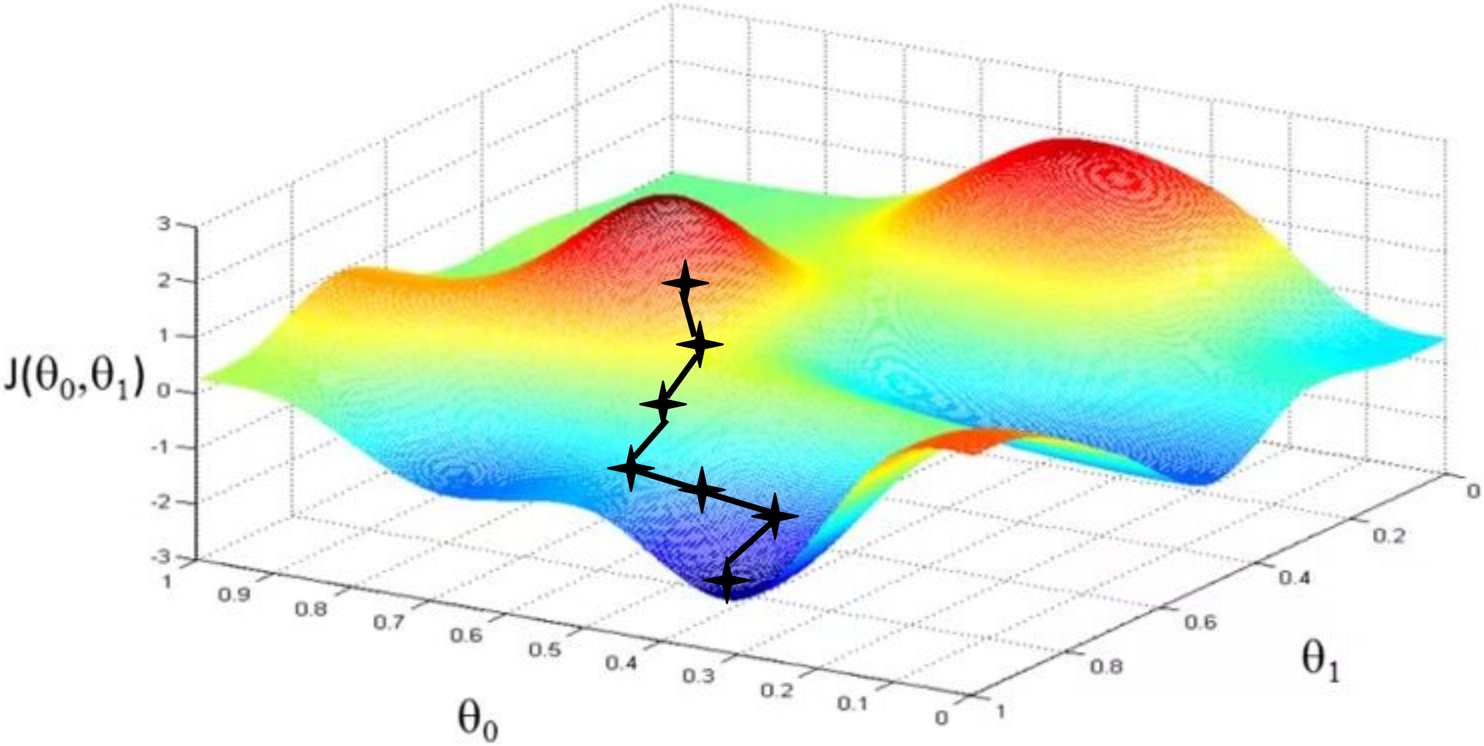
Gradient Descent

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

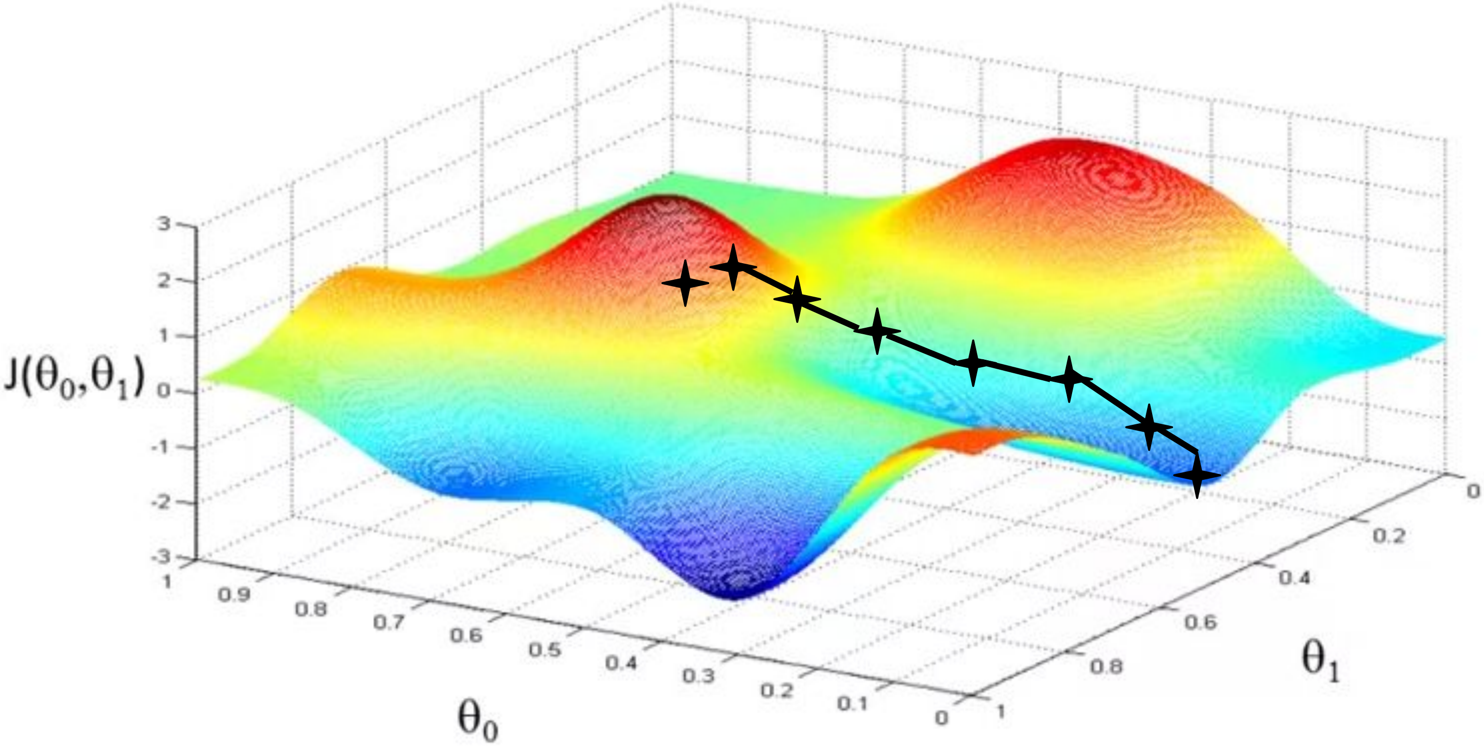
Want $\underset{\theta_0,\theta_1}{\text{minimize}}\, J(\theta_0,\theta_1)$

Outline:

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



$J(\theta_0,\theta_1)$
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 {

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

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

}

Gradient Descent algorithm

repeat until convergence {

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

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

}

Learning rate

Derivative term

Gradient Descent algorithm

repeat until convergence {

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

}

Correct: Simultaneous update

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

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

$$\theta_0 := \text{temp0}$$

$$\theta_1 := \text{temp1}$$

Gradient Descent algorithm

repeat until convergence {

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

}

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

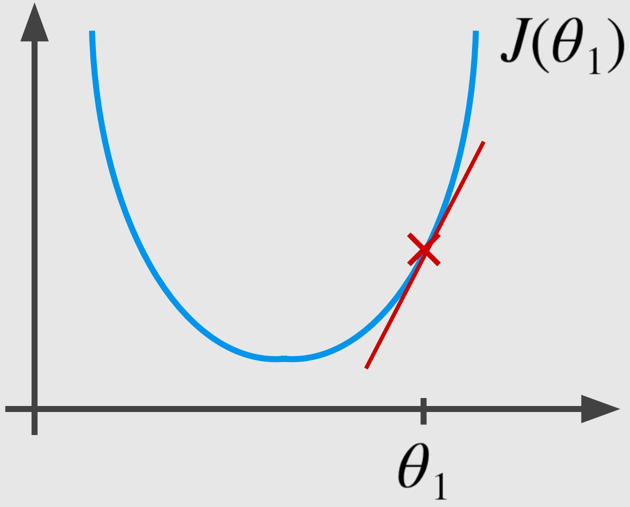
$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp0}$$

