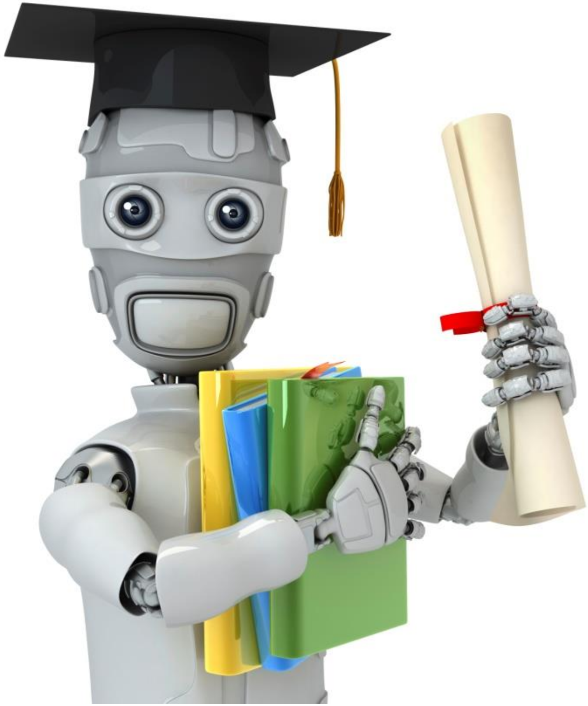


Chpt2. Linear Regression

Machine Learning

Outline

1. Applications(Task) and Model
2. Model Representation
3. Cost Functions of Task
4. Optimization

1. Linear Regression

- Linear regression predicts a real-valued output by a linear model based on an input vector or value.
  - 用途：定价（房屋、债券、股票）、资信、物质成分浓度
- 历史
  - 1855年，**高尔顿**发表《**遗传**的身高向平均数方向的回归》
  - 他和他的学生K·Pearson通过观察1078对夫妇，以每对夫妇的平均身高作为自变量，取他们的一个成年儿子的身高作为因变量，分析儿子身高与**父母身高**之间的关系，发现父母的身高可以预测**子女**的身高，两者近乎一条直线。
  - **“即使父母的身高都‘极端’高，其子女不见得会比父母高，而是有“衰退”（regression）（也称作“回归）至平均身高的倾向”**

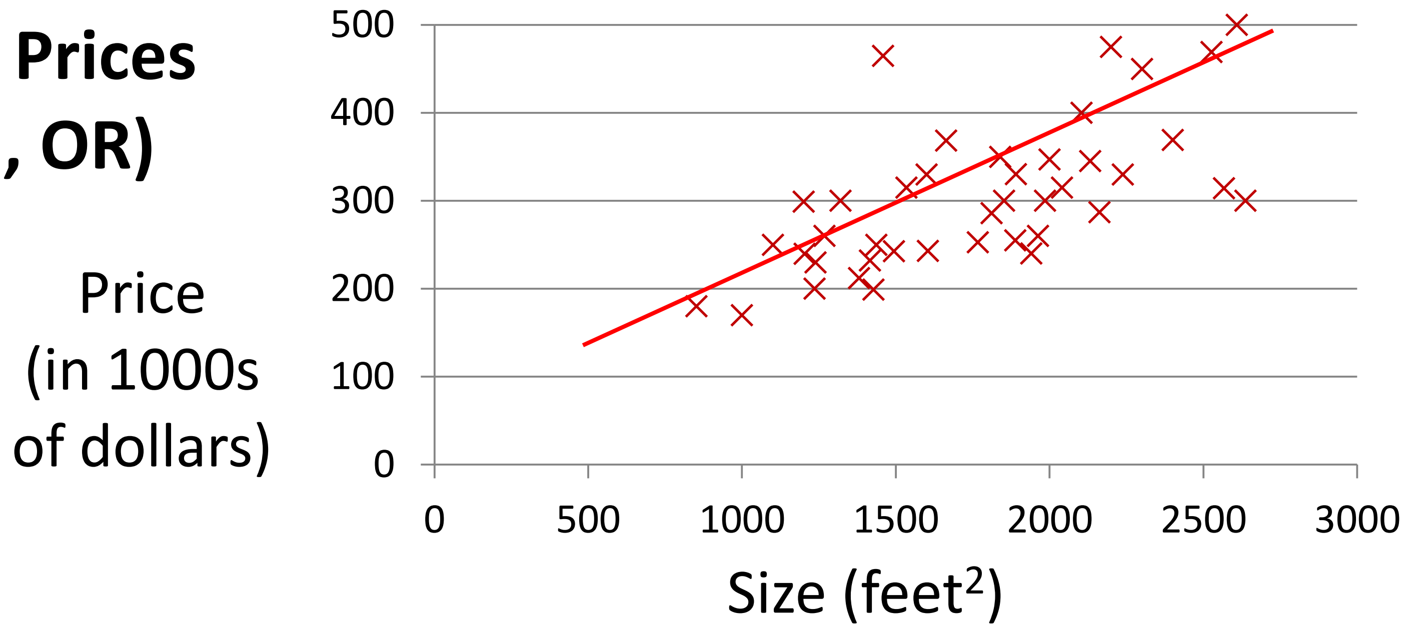
1. Application: Predicting Housing Prices (Portland, OR)

- Given some features of a house, such as
  - size,
  - numbers of bedrooms,
  - no. of floors,
  - age of home
- Predicting the housing price.

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

1. Housing Prices (Portland, OR)

将$\{(x^{(i)},y^{(i)})\}$特征化后，看作特征空间中的点，然后寻找线去拟合样本分布



<u>Supervised Learning</u>

Given the “right answer” for each example in the data.

<u>Regression Problem</u>

Predict real-valued output

<u>Classification Problem</u>

Predict discreted output

1. Training set of housing prices (Portland, OR)

	Size in feet^2 (x)	Price ($) in 1000's (y)
$m=41$	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

(x, y) : training sample

$(x^{(i)}, y^{(i)})$: i-th training sample

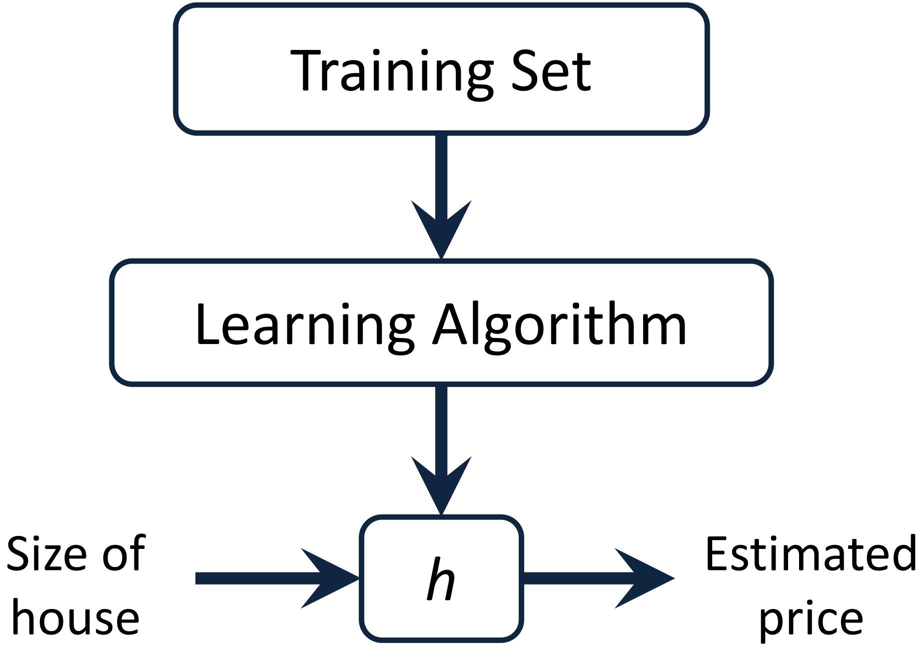
$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

