

# L9: Linear Regression

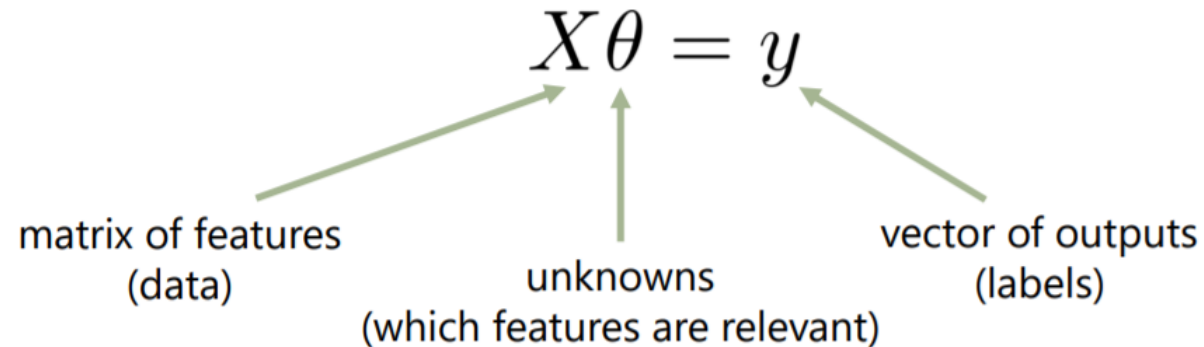
Prof. Xun Jiao

# Before class

- HW1 Answers
- HW2 due now
- HW3 will be posted

# What is a linear equation?

**Linear regression** assumes a predictor of the form



The diagram shows the equation  $X\theta = y$  centered at the top. Three green arrows point from descriptive text below to the components of the equation: one from 'matrix of features (data)' to  $X$ , one from 'unknowns (which features are relevant)' to  $\theta$ , and one from 'vector of outputs (labels)' to  $y$ .

$$X\theta = y$$

matrix of features  
(data)

unknowns  
(which features are relevant)

vector of outputs  
(labels)

(or  $Ax = b$  if you prefer)

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

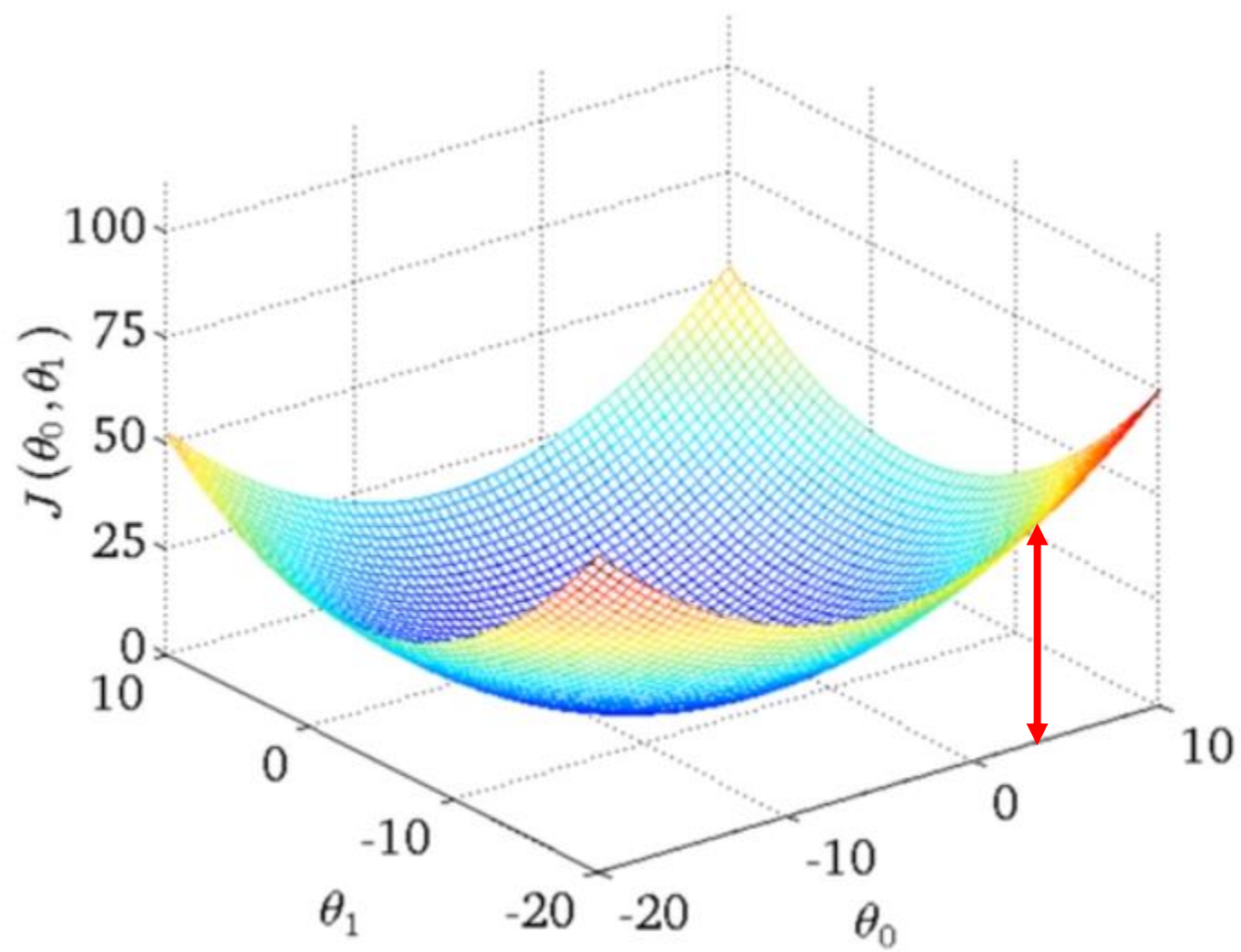
Parameters:  $\theta_0, \theta_1$

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

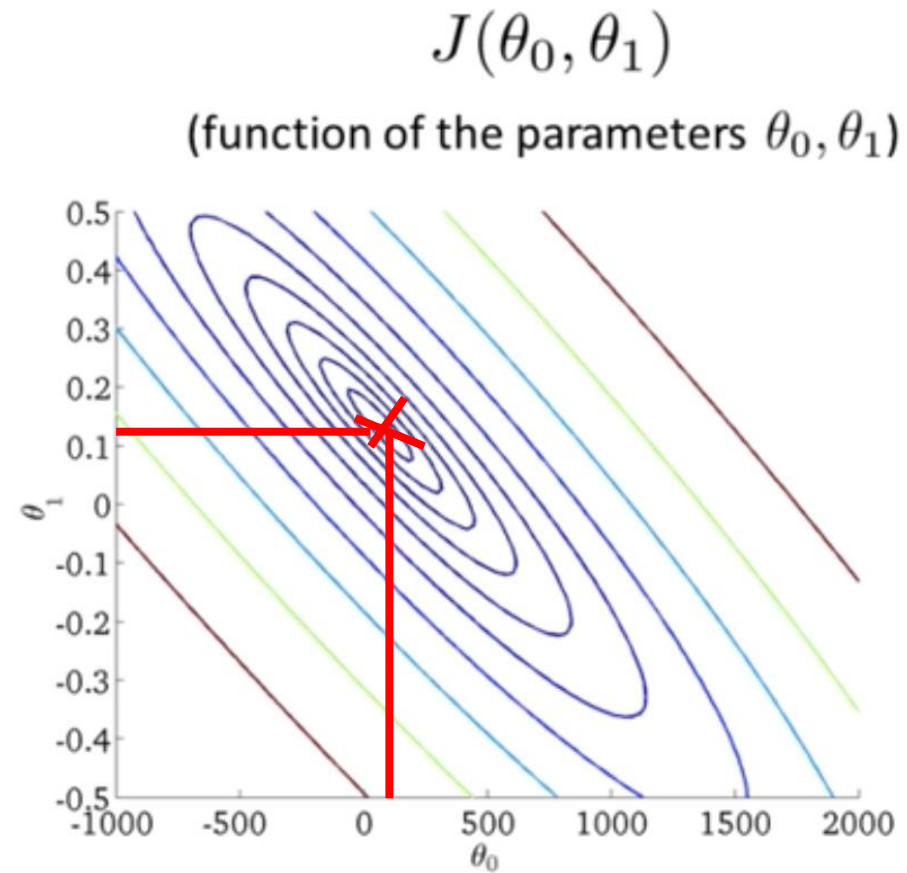
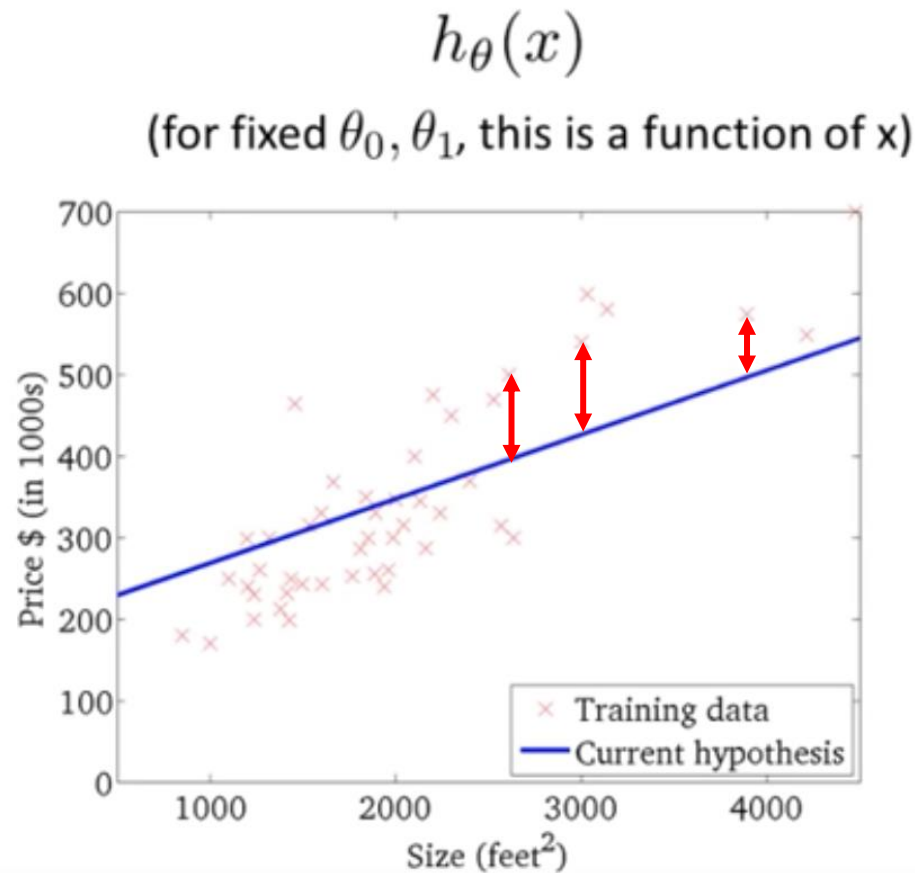
Goal:  $\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$