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

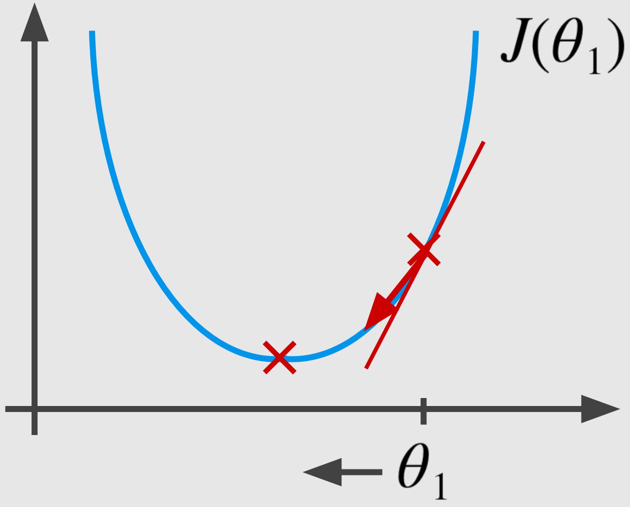
$$\theta_1 := \text{temp1}$$

$$\theta_1 \in \mathbb{R}$$



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

$\theta_1 \in \mathbb{R}$

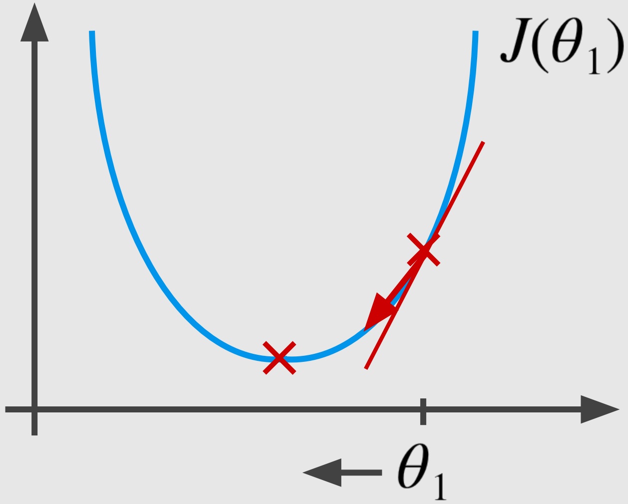


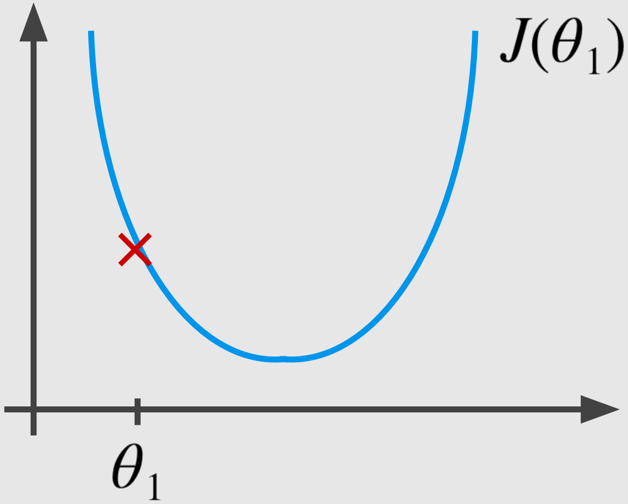
≥ 0

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

$\theta_1 := \theta_1 - \alpha \cdot$ (positive number)

$$\theta_1 \in \mathbb{R}$$

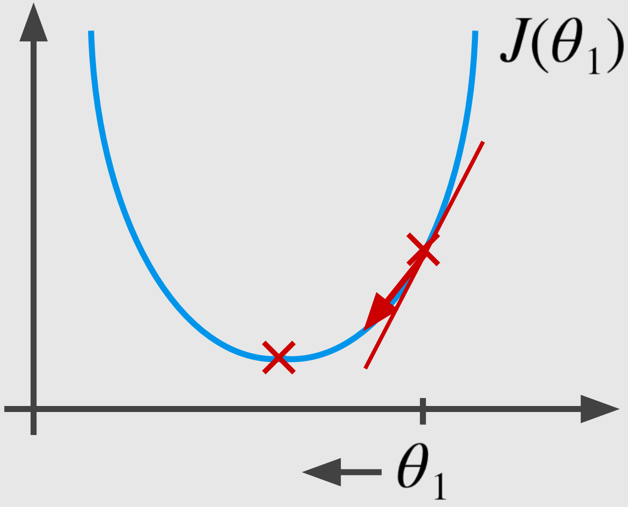


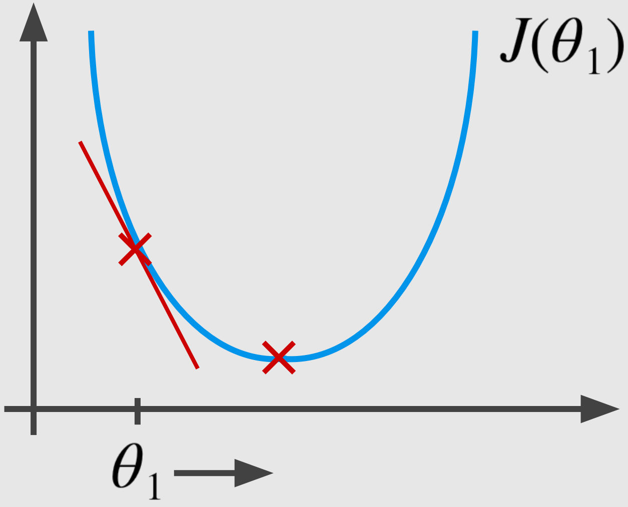


$$\theta_1 := \theta_1 - \alpha \boxed{\frac{d}{d\theta_1} J(\theta_1)} \geq 0$$

$$\theta_1 := \theta_1 - \alpha \cdot \text{(positive number)}$$

$$\theta_1 \in \mathbb{R}$$





≥ 0

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

$$\theta_1 := \theta_1 - \alpha \cdot \text{(positive number)}$$

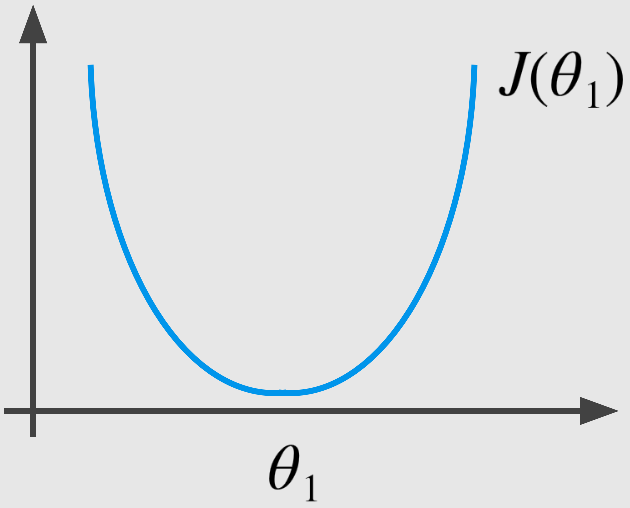
≤ 0

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

$$\theta_1 := \theta_1 - \alpha \cdot \text{(negative number)}$$

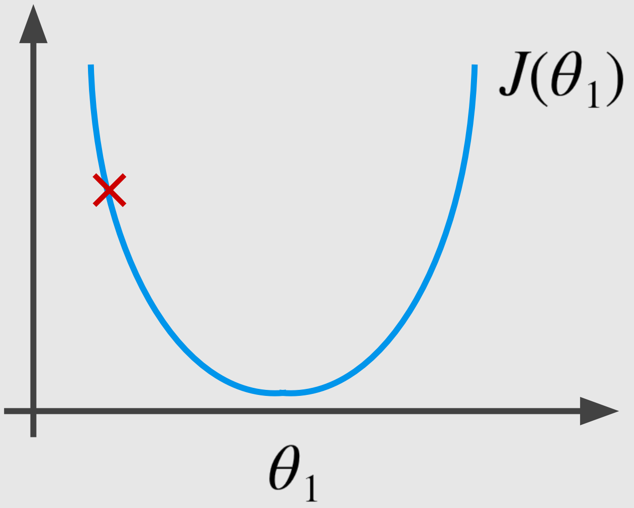
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be ...



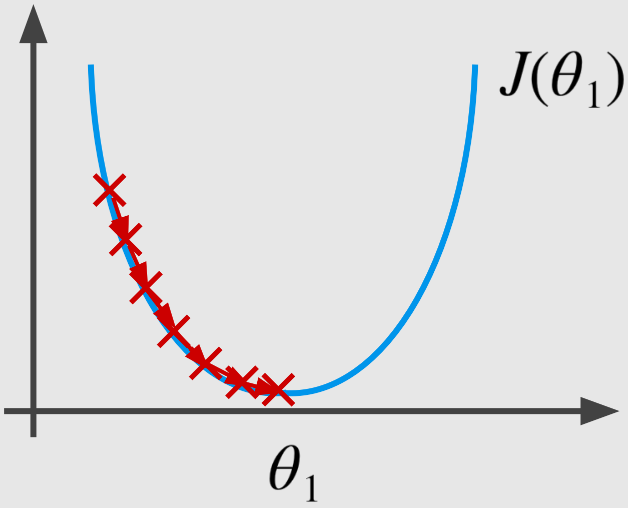
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be ...