$y^{(4)} =$

2. Model Representation of Linear Regression

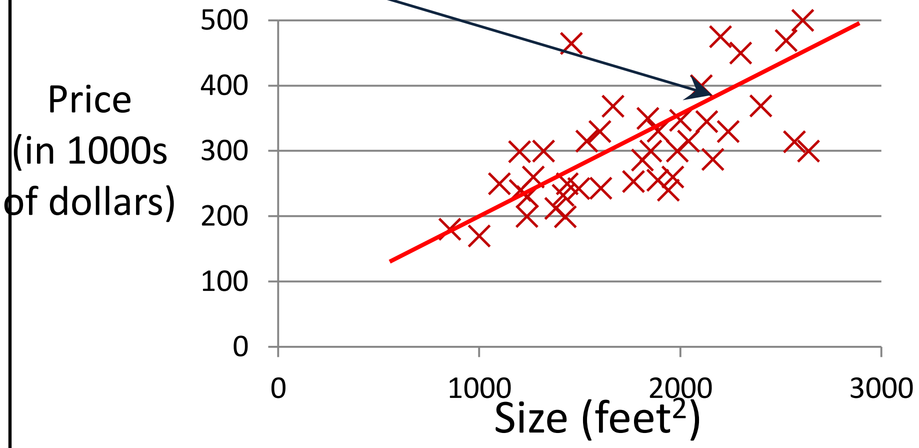


h (hypothesis,假设)：是从房间大小到房价的一个关系映射

映射可能是线性关系，也可能是指数、二次等。

How do we represent *h* ?

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



Linear regression with one variable.
Univariate linear regression.

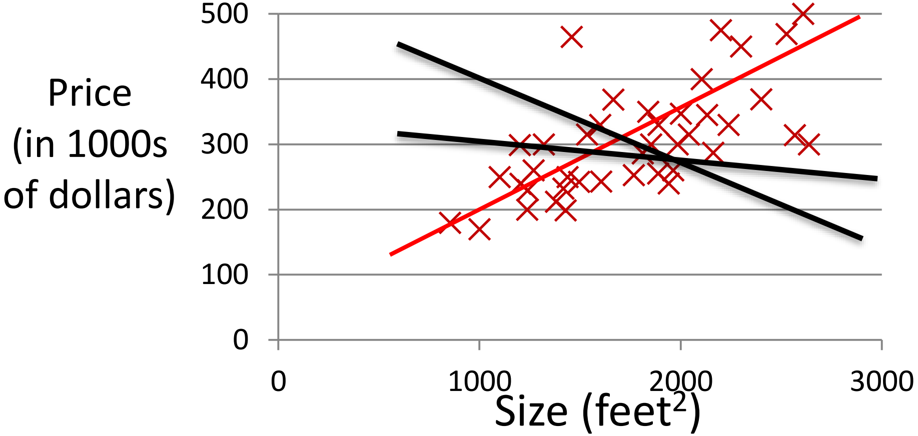
Many combinations of parameters θ_i can be chosen.

Which one is the best?

How to evaluate the best representation (parameters)?

How do we represent *h* ?

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

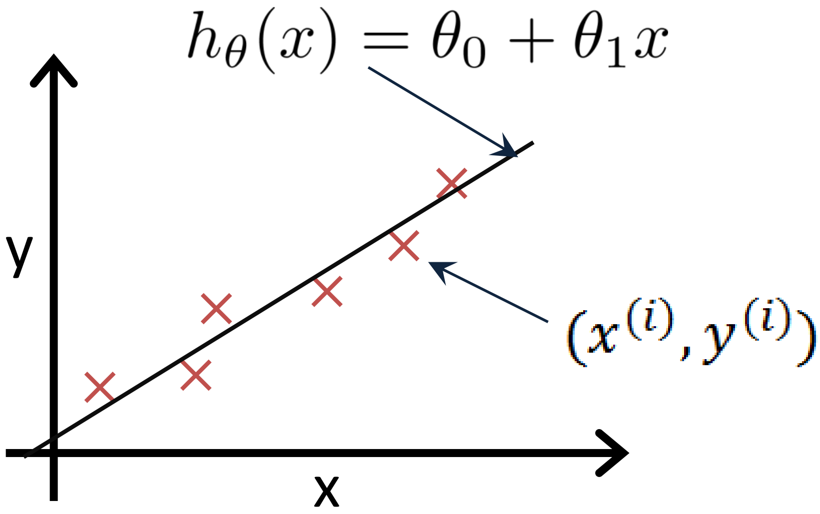


Linear regression with one variable.
Univariate linear regression.

Summary

- 机器学习的解决问题的方法
  - 从经验数据中总结经验的准备工作
    - 数据 v.s. 经验数据 （数据标注）
    - 什么是经验？
  - 如何从数据中学习到经验？ （机器学习）
    - 提假设(如线性)、总结数据、模型及参数表达经验
  - 如何应用经验？
    - 应用带参模型对目标任务进行预测

3. Cost Function



Cost Function:

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

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

Goal:

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

3. Simplified Cost Function

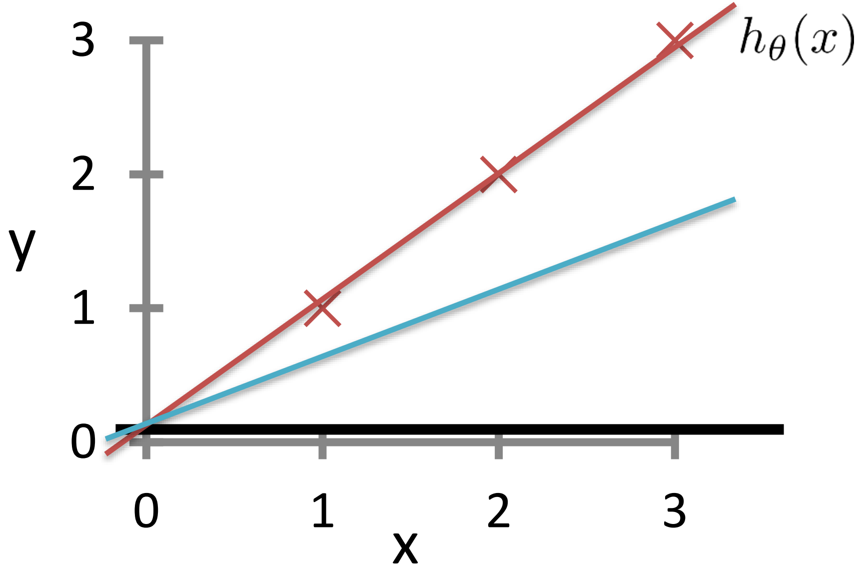
	Simplified
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$	$h_\theta(x) = \theta_1 x$
Parameters: θ_0, θ_1	θ_1
Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$	$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$	$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