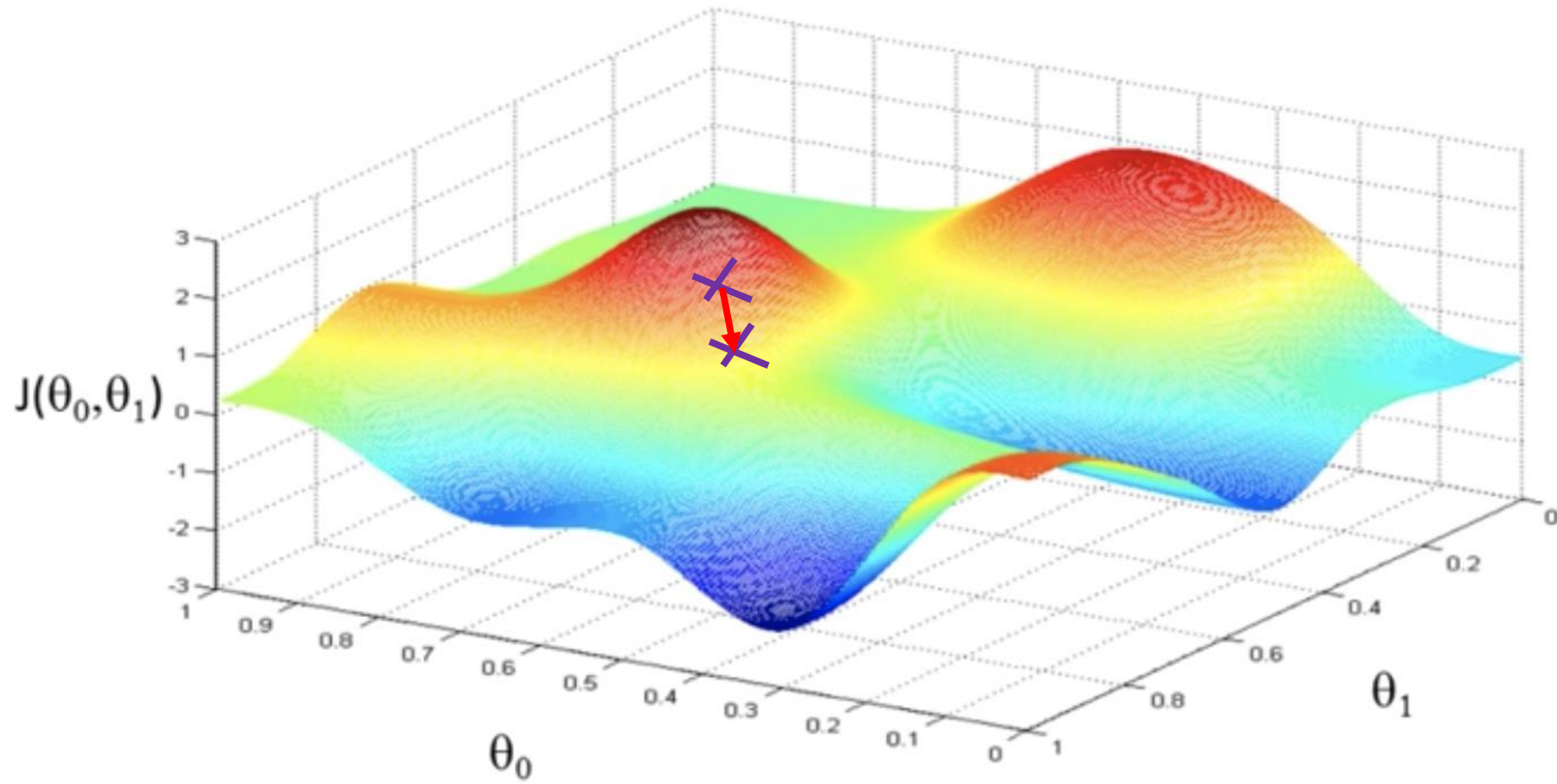
Squared error function:  
most-widely used cost  
function.

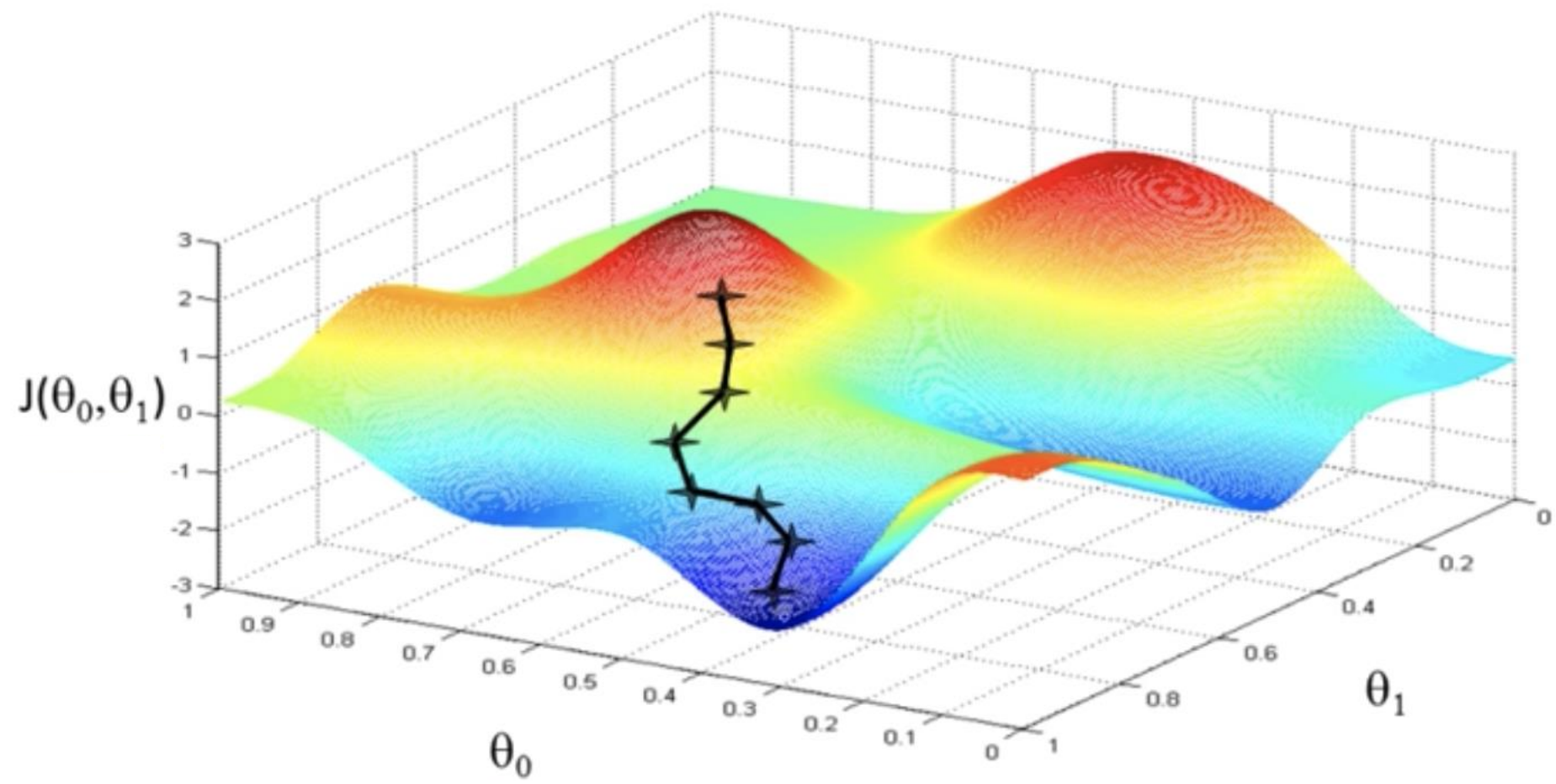


# Contour plot: central point



Which direction I should move if I want to go down (as rapidly as possible) ?