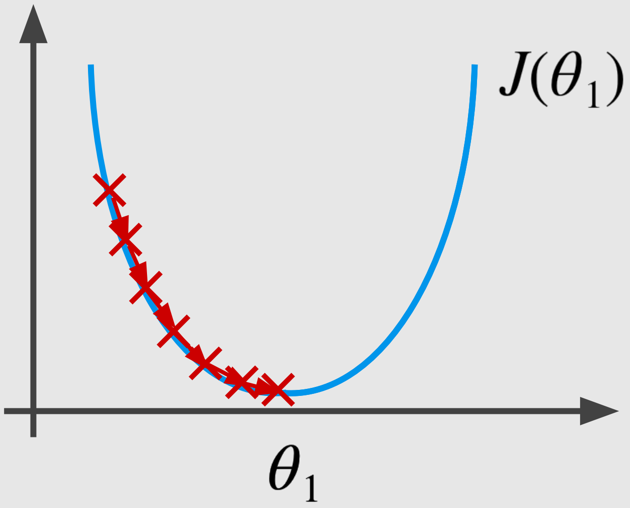
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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

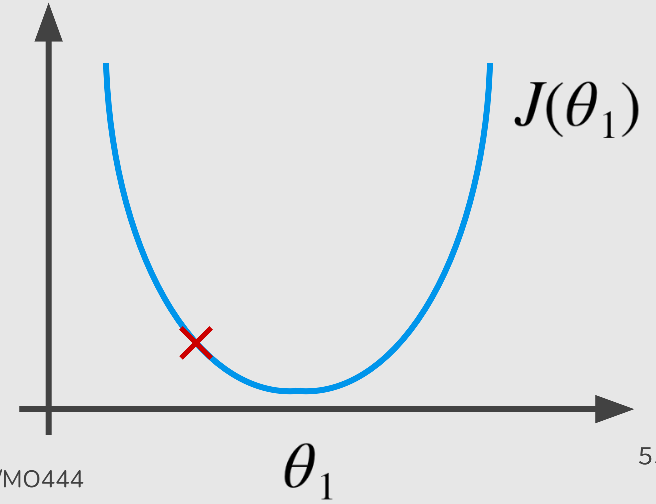


$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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

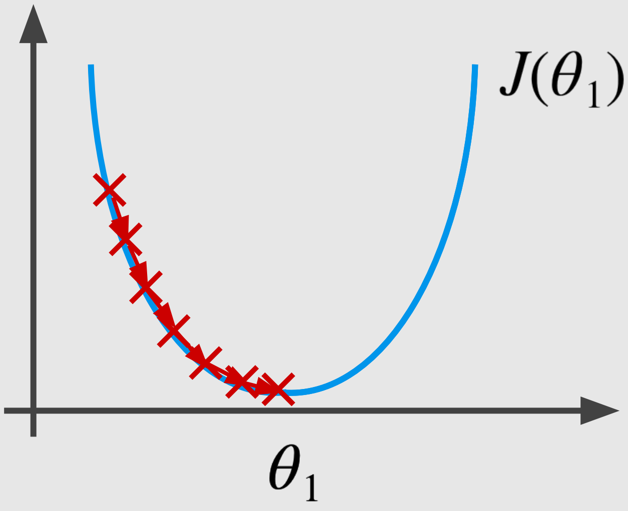


If α is too large, gradient descent can be ...

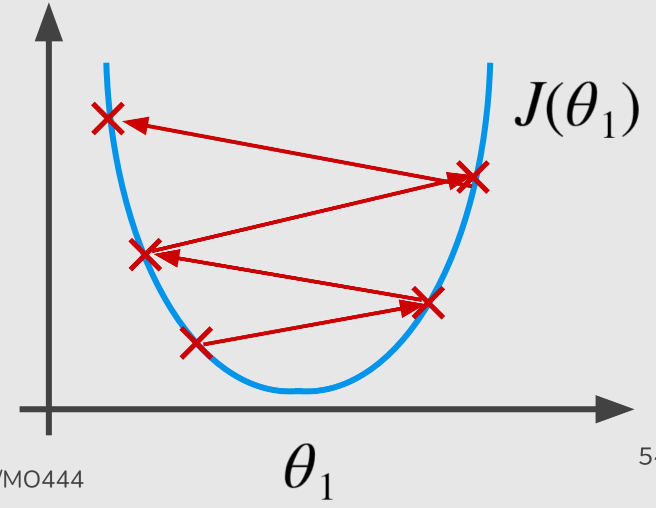


$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

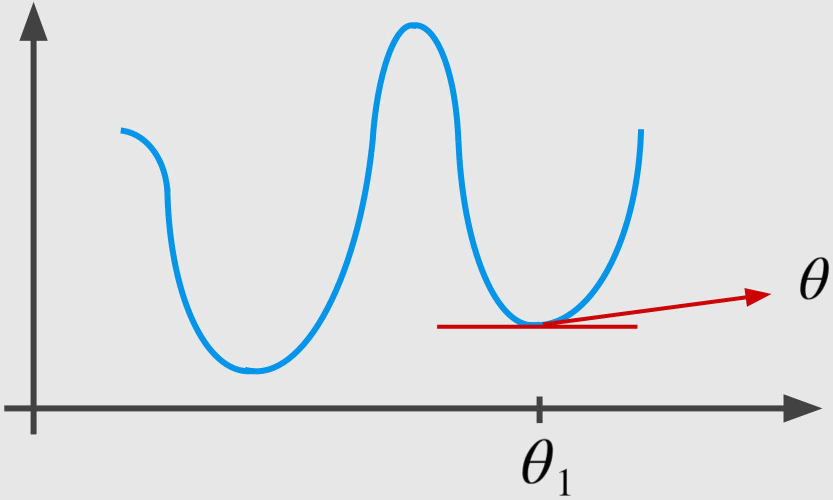
If α is too small, gradient descent can be slow.



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



What will one step of gradient descent $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$ do?



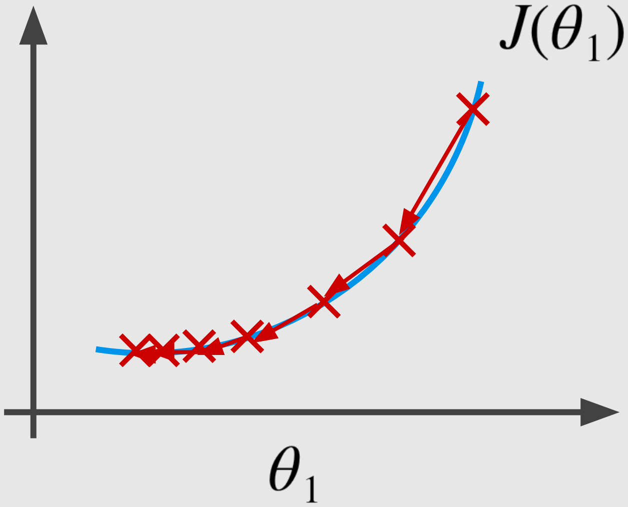
$$\theta_1 := \theta_1 - \alpha \boxed{\frac{d}{d\theta_1} J(\theta_1)}$$