Andrew Ng

3. Plot of Cost Function

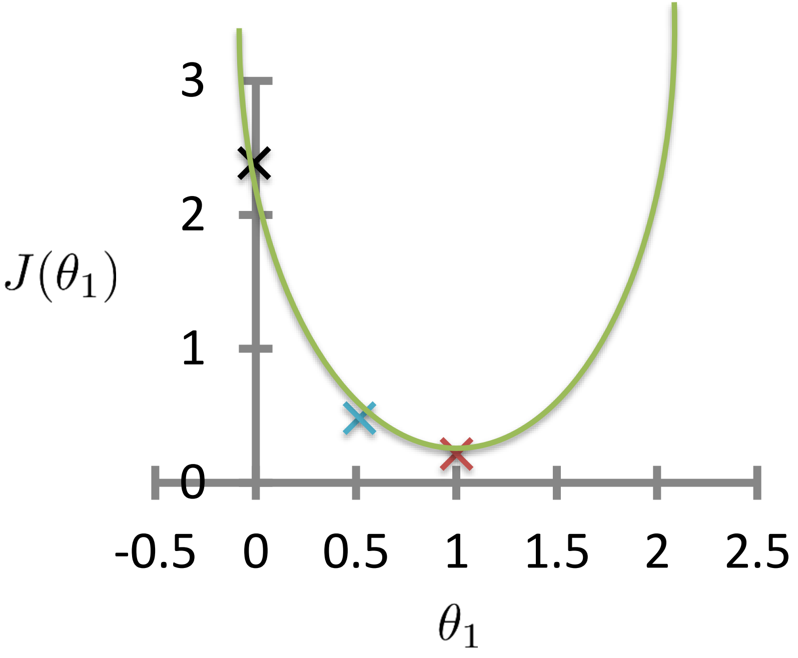
Predicting function: $h_\theta(x)$

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



Loss function: $J(\theta_1)$

(function of the parameter θ_1)

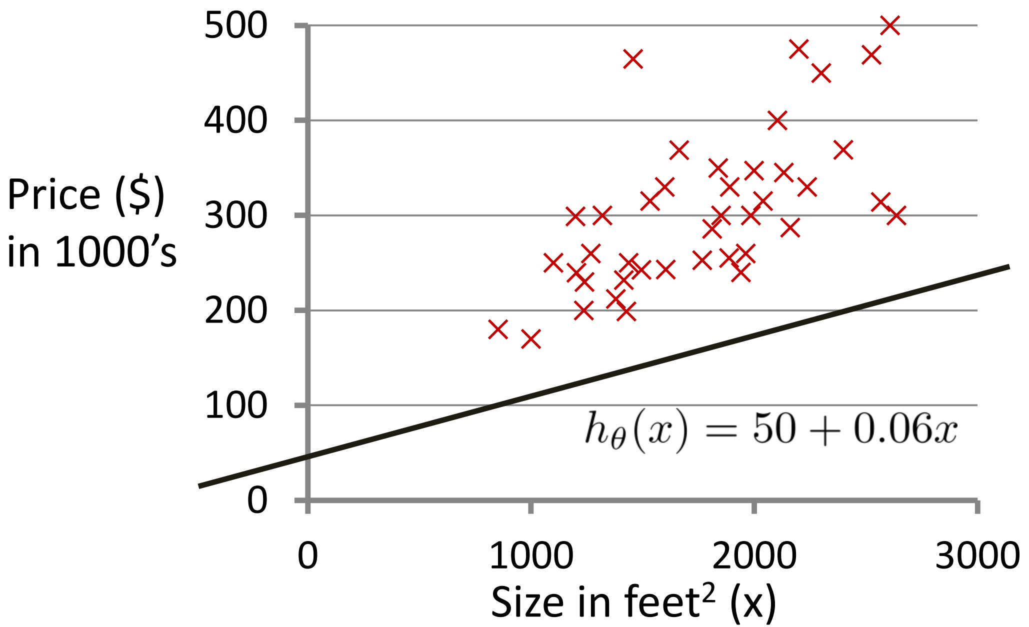


Andrew Ng

Cost Function of LR with One Variable

$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

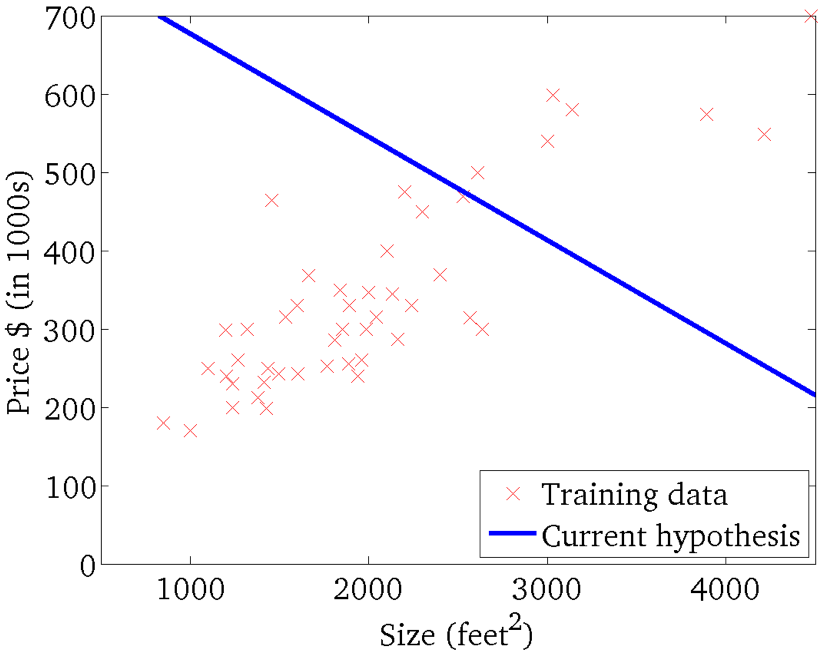
(function of the parameters θ_0, θ_1)