# Gradient descent algorithm

repeat until convergence {

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

}

learning rate

---

Correct: Simultaneous update

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

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

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

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

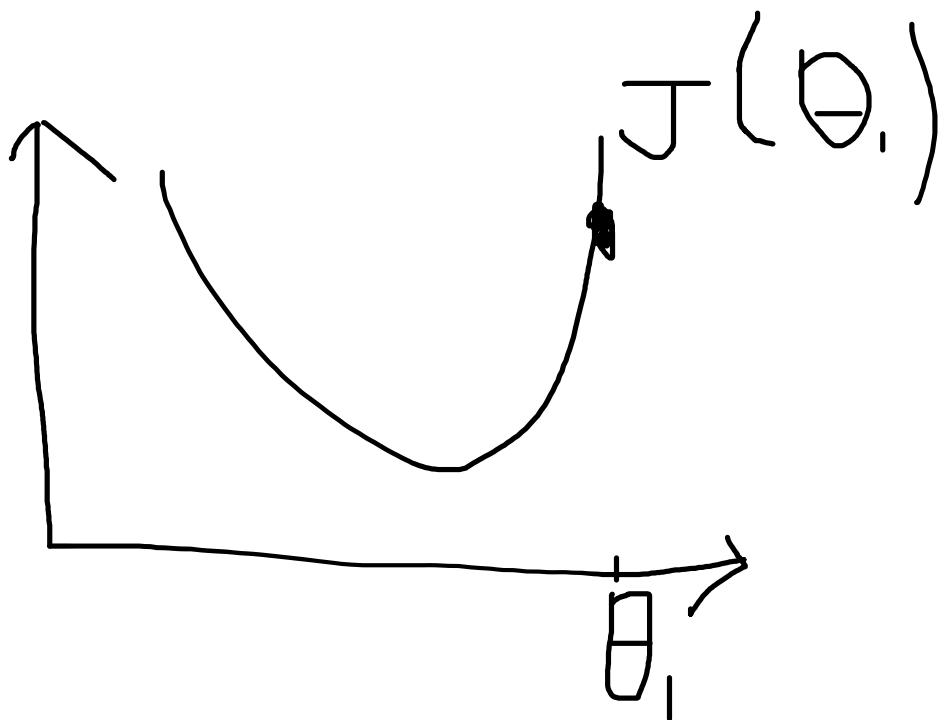
Incorrect:

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

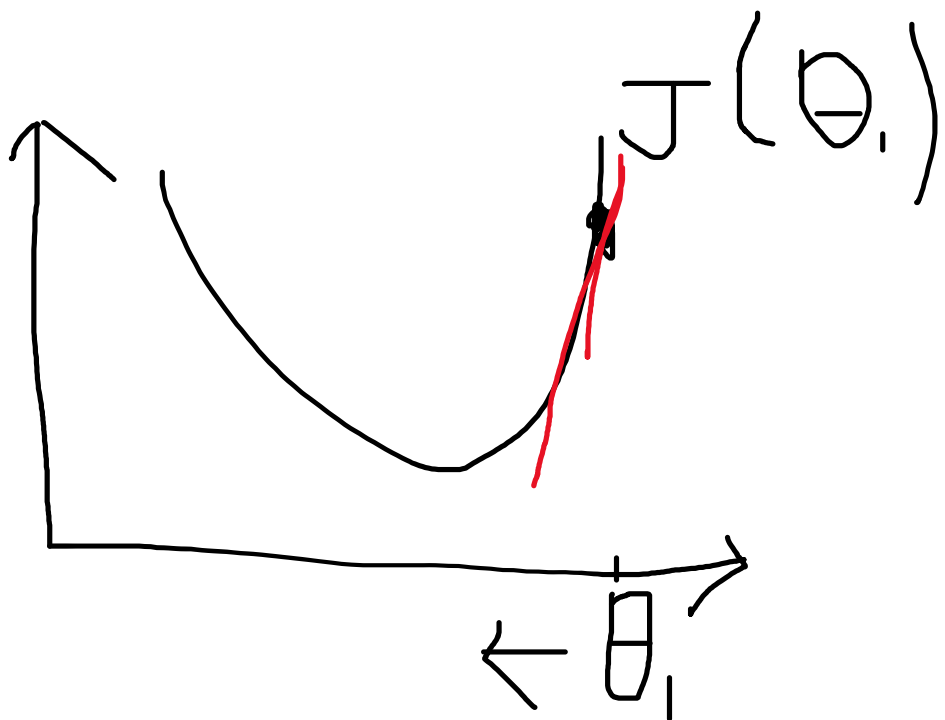
$$\theta_0 := \text{temp0}$$

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

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

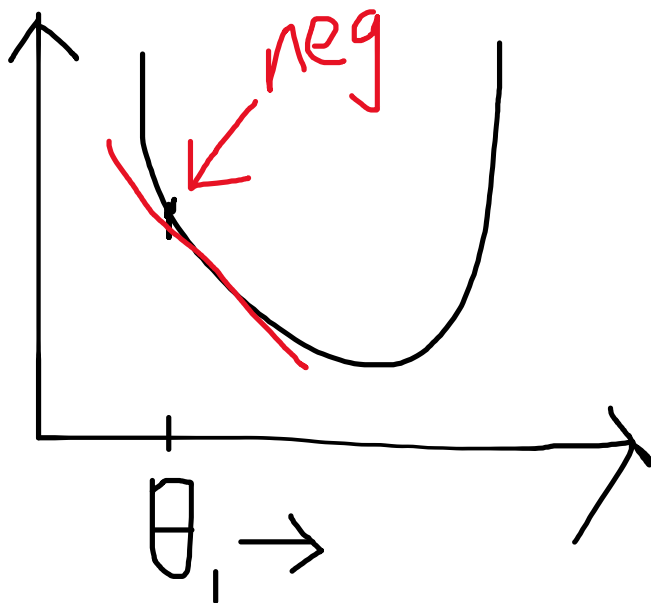


$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$



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

$$\theta_1 = \theta_1 - \alpha (\text{pos})$$

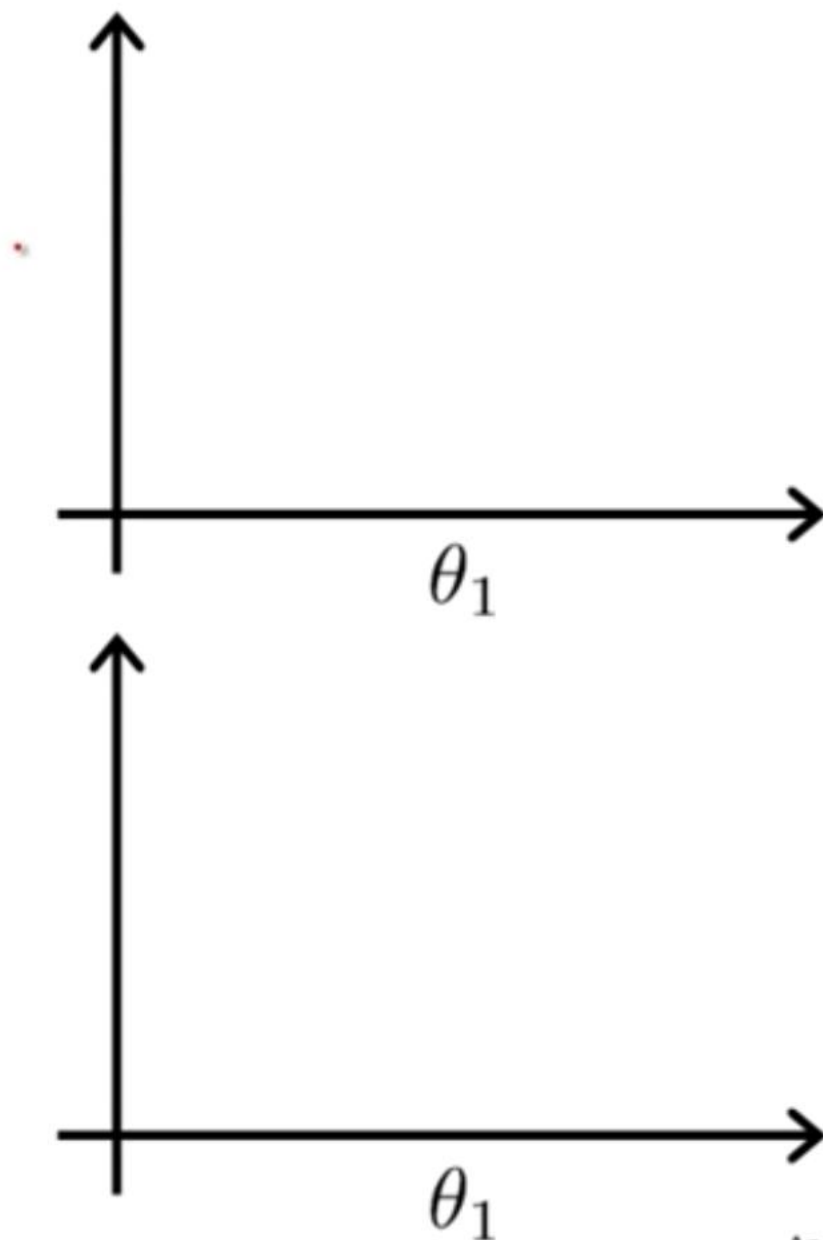


$$\theta_1 = \theta_1 - \alpha (\text{neg})$$

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

If  $\alpha$  is too small, gradient descent can be slow.