θ_1 at local optima

$$= 0$$

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 algorithm

repeat until convergence {

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

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

}

Linear Regression Model

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

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

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

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

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

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

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

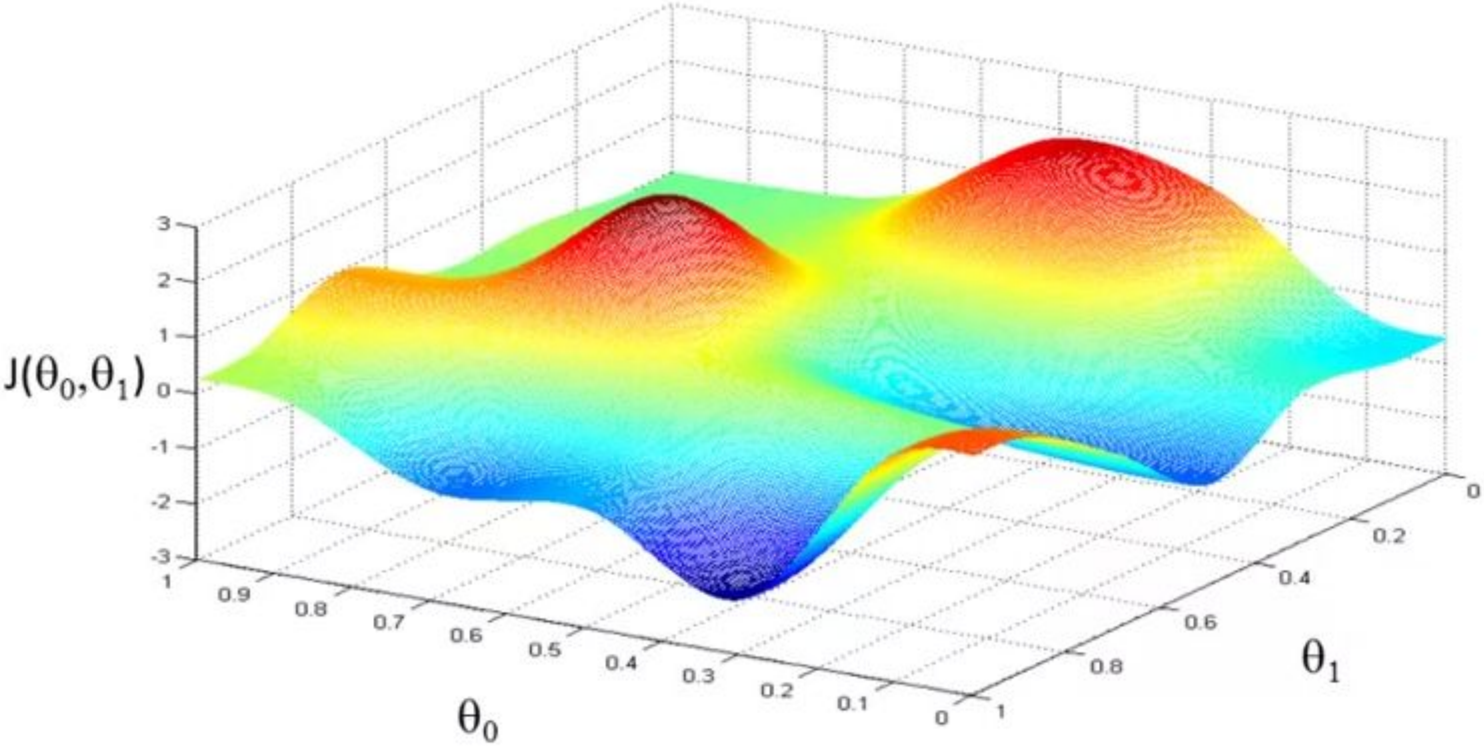
Gradient Descent algorithm

repeat until convergence {

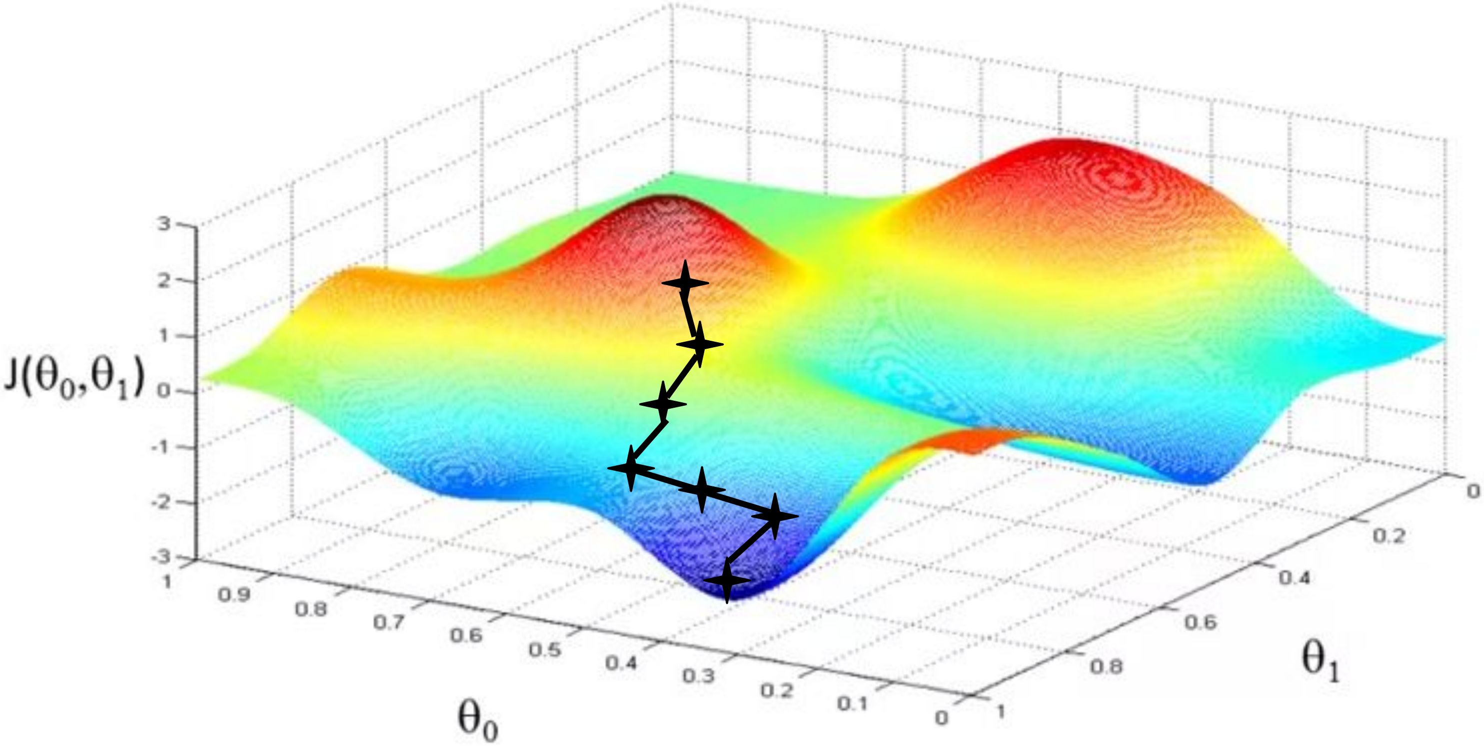
$$\left.\begin{aligned}\theta_0 &:= \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \\ \theta_1 &:= \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\,x^{(i)}\end{aligned}\right\}$$

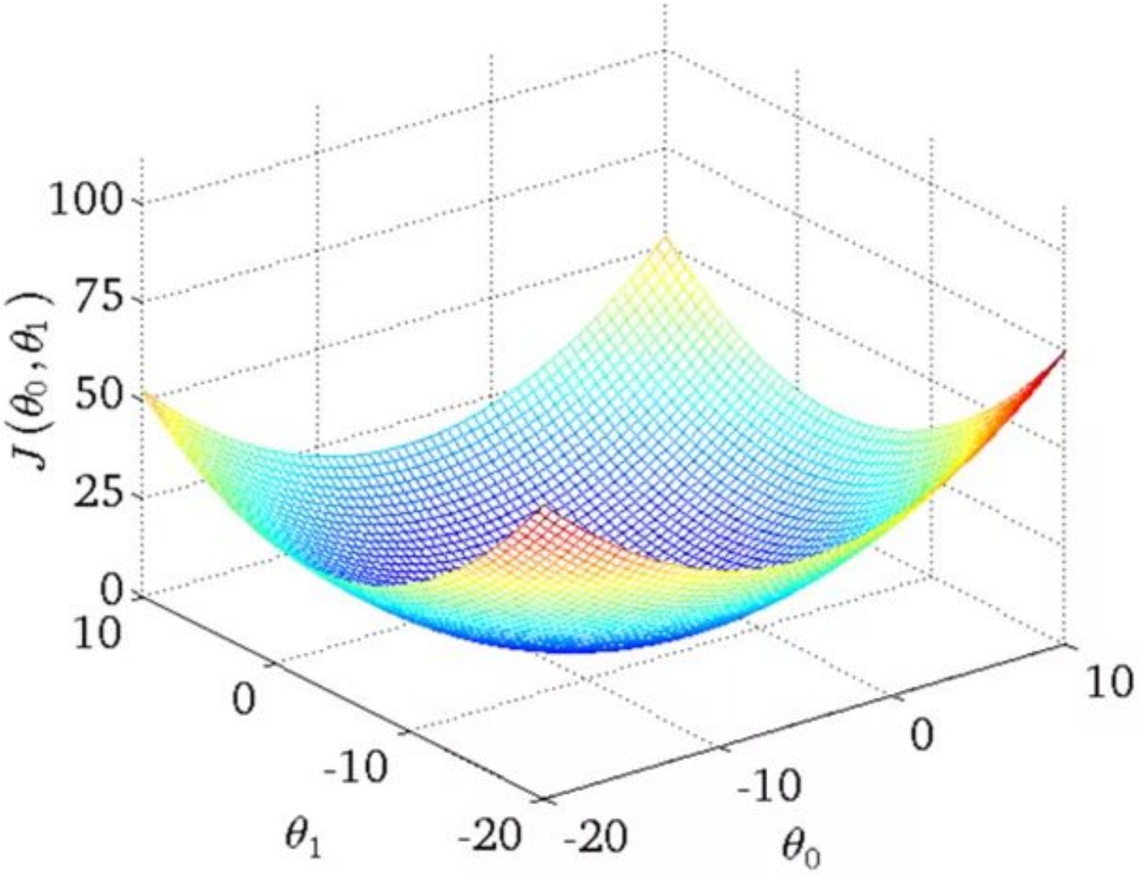
update θ_0 and θ_1 simultaneously

}



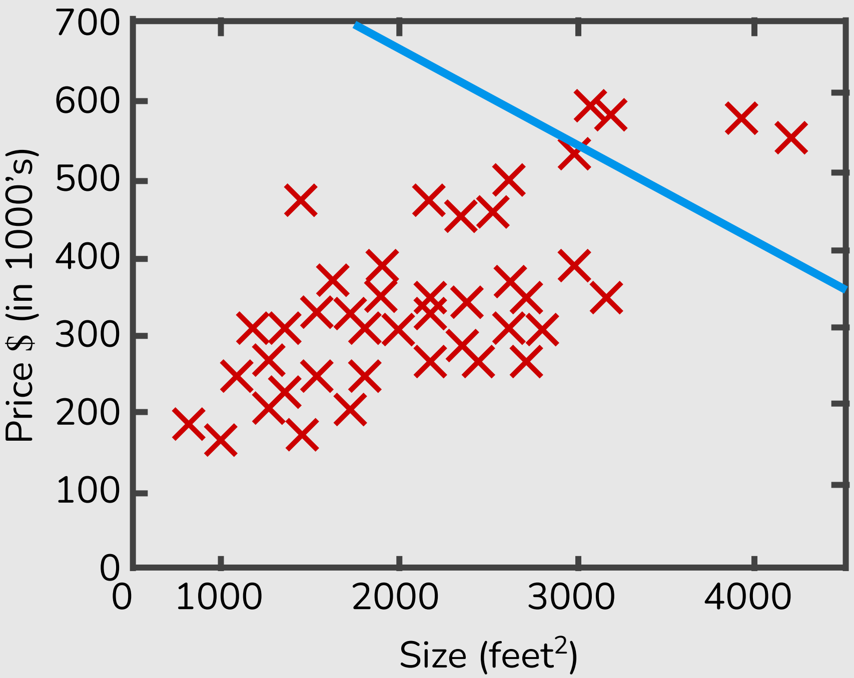
$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