Andrew Ng

Contour Plot of LR's Cost Function

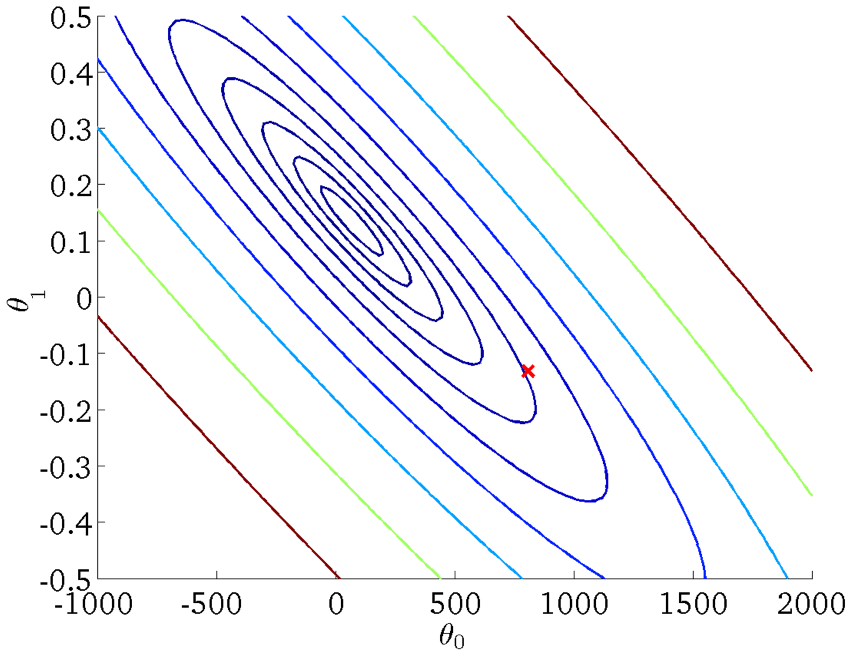
$$h_\theta(x)$$

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



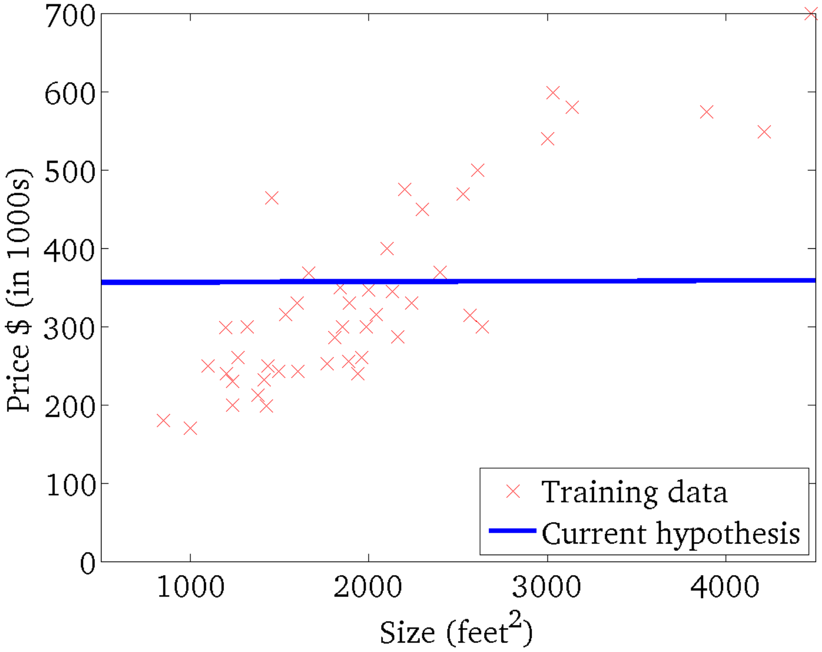
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



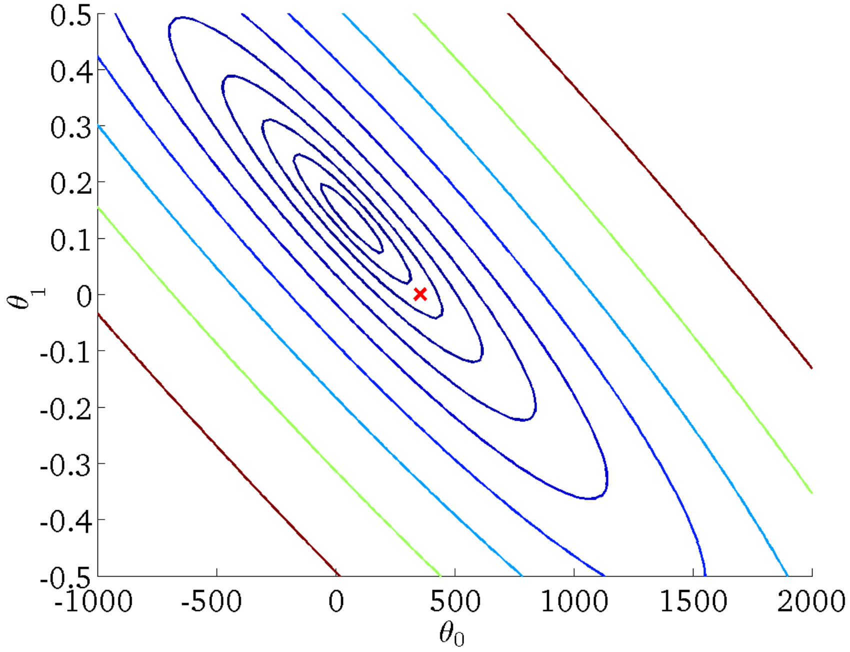
$$h_\theta(x)$$

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



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

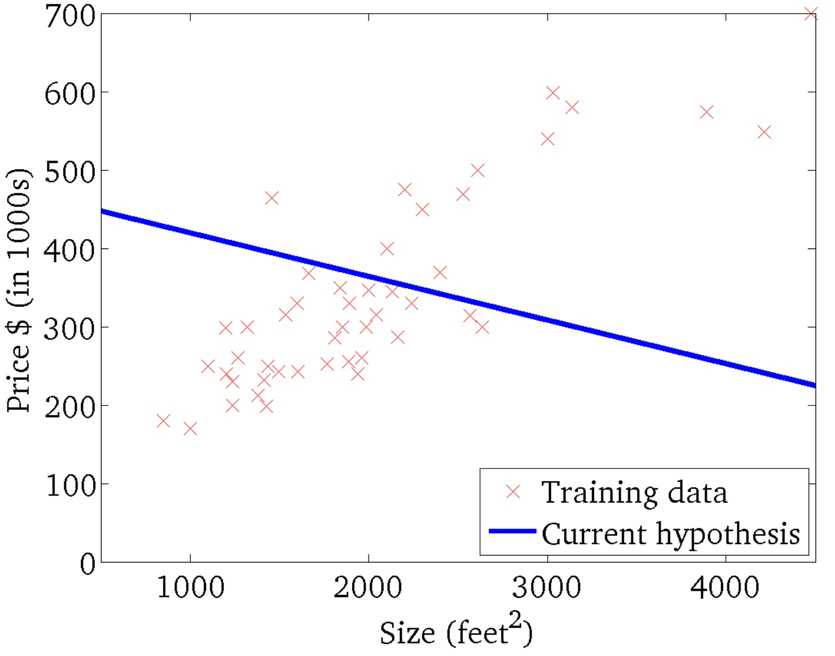
(function of the parameters θ_0, θ_1)