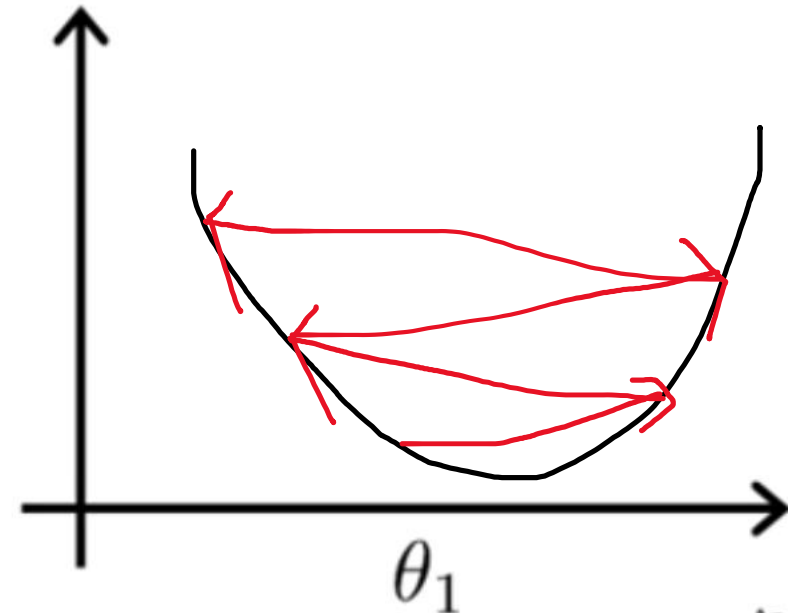
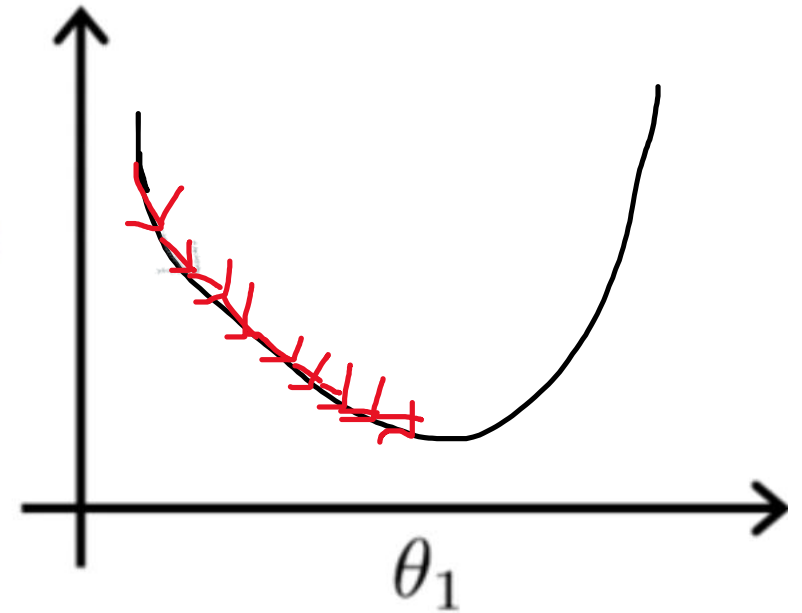
If  $\alpha$  is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



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

If  $\alpha$  is too small, gradient descent can be slow.

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



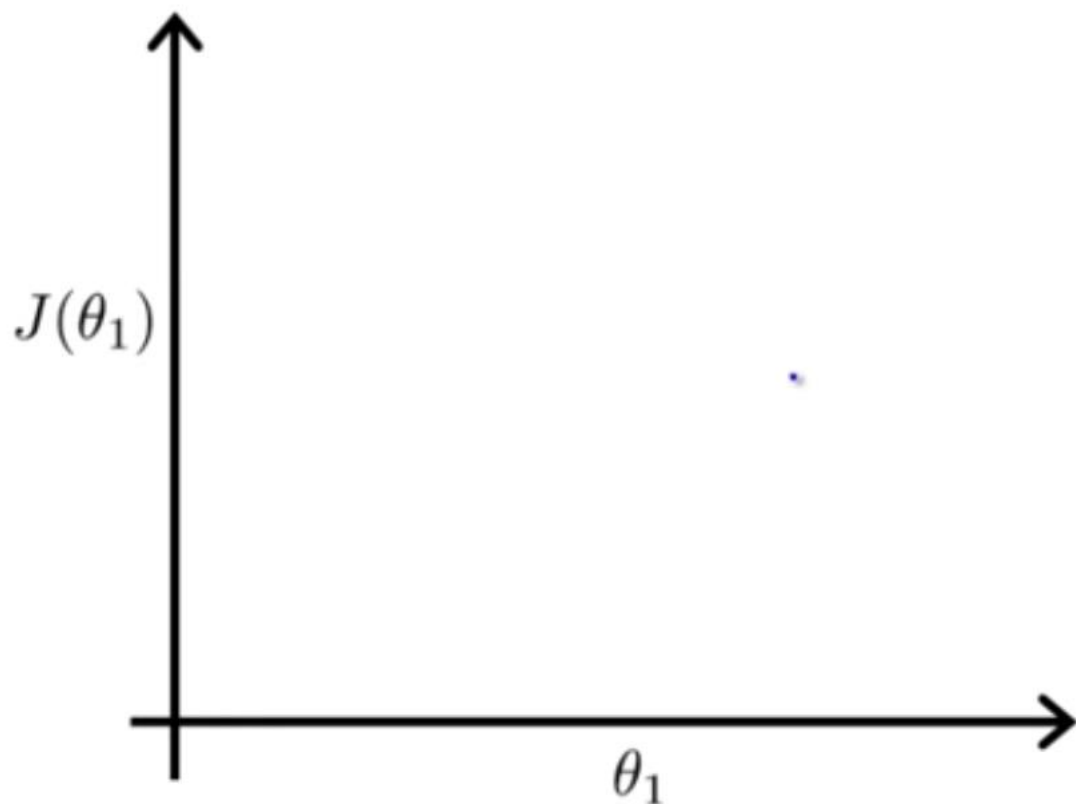
# Question?

- What if already in local optimum?

Gradient descent can converge to a local minimum, even with the learning rate  $\alpha$  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  $\alpha$  over time.

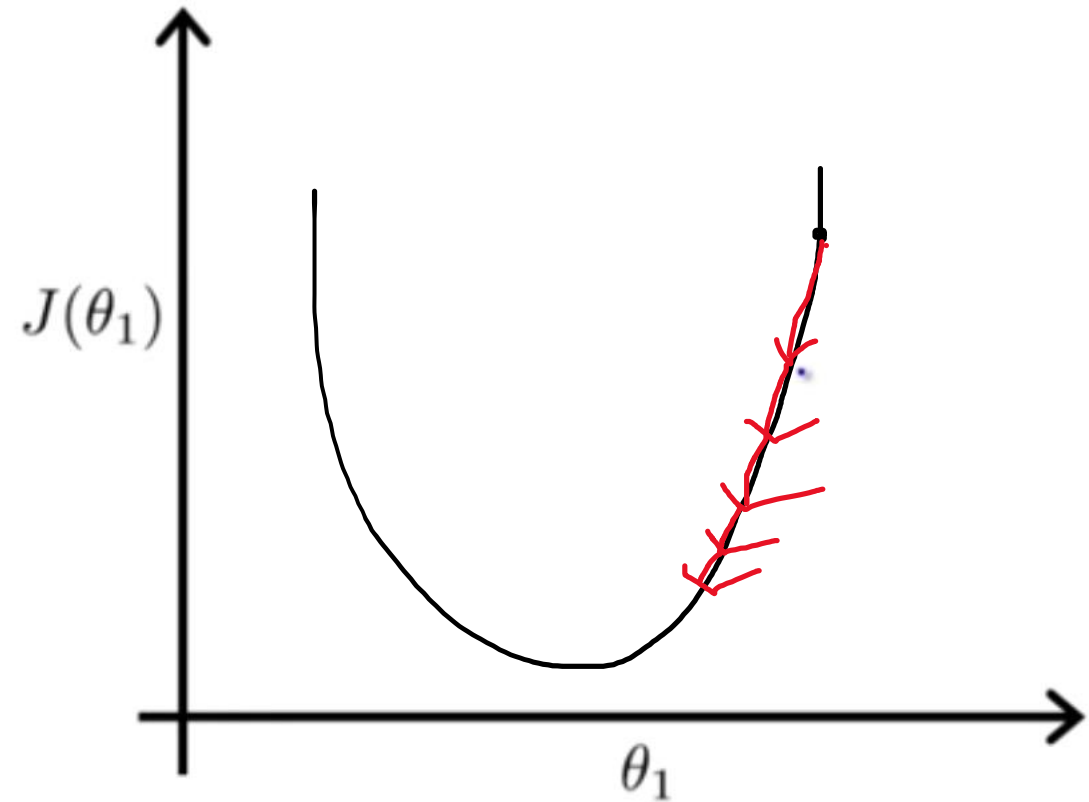




Gradient descent can converge to a local minimum, even with the learning rate  $\alpha$  fixed.

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

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease  $\alpha$  over time.