3
2
1
0
-1
-2
-3
J(θ0,θ1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1


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

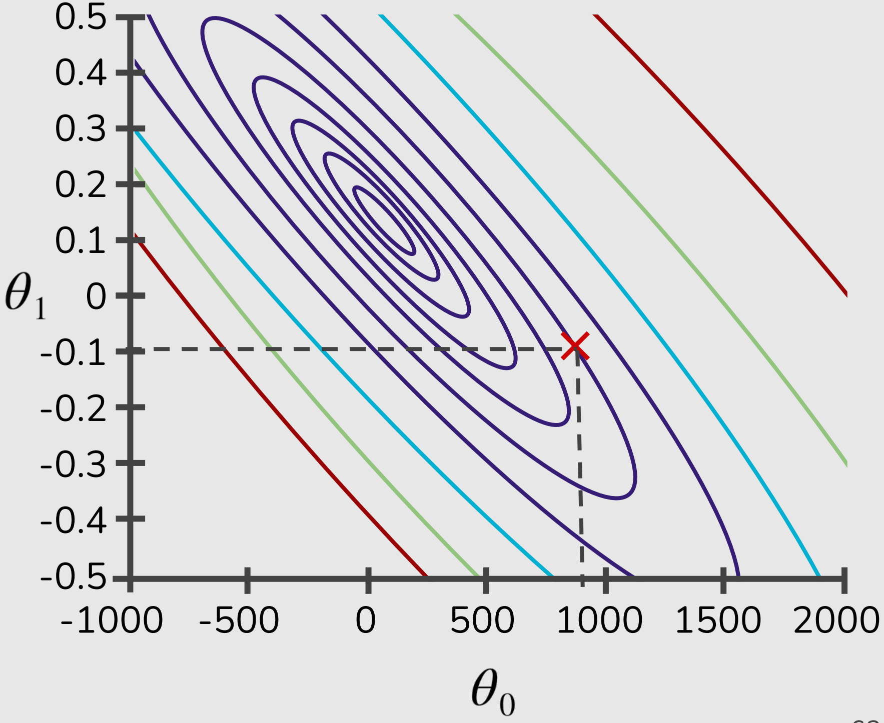
$h_\theta(x)$

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



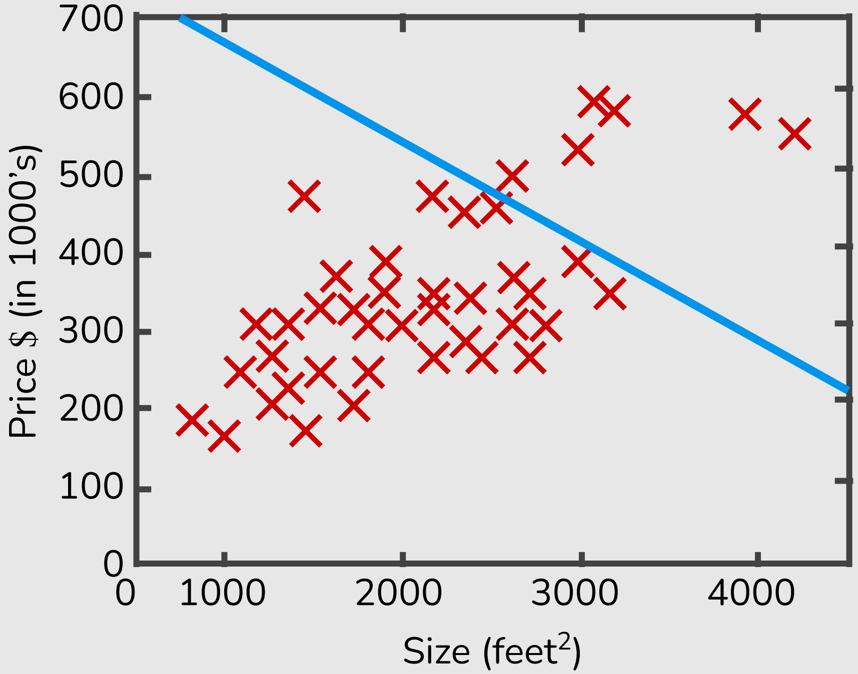
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



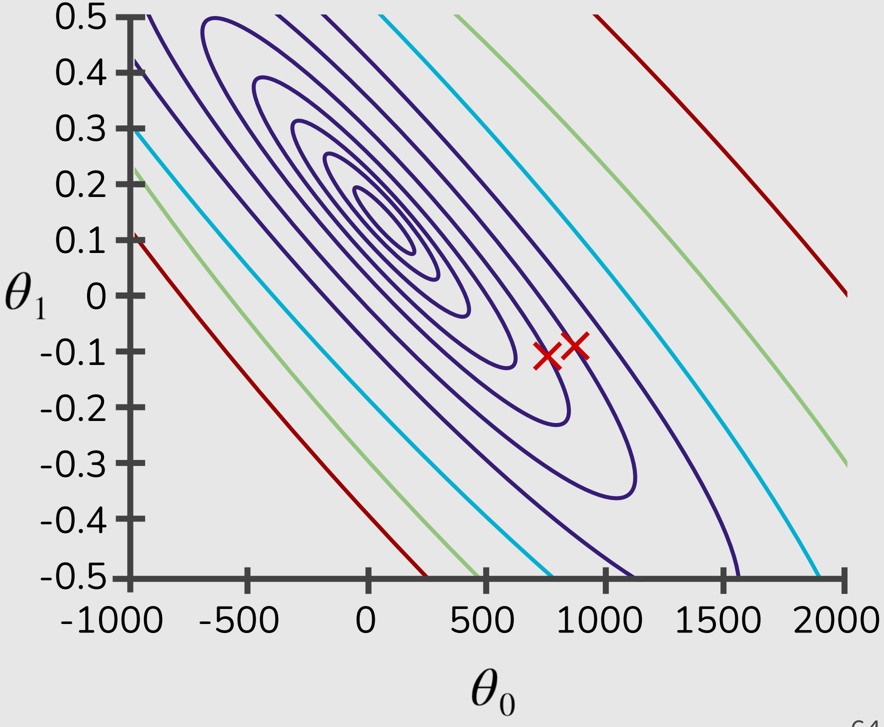
$$h_\theta(x)$$

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



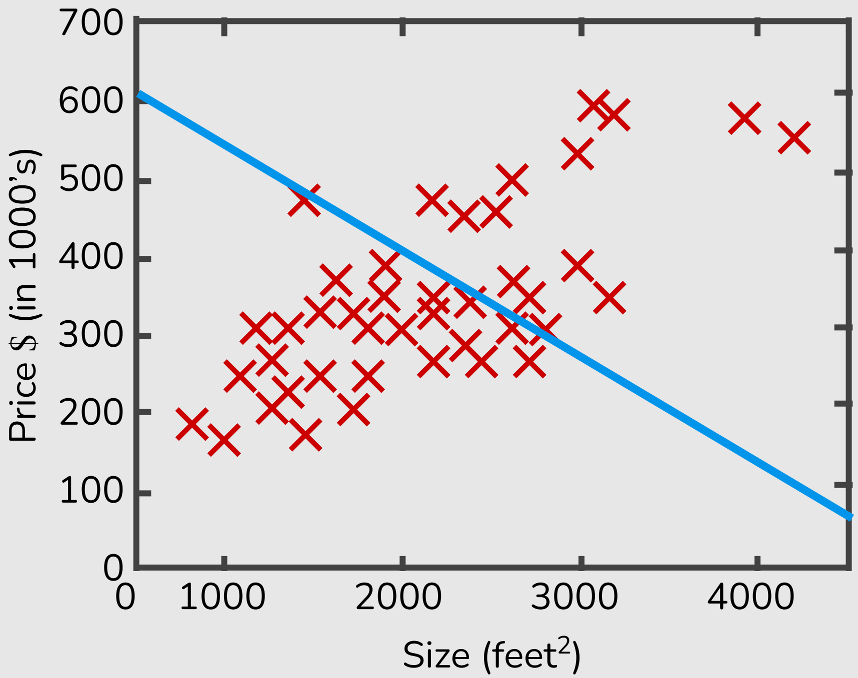
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



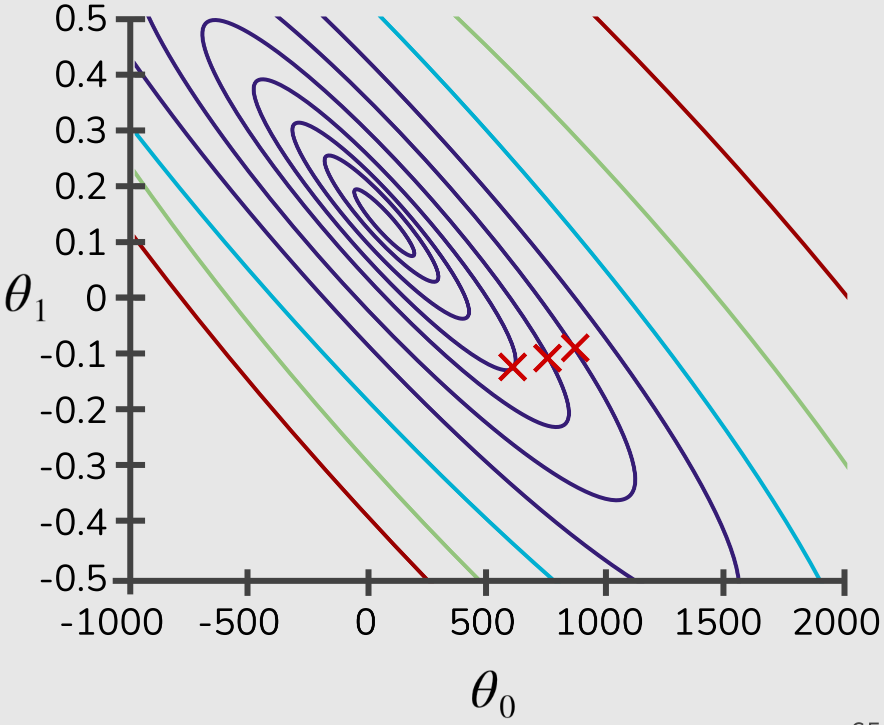
$h_\theta(x)$

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



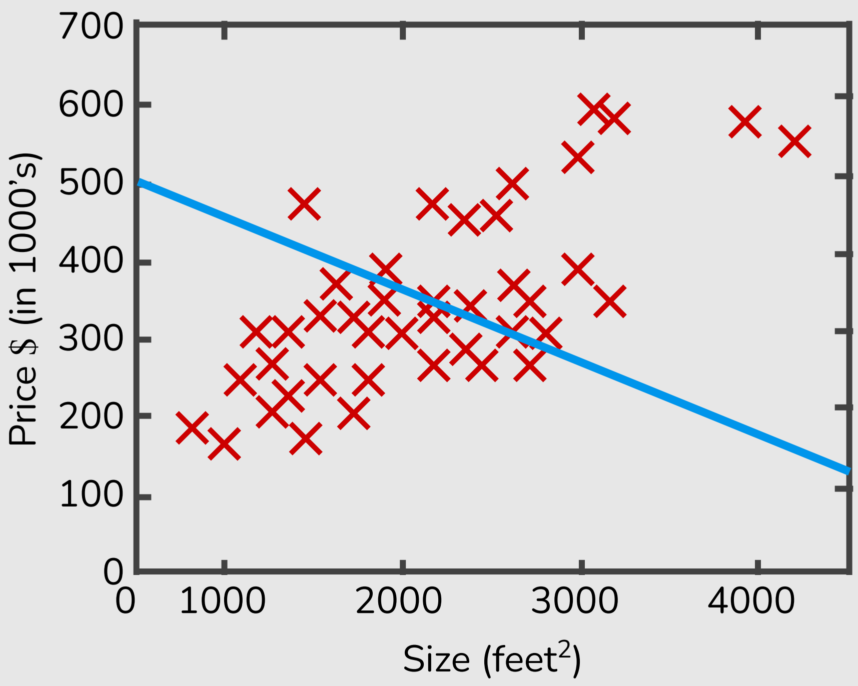
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