Andrew Ng

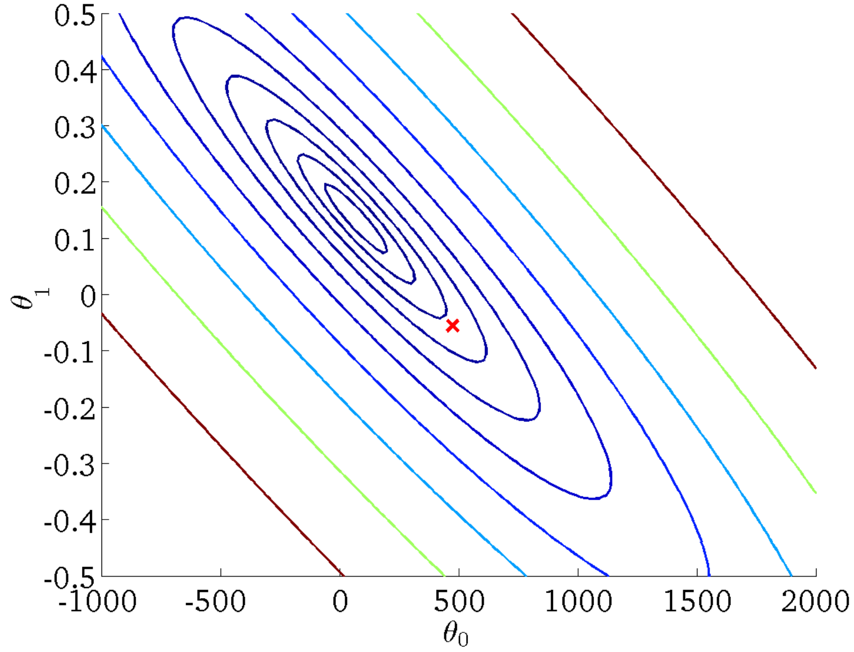
$h_\theta(x)$

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



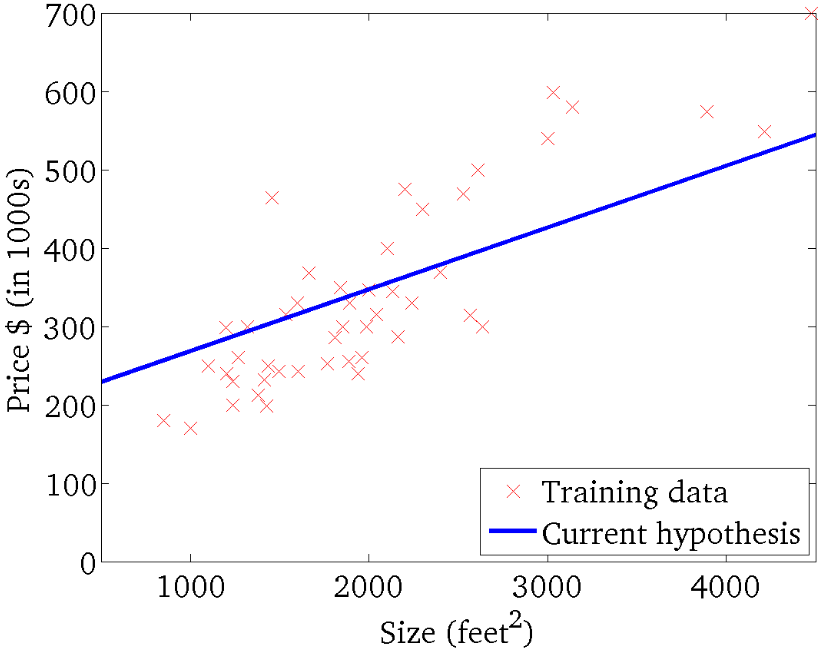
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



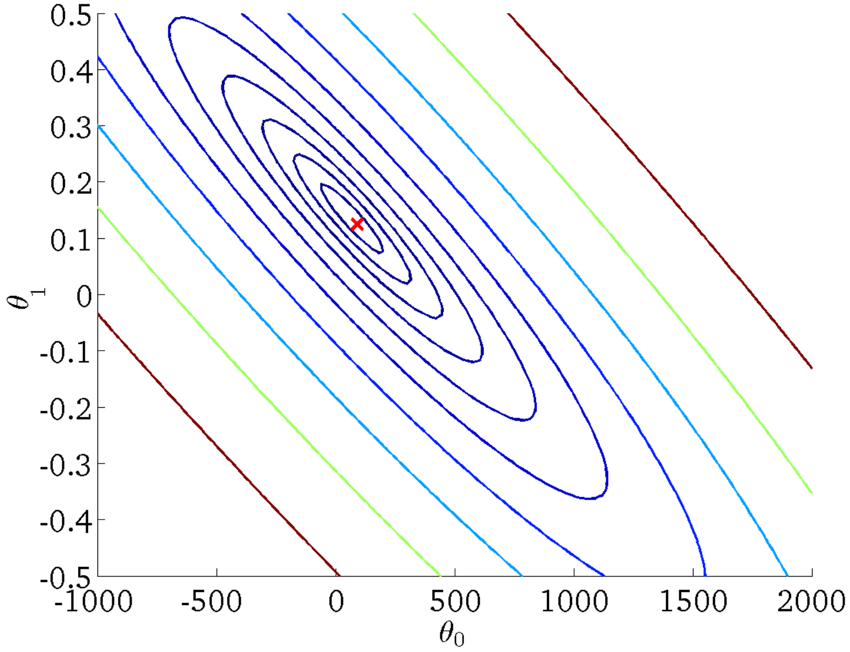
$h_\theta(x)$

(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)

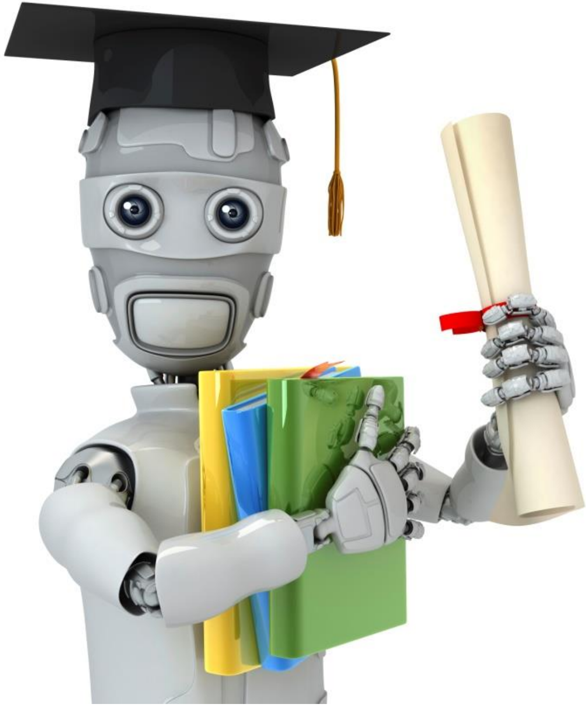


Andrew Ng

Linear Regression with One Variable

1. Represent the prediction model by using a linear function
2. Measure the models (i.e. parameters of linear function) with cost function
3. How to minimize the cost function to get the best model?

Minimize the cost function J(theta) with Gradient Descent



Machine Learning

Linear regression with one variable

Gradient descent

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

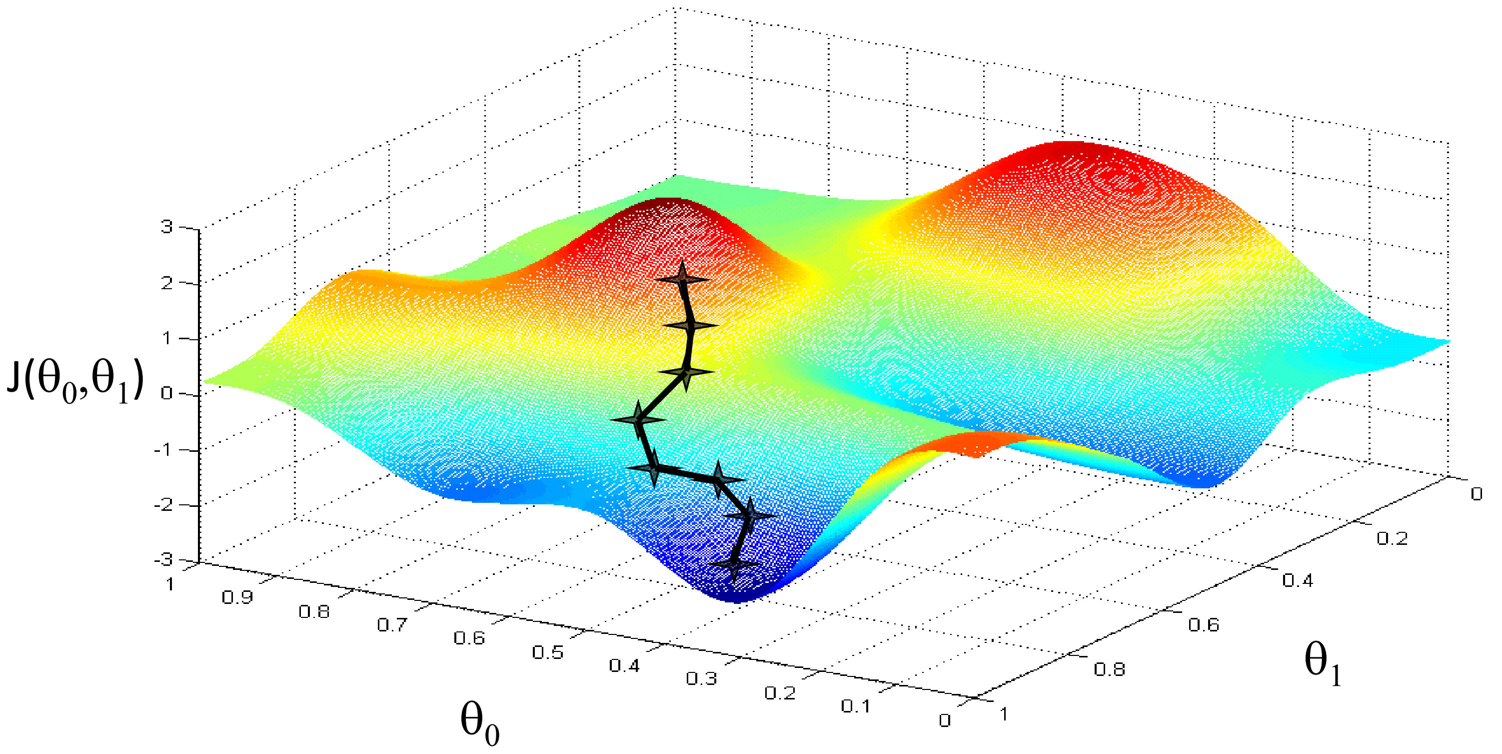
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