Now, we should apply GD to our problem

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

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

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

Now, we should apply GD to our problem

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

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

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

# Gradient Descent

- Initialize  $\theta$
- Repeat until convergence

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

simultaneous update  
for  $j = 0 \dots d$

For Linear Regression:  $\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left( h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$

# Gradient Descent

- Initialize  $\theta$
- Repeat until convergence

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

simultaneous update  
for  $j = 0 \dots d$

For Linear Regression: 
$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left( h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left( \sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2 \end{aligned}$$

# Gradient Descent

- Initialize  $\theta$
- Repeat until convergence

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

simultaneous update  
for  $j = 0 \dots d$

For Linear Regression: 
$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left( h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left( \sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left( \sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left( \sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \end{aligned}$$

# Gradient Descent

- Initialize  $\theta$
- Repeat until convergence

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

simultaneous update  
for  $j = 0 \dots d$

For Linear Regression: 
$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left( h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left( \sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left( \sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left( \sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \\ &= \frac{1}{n} \sum_{i=1}^n \left( \sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)} \end{aligned}$$

# Gradient Descent for Linear Regression

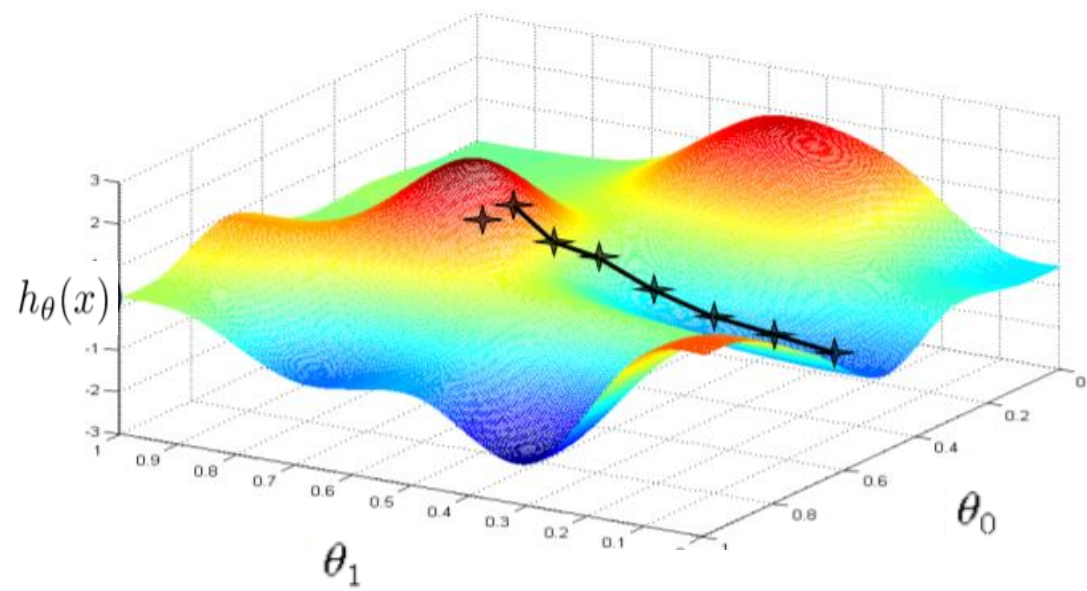
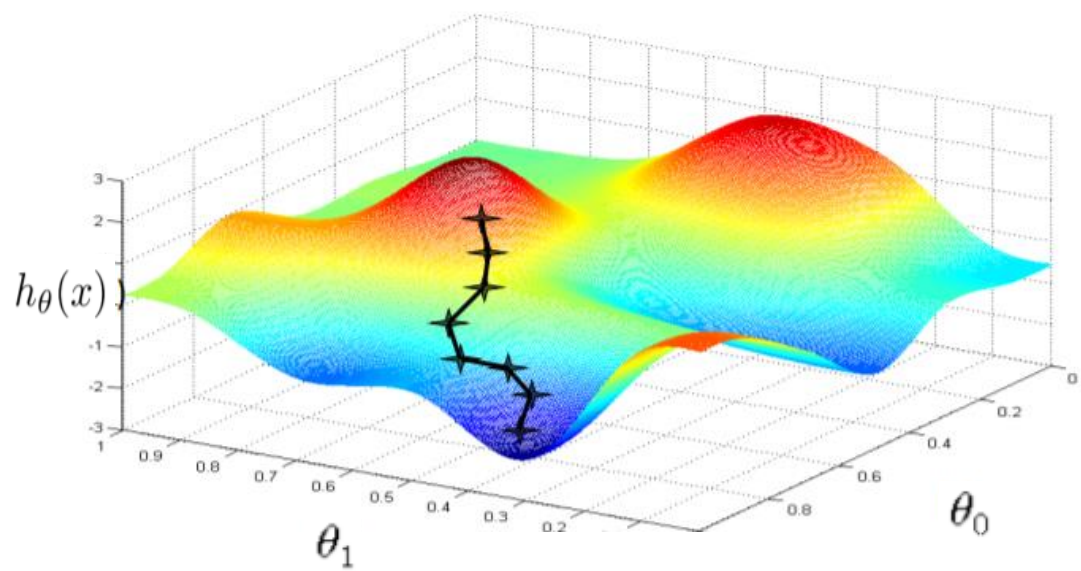
- Initialize  $\theta$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left( h_{\theta} \left( \mathbf{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \quad \begin{array}{l} \text{simultaneous} \\ \text{update} \\ \text{for } j = 0 \dots d \end{array}$$

- To achieve simultaneous update
  - At the start of each GD iteration, compute  $h_{\theta} \left( \mathbf{x}^{(i)} \right)$
  - Use this stored value in the update step loop
- Assume convergence when  $\|\theta_{new} - \theta_{old}\|_2 < \epsilon$

$$\text{L}_2 \text{ norm: } \quad \|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|\mathbf{v}|}^2}$$

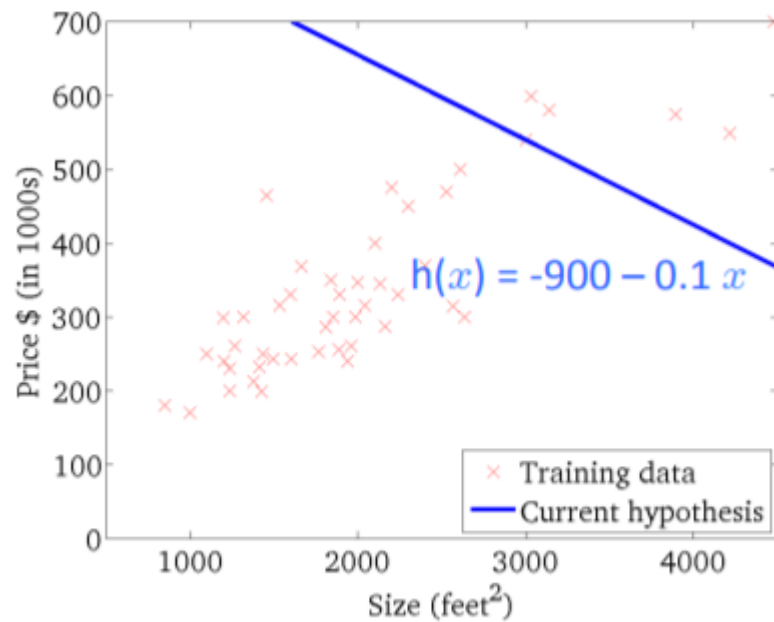




# Gradient Descent

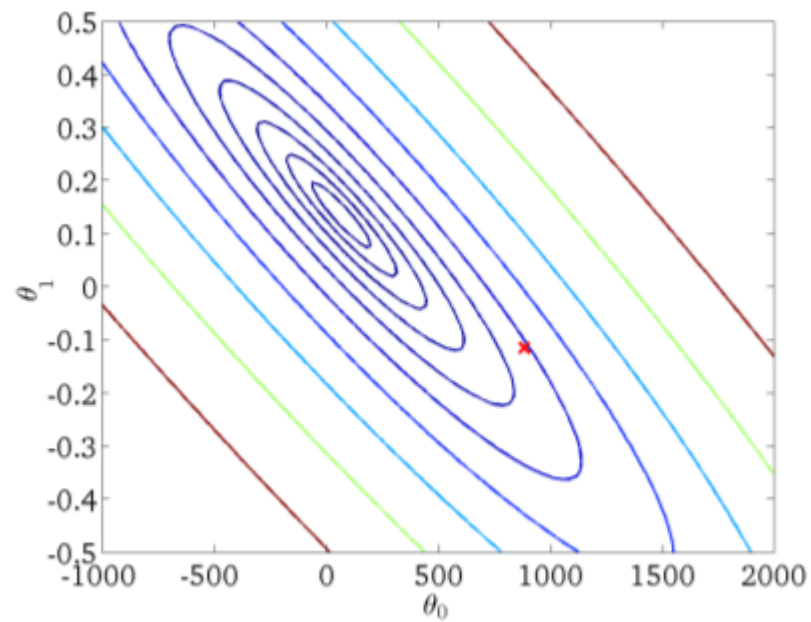
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

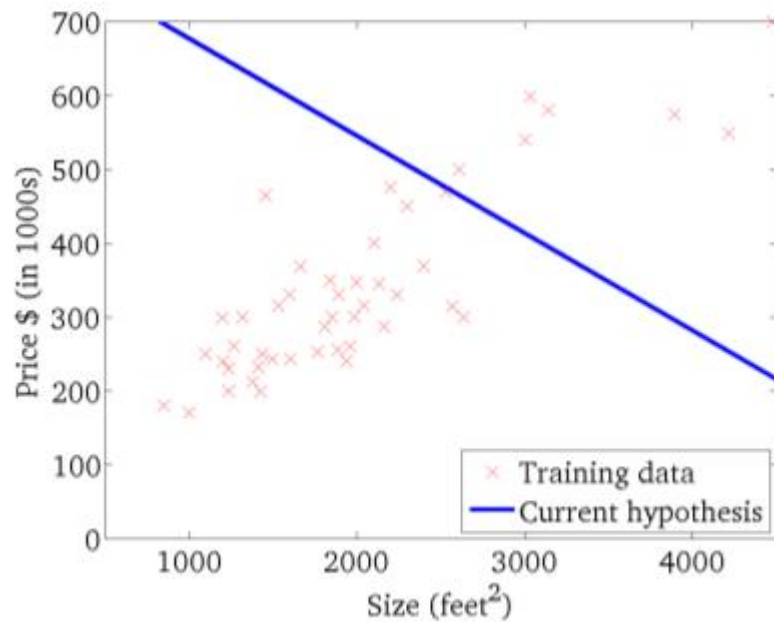
(function of the parameters  $\theta_0, \theta_1$ )