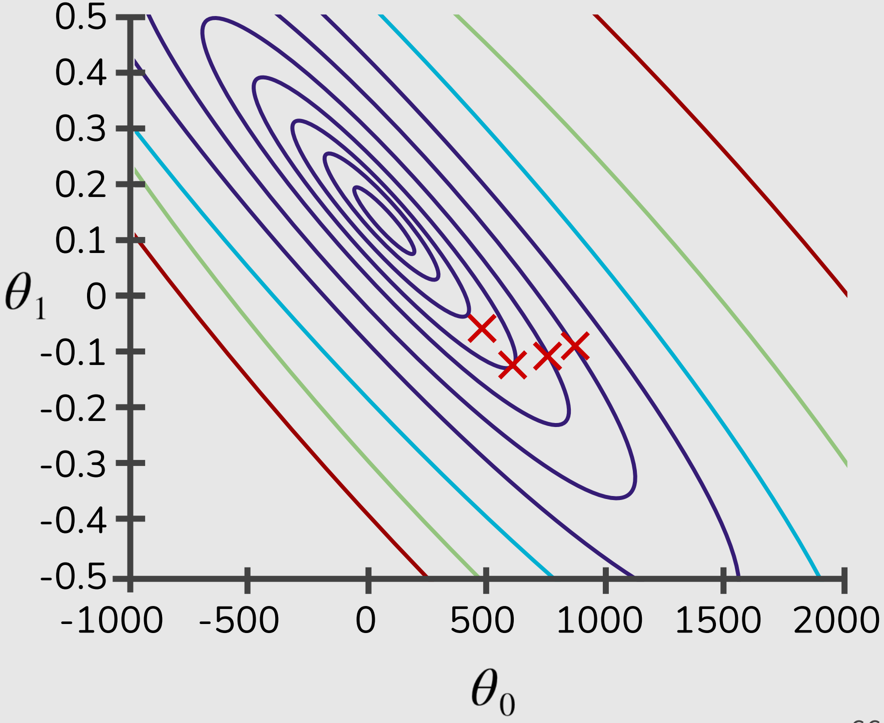
$h_\theta(x)$

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



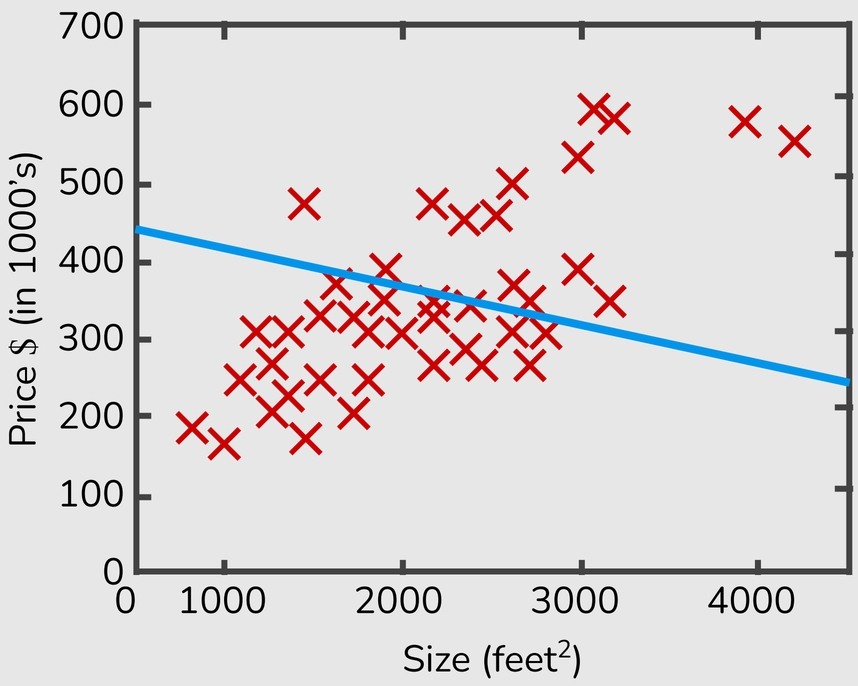
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



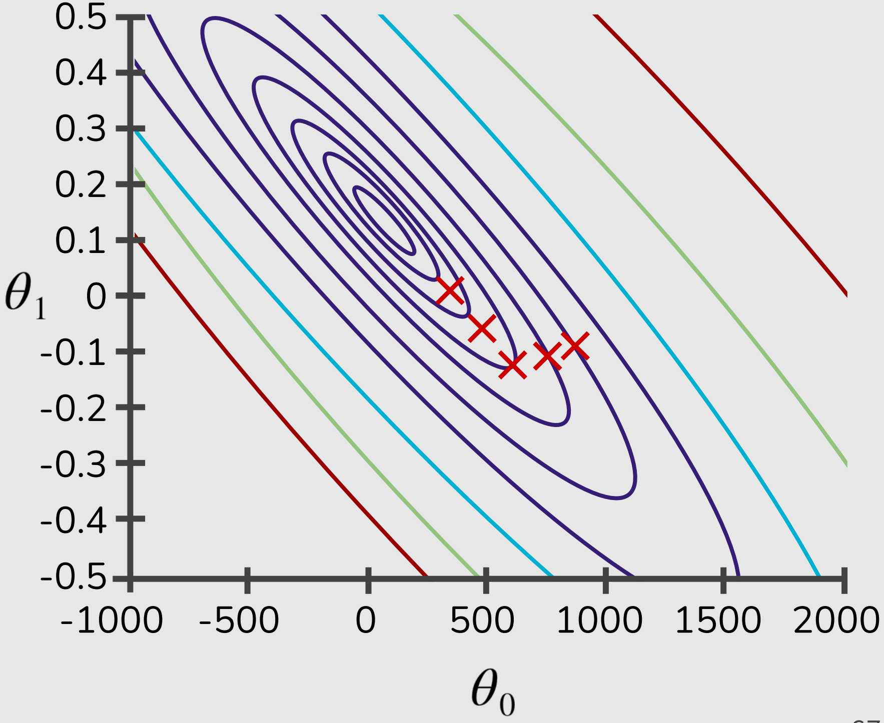
$h_\theta(x)$

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



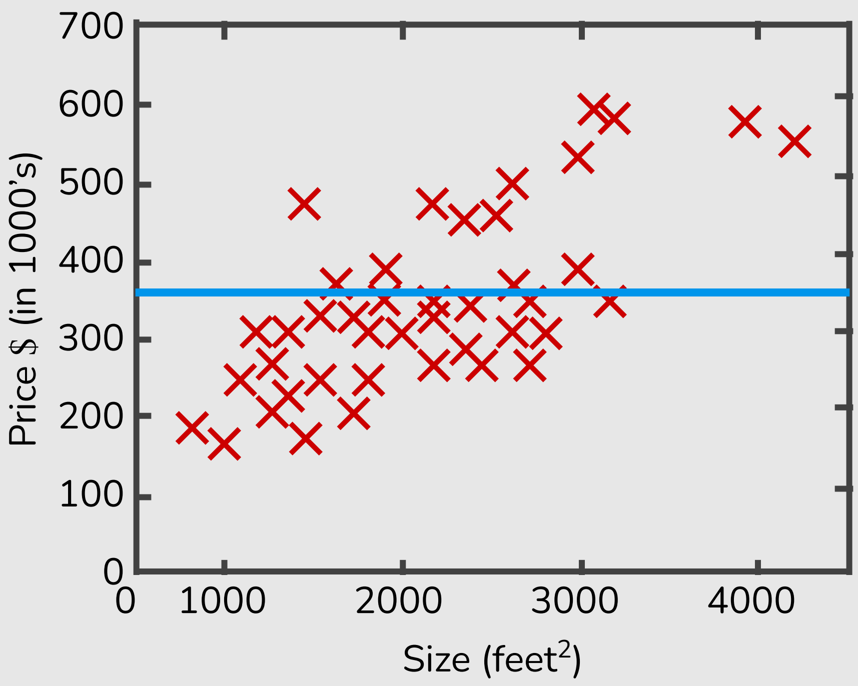
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