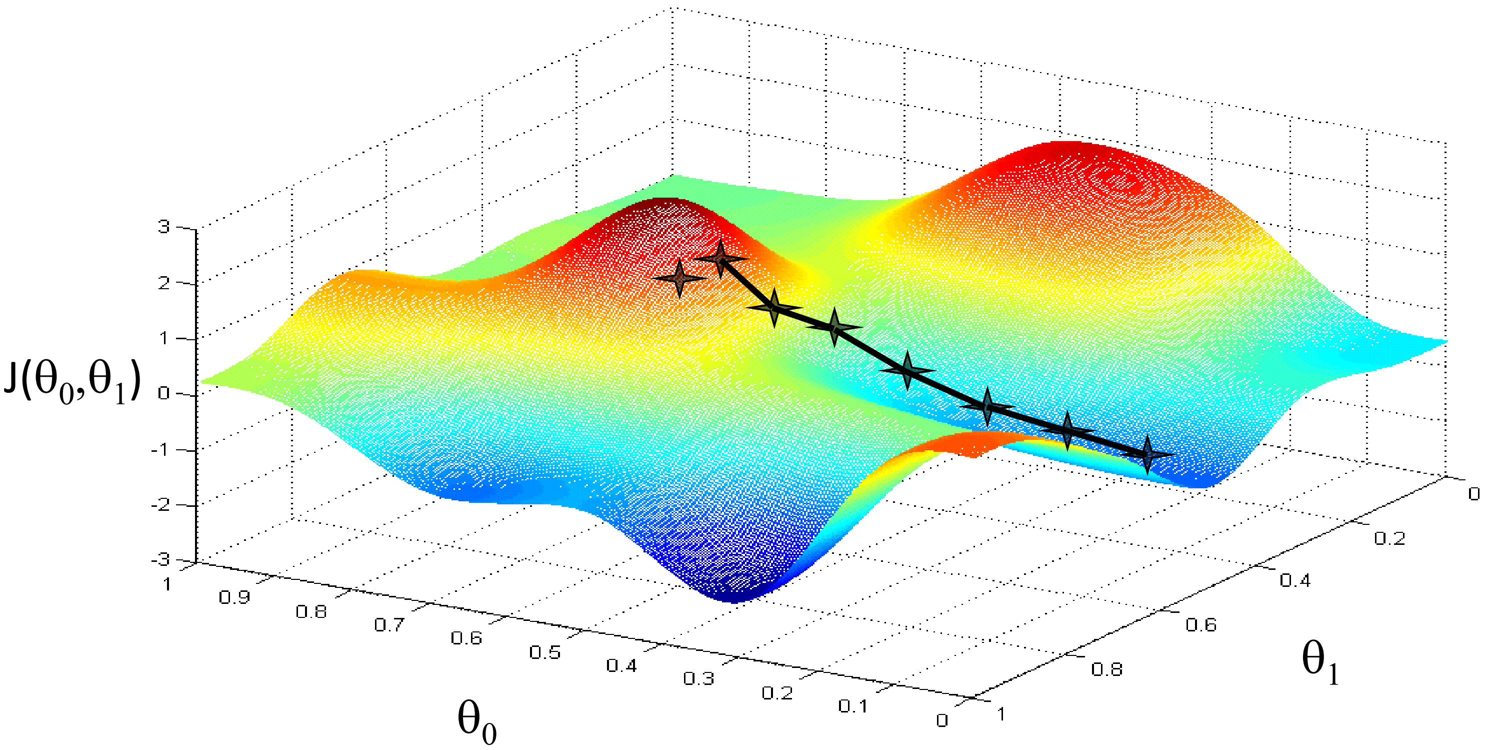
- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

  until we hopefully end up at a minimum

Gradient Descent Reflected by Different Initial Value 1/2



Gradient Descent Reflected by Different Initial Value 2/2



Andrew Ng

Gradient descent algorithm

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \underbrace{\alpha}_{\text{Learning Rate}} \underbrace{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}_{\text{Derivative}} \quad \text{(simultaneously update } j = 0 \text{ and } j = 1)$$
$$\}$$

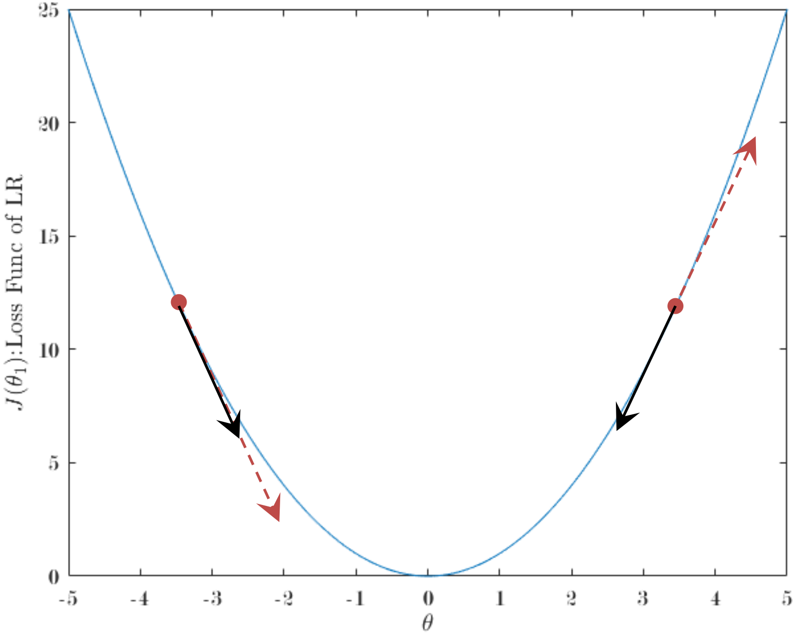
Correct: Simultaneous update

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_0 := \text{temp0}$
$\theta_1 := \text{temp1}$

Incorrect:

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\theta_0 := \text{temp0}$
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_1 := \text{temp1}$

Illustration of Gradient Descent for Minimizing the LF



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

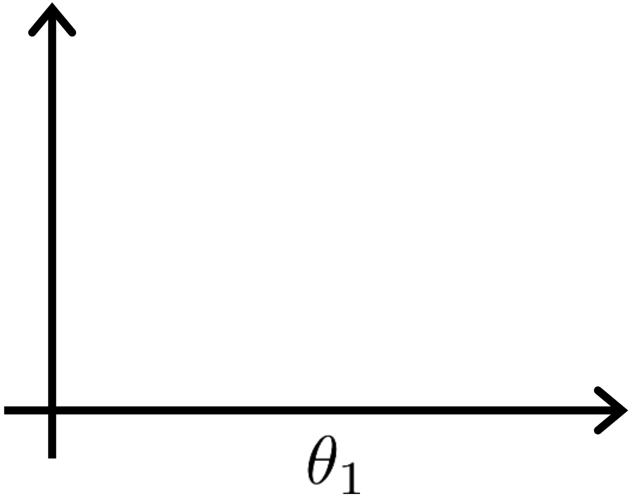
when $\dfrac{\partial J(\theta_1)}{\partial \theta_1}$ is positive, θ_1 will be decreased.