# Gradient Descent

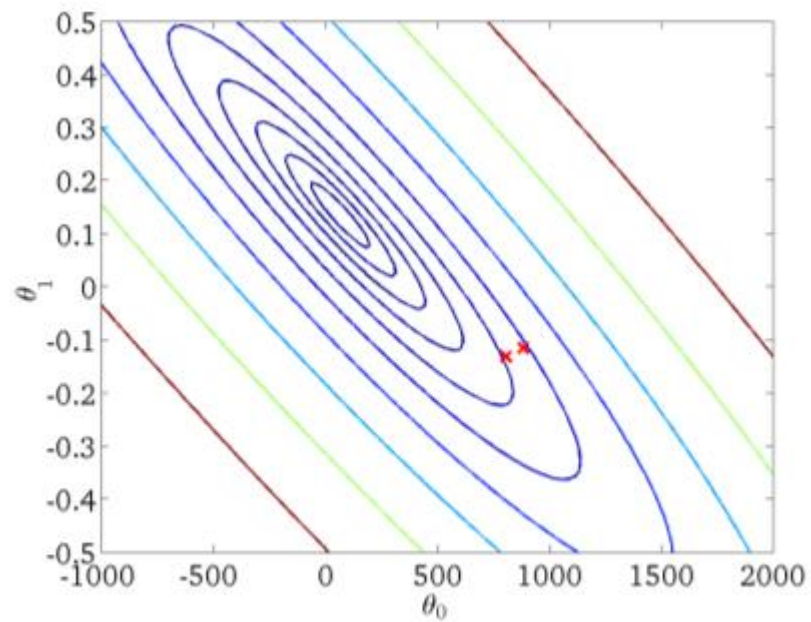
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

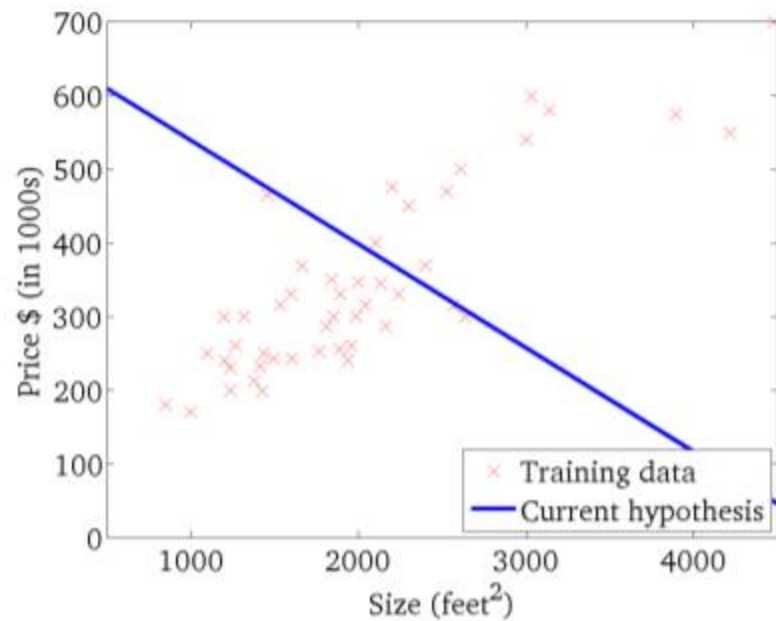
(function of the parameters  $\theta_0, \theta_1$ )



# Gradient Descent

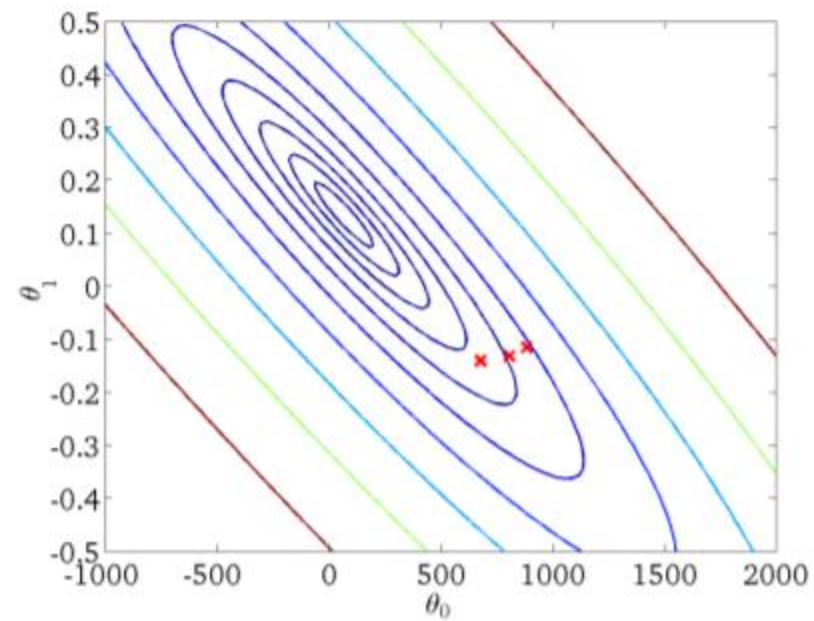
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

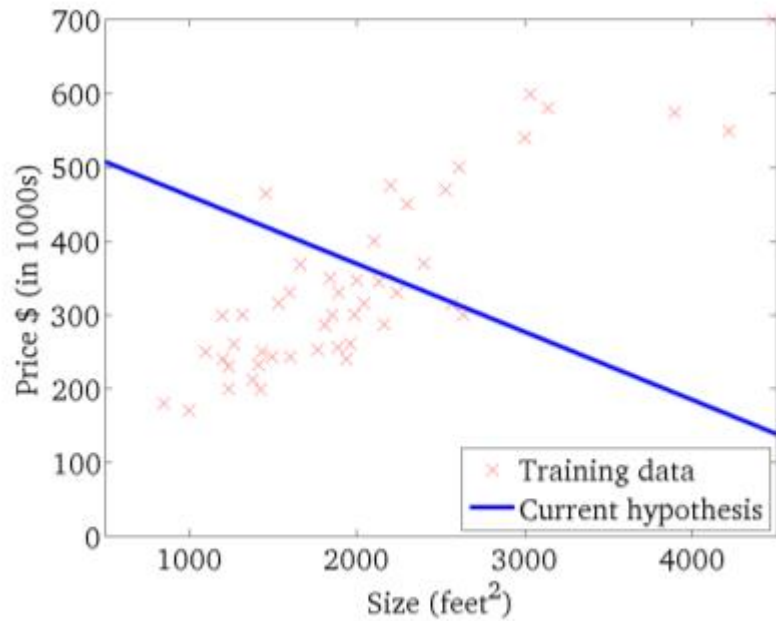
(function of the parameters  $\theta_0, \theta_1$ )



# Gradient Descent

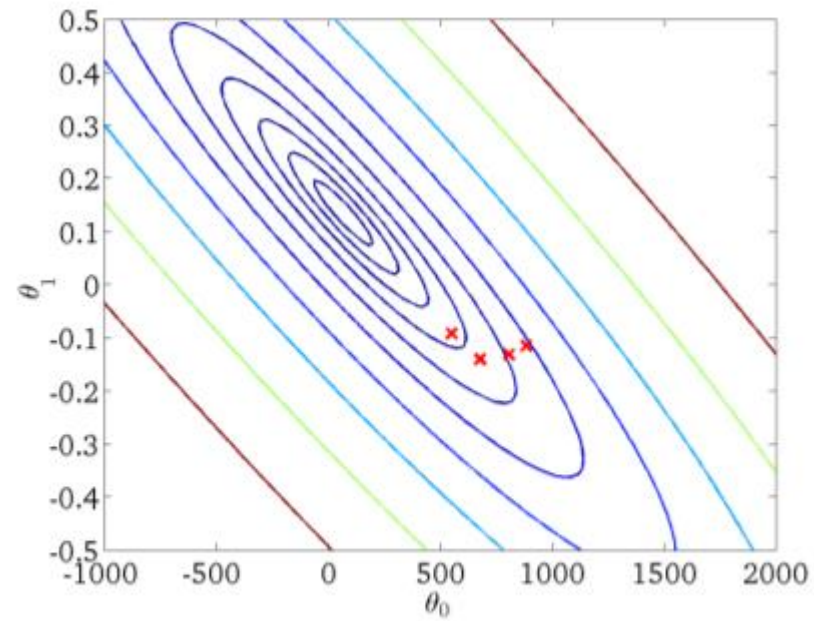
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )