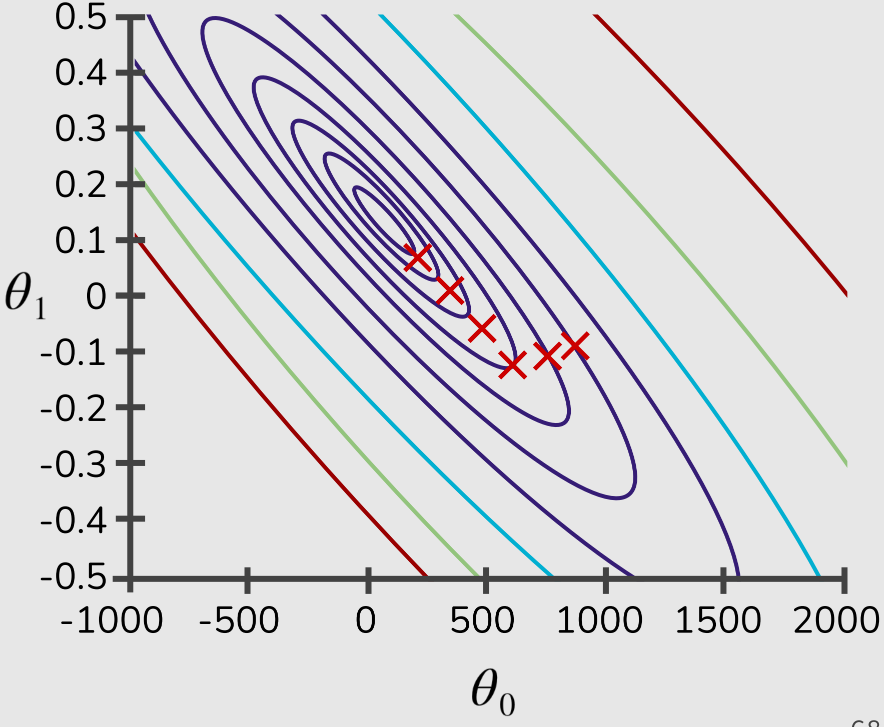
$h_\theta(x)$

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



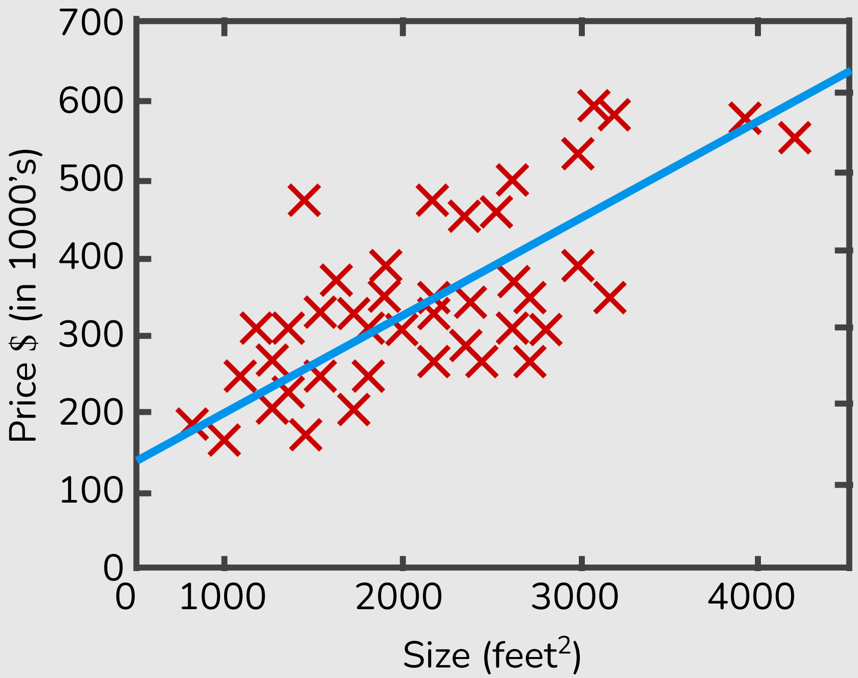
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