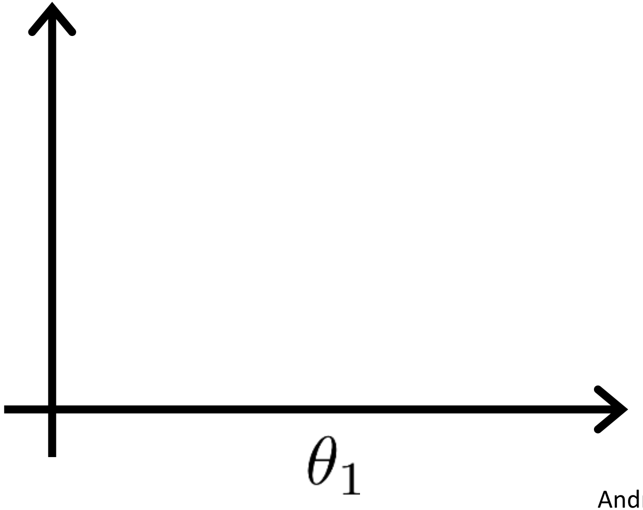
when $\dfrac{\partial J(\theta_1)}{\partial \theta_1}$ is negative, θ_1 will be increased.

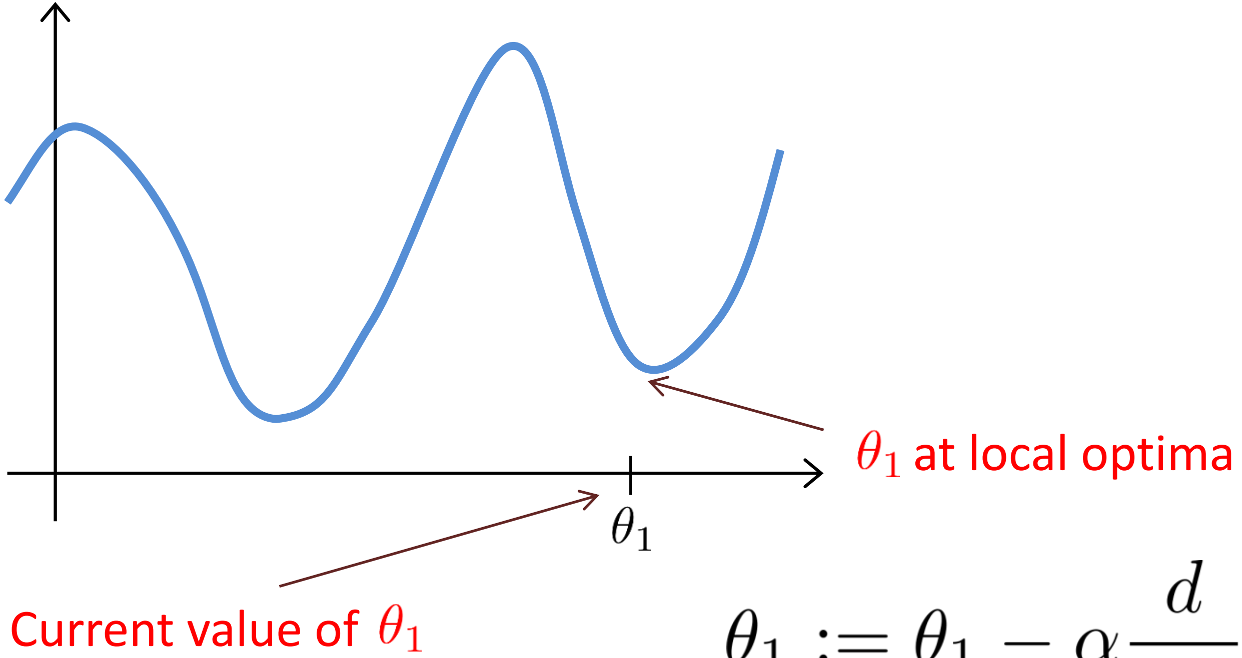
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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.





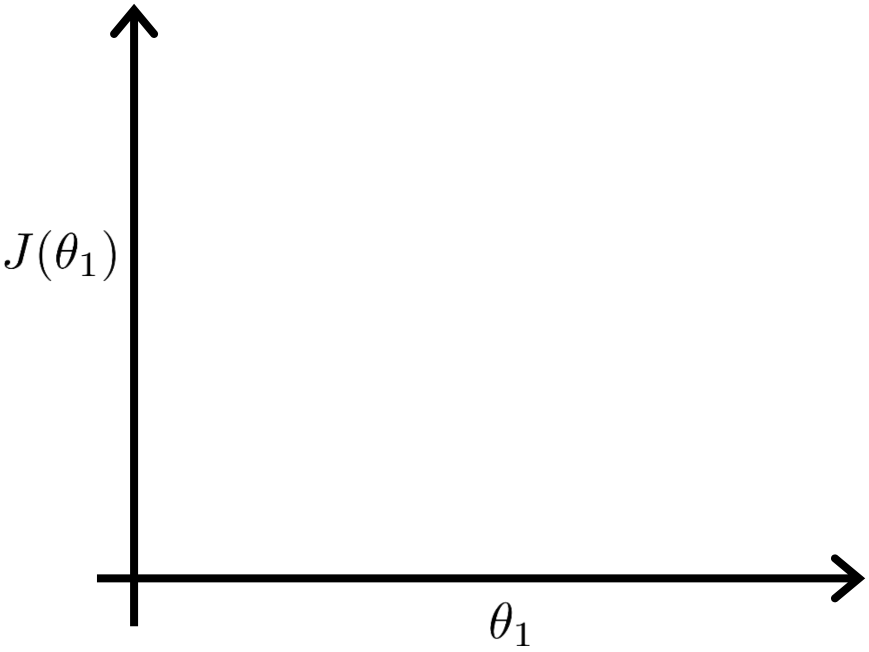


$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

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.