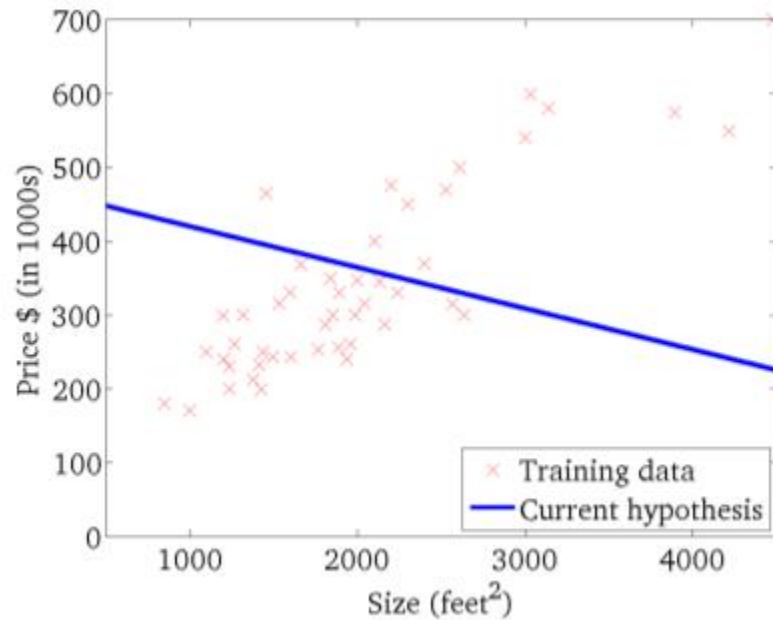




# Gradient Descent

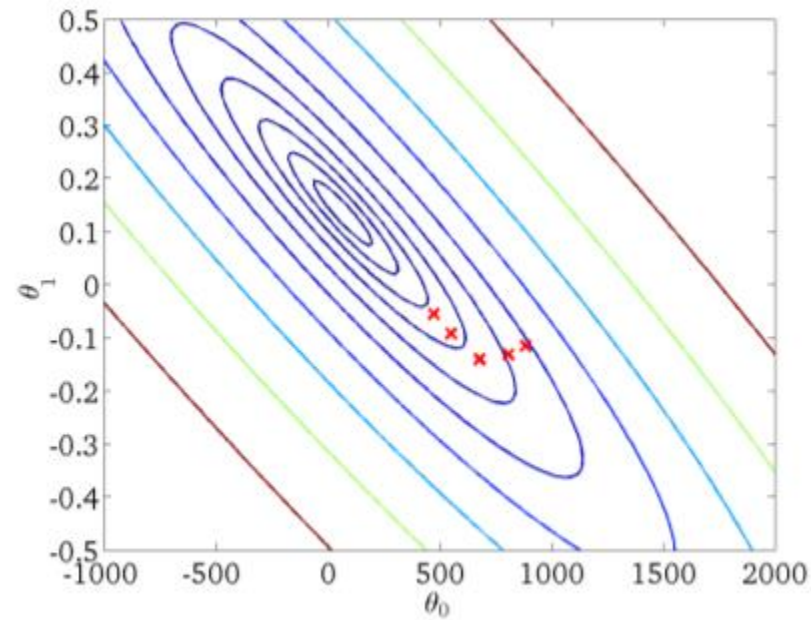
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

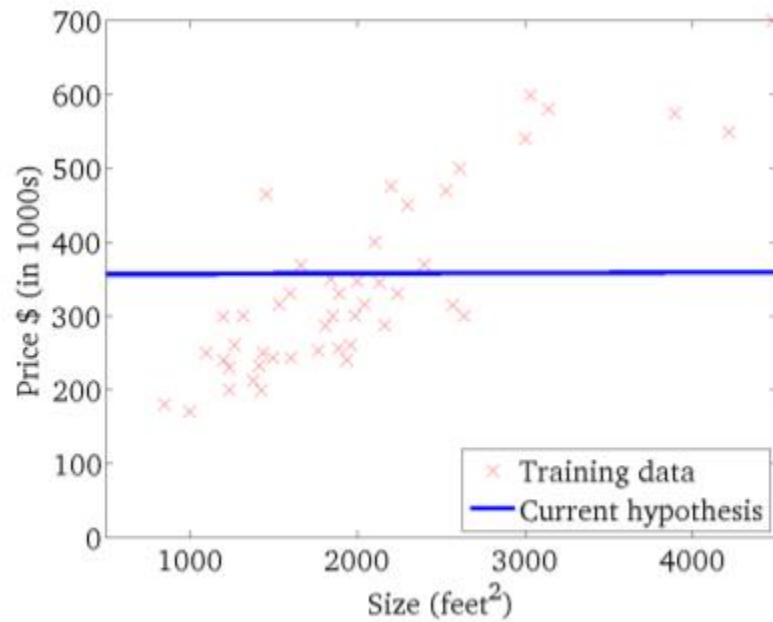
(function of the parameters  $\theta_0, \theta_1$ )



# Gradient Descent

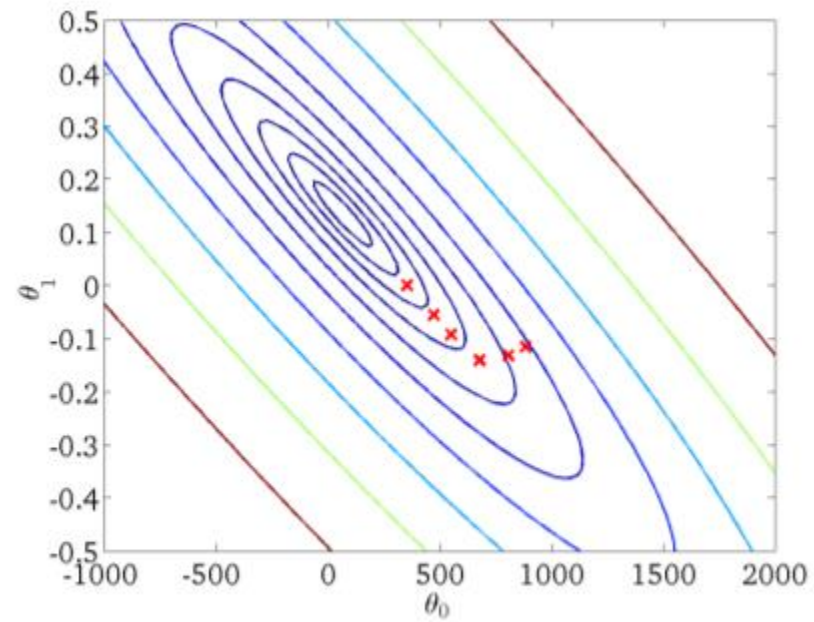
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

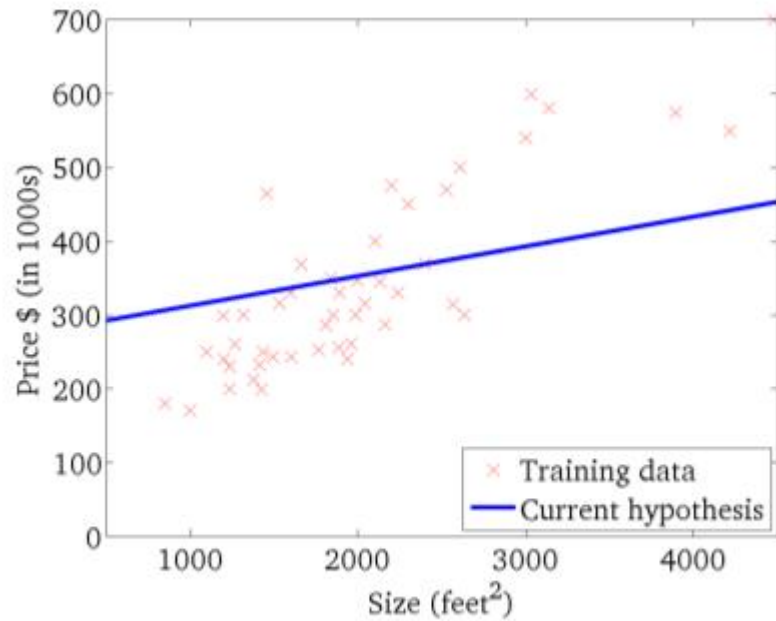
(function of the parameters  $\theta_0, \theta_1$ )



# Gradient Descent

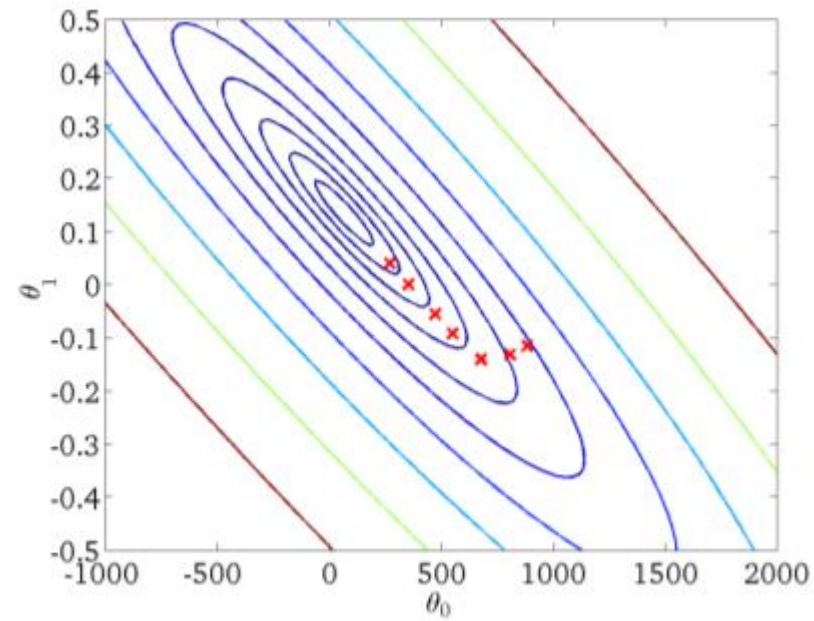
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )