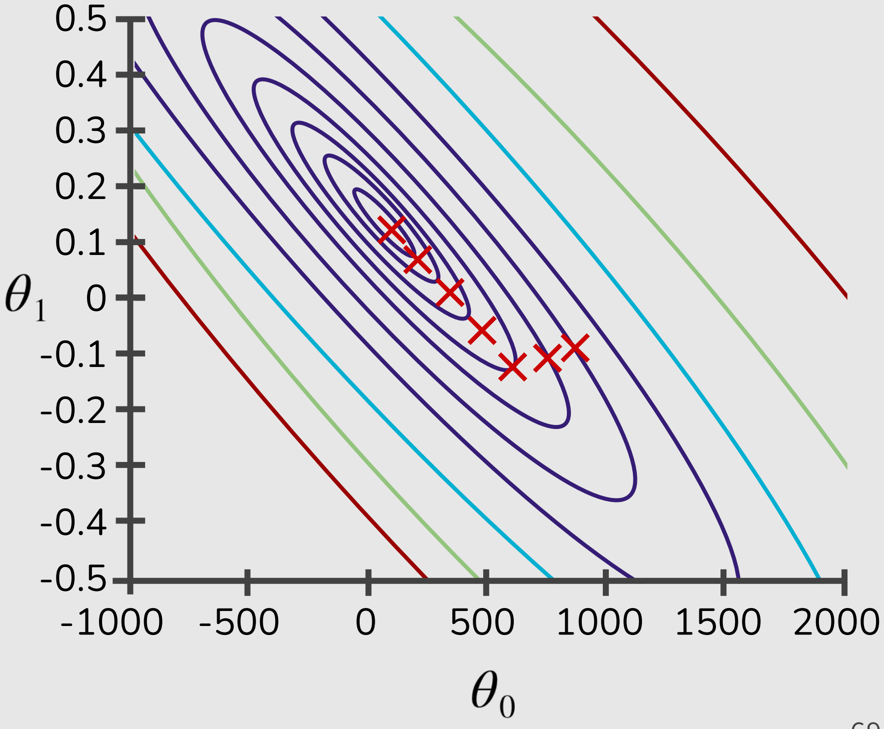
$h_\theta(x)$

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

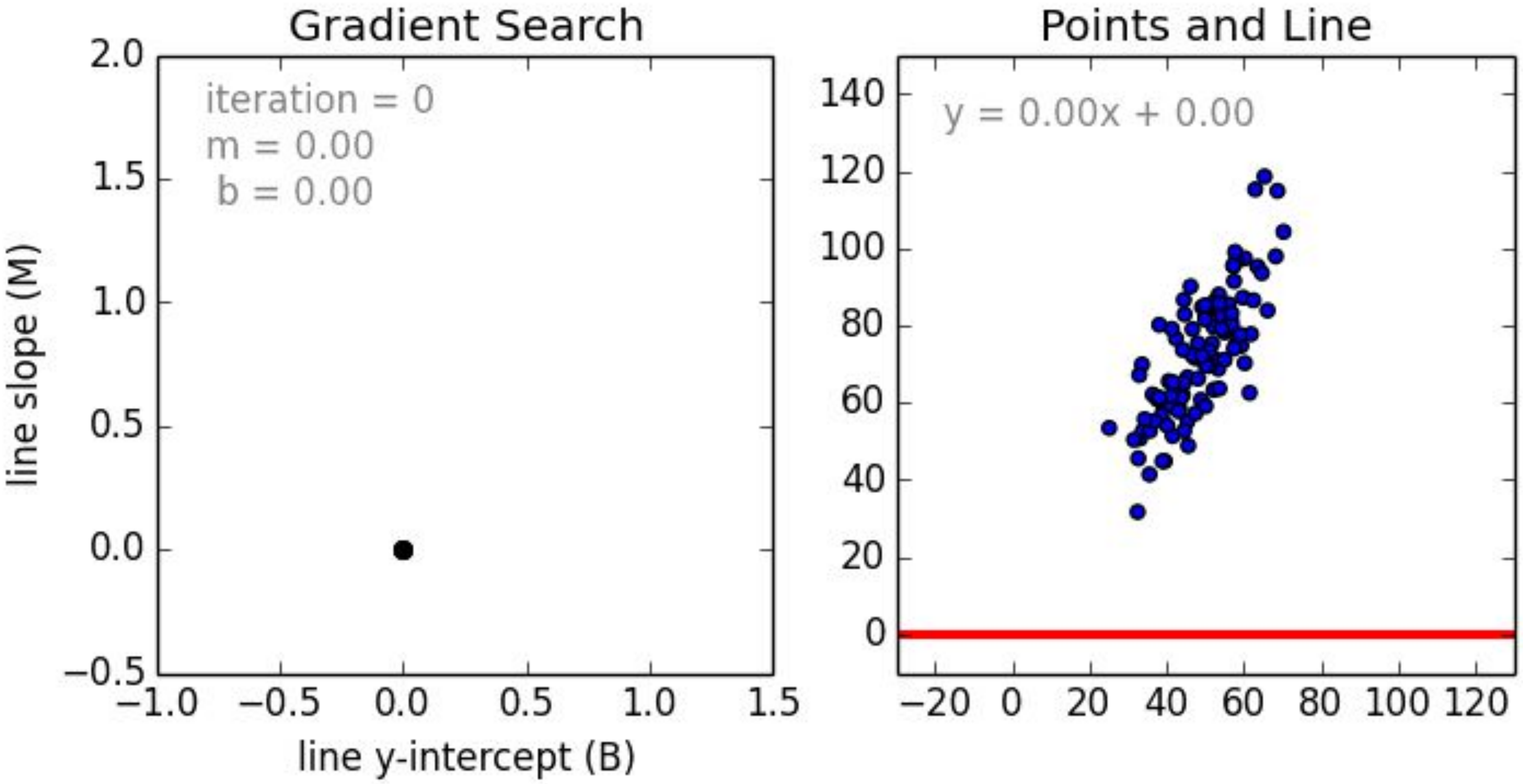


$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

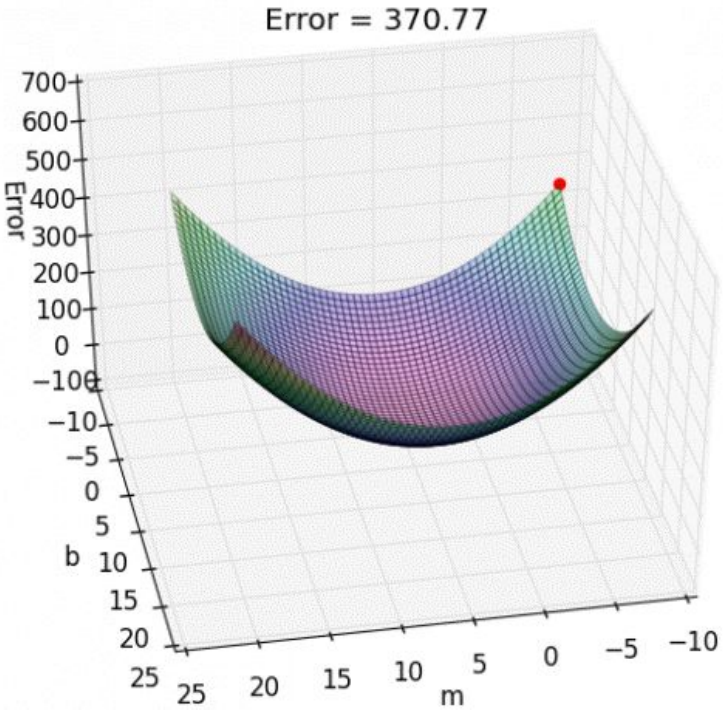


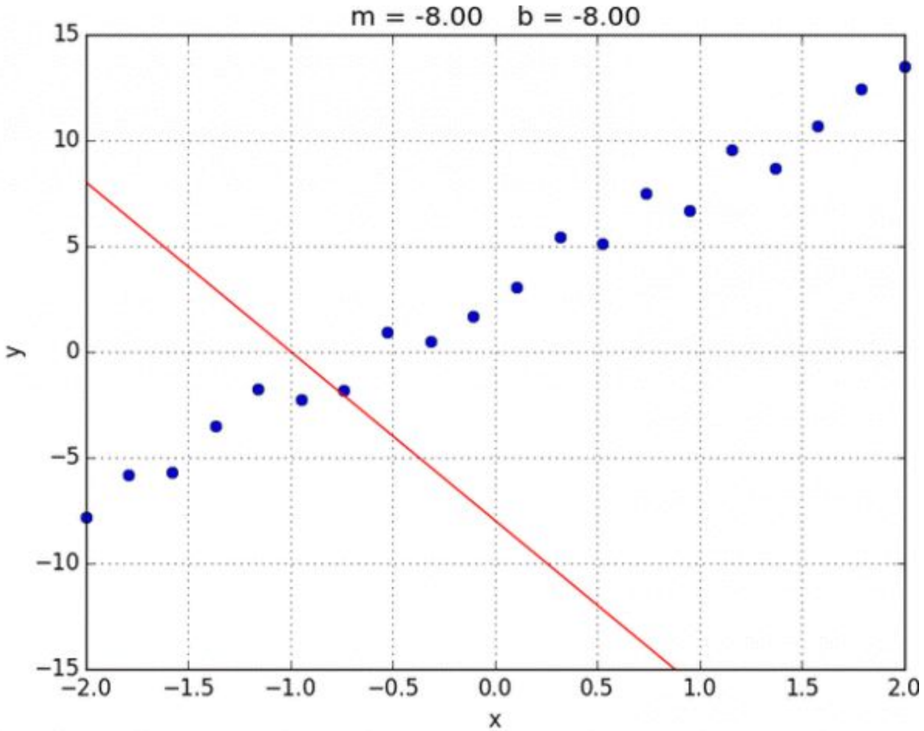
$$h_\theta(x) = \theta_0 + \theta_1 x \quad \Rightarrow \quad y = b + mx$$



Credit: https://github.com/mattnedrich/GradientDescentExample/raw/master/gradient_descent_example.gif

$$y = b + mx$$





Credit: https://alykhantejani.github.io/a-brief-introduction-to-gradient-descent/

“Batch” Gradient Descent

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

- Stochastic Gradient Descent
- Mini-batch Gradient Descent

"Batch" Gradient Descent

repeat until convergence {

$$\left.\begin{aligned}\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x^{(i)}\end{aligned}\right\} \text{update } \theta_0 \text{ and } \theta_1 \text{ simultaneously}$$

}

Stochastic Gradient Descent

Each step of gradient descent uses **one training example**.

repeat until convergence {

for $i \coloneqq 1, \ldots, m$ {

$$\theta_0 := \theta_0 - \alpha(h_\theta(x^{(i)}) - y^{(i)})$$

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

}

}

Mini-batch Gradient Descent

Each step of gradient descent uses ***b* training examples**.

Say $b = 10, m = 1000.$
repeat until convergence {

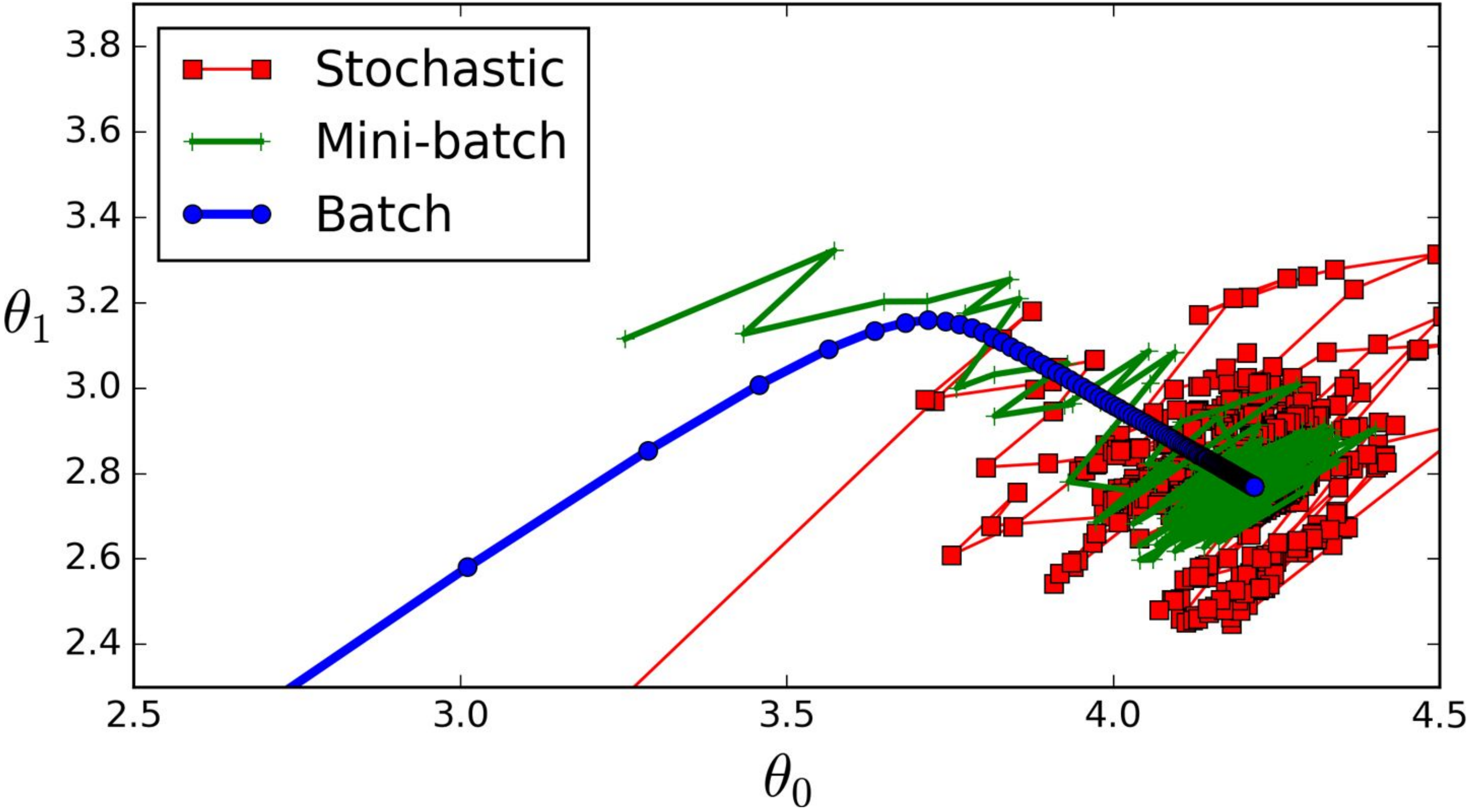
for $i \coloneqq 1, 11, 21\ldots, 991$ {

$$\theta_0 := \theta_0 - \alpha \frac{1}{10} \sum_{i=k}^{i+9} (h_\theta(x^{(k)}) - y^{(k)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{10} \sum_{i=k}^{i+9} (h_\theta(x^{(k)}) - y^{(k)})\, x^{(k)}$$

} }

Batch vs. Stochastic vs. Mini-batch



References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 2 & 4
- Pattern Recognition and Machine Learning, Chap. 3
- Probabilistic Machine Learning: An Introduction, Chap. 11

References

Machine Learning Courses

- https://www.coursera.org/learn/machine-learning, Week 1 & 2
- https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html

Machine Learning Videos

- Linear Regression: A friendly introduction, https://youtu.be/wYPUhge9w5c
- Linear Regression, Clearly Explained!!!, https://youtu.be/nk2CQITm_eo

Today's Agenda

- Linear Regression with One Variable
  - Model Representation
  - Cost Function
  - Gradient Descent
- Linear Regression with Multiple Variables
  - Gradient Descent for Multiple Variables
  - Feature Scaling
  - Learning Rate
  - Features and Polynomial Regression
  - Normal Equation