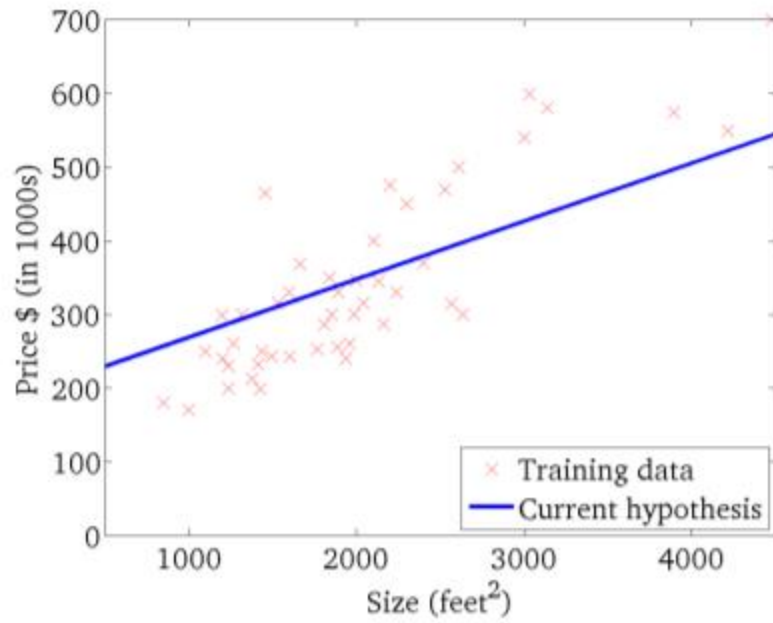




# Gradient Descent

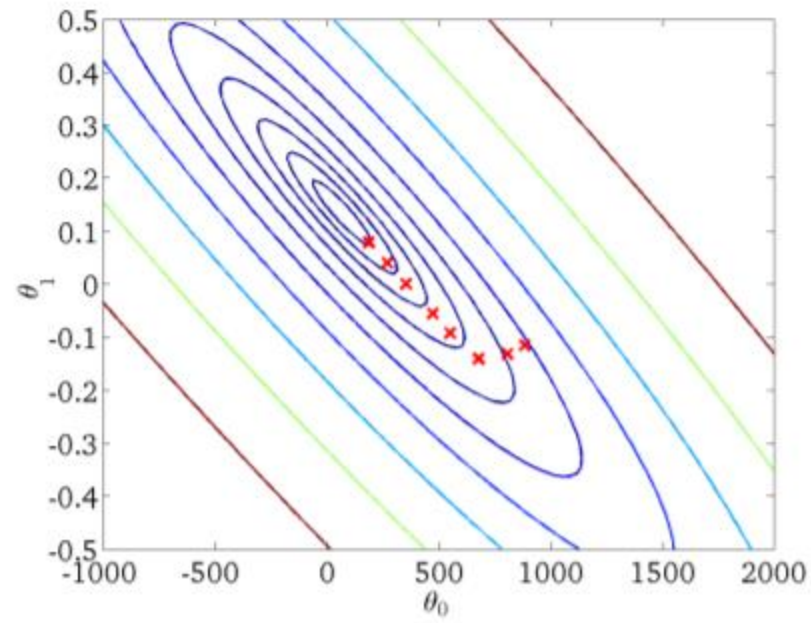
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

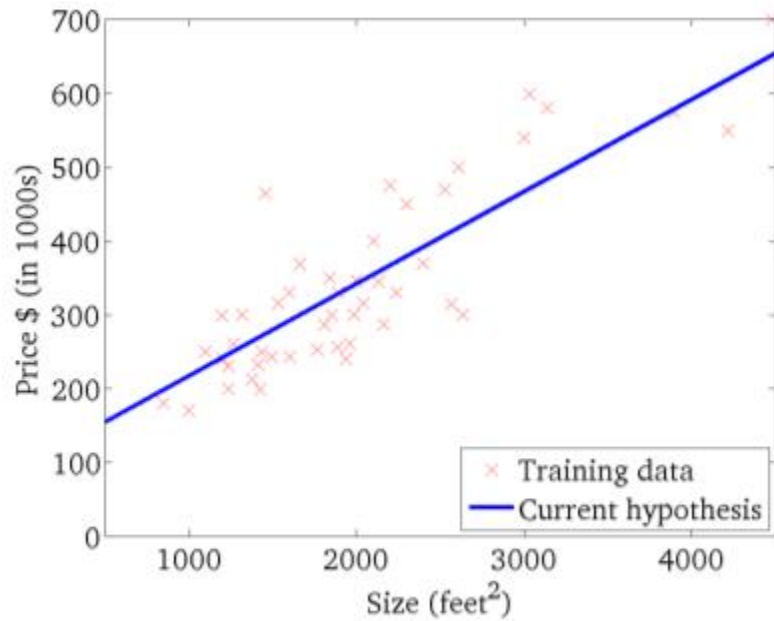
(function of the parameters  $\theta_0, \theta_1$ )



# Gradient Descent

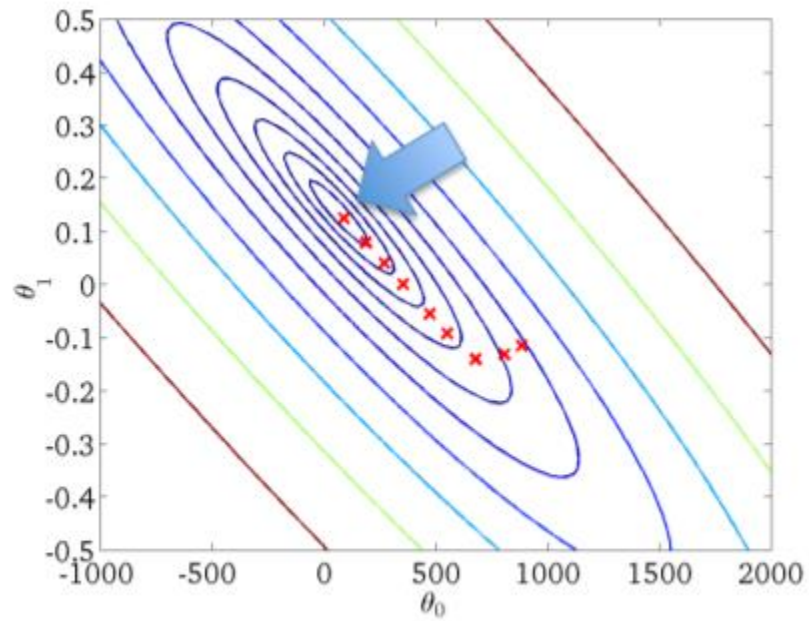
$$h_{\theta}(x)$$

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



# Extending Linear Regression to More Complex Models

- The inputs  $X$  for linear regression can be:
  - Original quantitative inputs
  - Transformation of quantitative inputs
    - e.g., log, exp, square root, square, etc.
  - Polynomial transformation
    - example:  $y = \theta_0 + \theta_1 * X + \theta_2 * X^2$

This allows use of linear regression techniques to fit non-linear datasets.

# Solve real problems

- How to train a predictor for beer rating?

## Beeradvocate

### Beers:



BA SCORE  
**100**  
world-class  
9,587 Ratings

THE BROS  
**95**  
world-class  
(view ratings)

Ratings: 9,587  
Reviews: 2,537  
rAvg: 4.59  
pDev: 9.59%  
Wants: 2,109  
Gots: 4,563 | FT: 472

Brewed by:  
Goose Island Beer Co.   
Illinois, United States


Style | ABV  
American Double / Imperial Stout | 13.80% ABV

Availability: Winter

Notes/Commercial Description:  
60 IBU  
(Beer added by: drewbage on 06-26-2003)

Displayed for educational use only;  
do not reuse.

### Ratings/reviews:



**4.35/5** rDev -5.2%  
look: 4 | smell: 4.25 | taste: 4.5 | feel: 4.25 | overall: 4.25

Serving: 355 mL bottle poured into a 9 oz Libbey Embassy snifter ("bottled on: 08AUG14 1109").

Appearance: Deep, dark near-black brown. Hazy, light brown fringe of foam and limited lacing; no head.

Smell: Roasted malt, vanilla, and some warming alcohol.

Taste: Roasted malts, cocoa, burnt caramel, molasses, vanilla and dark fruit. Bourbon barrel is hinted at but never takes over.

Mouthfeel: Medium to full body and light carbonation with a very lush, silky smooth feel.

Overall: Not as complex or intense as some newer barrel-aged stouts, but so smooth and balanced with all the elements tightly integrated.

HipCzech, Yesterday at 05:38 AM

### User profiles:



**HipCzech**  
Aficionado  
Male, from Texas

Profile Page

Member Since: **Jul 12, 2014** HipCzech was last seen:  
Points: **175** Today at 12:19 AM  
Beers: **108**  
Places: **6**  
Posts: **0**  
Likes Received: **0**  
Trading: **0% | 0**

50,000 beer rating reviews are available on

- [https://www.ece.villanova.edu/~xjiao/course/ECE5400/dataset/beer\\_50000.json](https://www.ece.villanova.edu/~xjiao/course/ECE5400/dataset/beer_50000.json)

# Example 1

---

- Using ordinary linear regression, train a predictor that uses the age ('ageInSeconds') to predict the beer rating ('review/overall'), i.e.,
  - $\text{review/overall} = \theta_0 + \theta_1 * \text{ageInSeconds}$
  - You may use Python libraries to do so. What are the fitted values of  $\theta_0$  and  $\theta_1$ ?

```
import numpy
import urllib
import urllib.request
import scipy.optimize
import random

def parseData(fname):
    for l in urllib.request.urlopen(fname):
        yield eval(l)

print ("Reading data")
data = list(parseData("https://www.ece.villanova.edu/~xjiao/course/ECE5400/dataset/beer_50000.json"))
print ("done")

### Do older people rate beer more highly? ###

data2 = [d for d in data if 'user/ageInSeconds' in d]

def feature(datum):
    feat = [1]
    feat.append(datum['user/ageInSeconds'])
    return feat

X = [feature(d) for d in data2]
y = [d['review/overall'] for d in data2]
theta, residuals, rank, s = numpy.linalg.lstsq(X, y)
print(theta)

~
```