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

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

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

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

Gradient descent algorithm

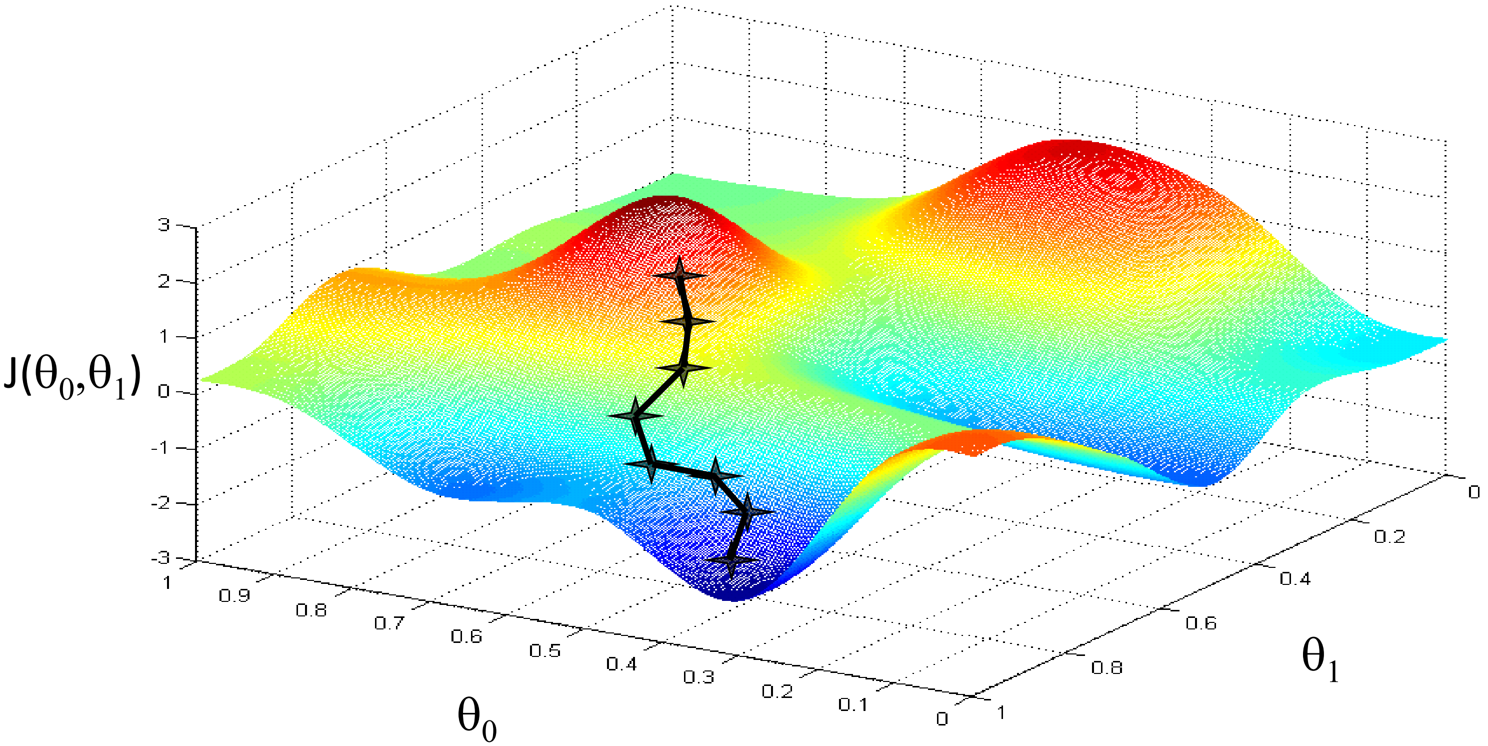
repeat until convergence $\{$

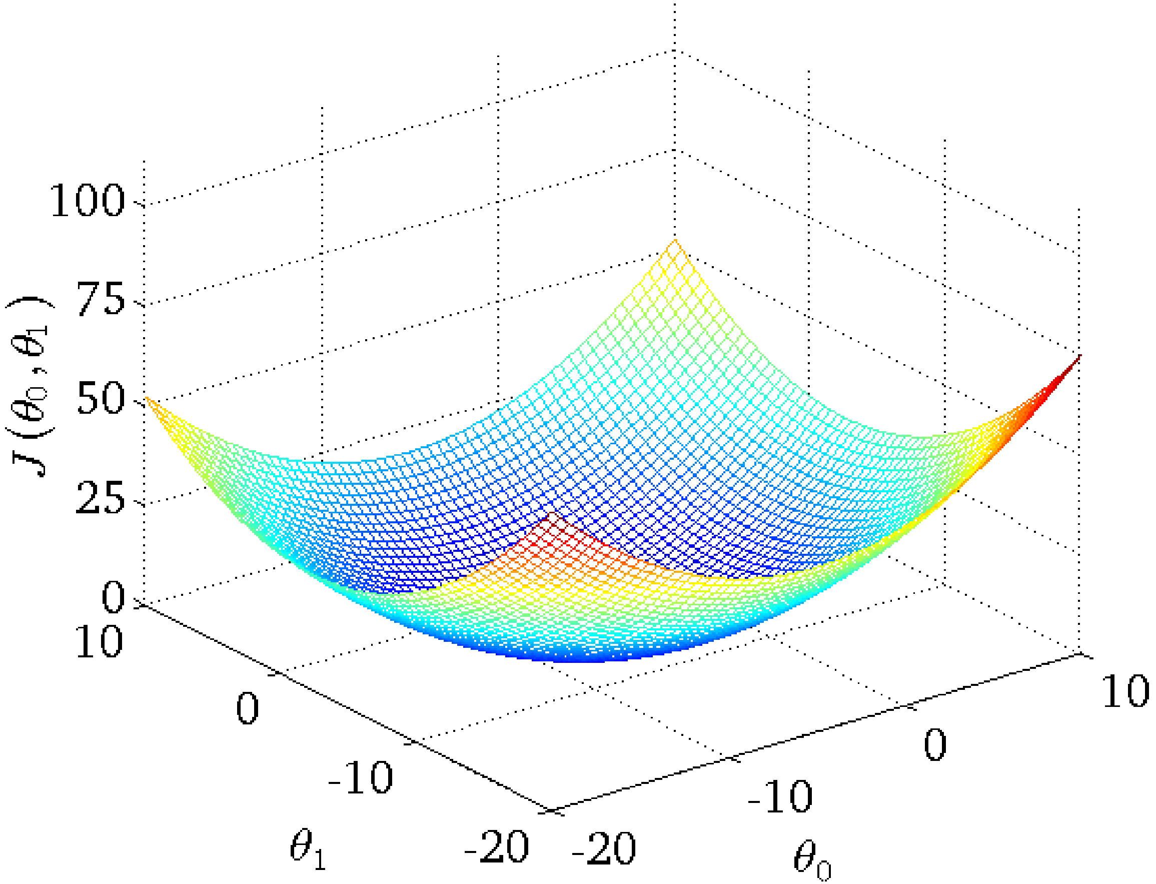
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

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

$\}$

update θ_0 and θ_1 simultaneously

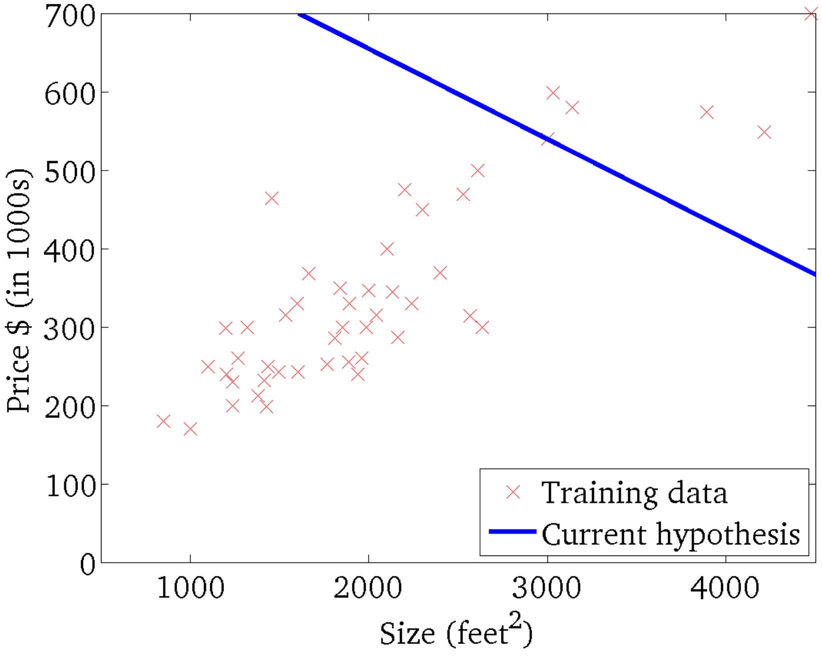

3
2
1
0
-1
-2
-3
J(θ0,θ1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1



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

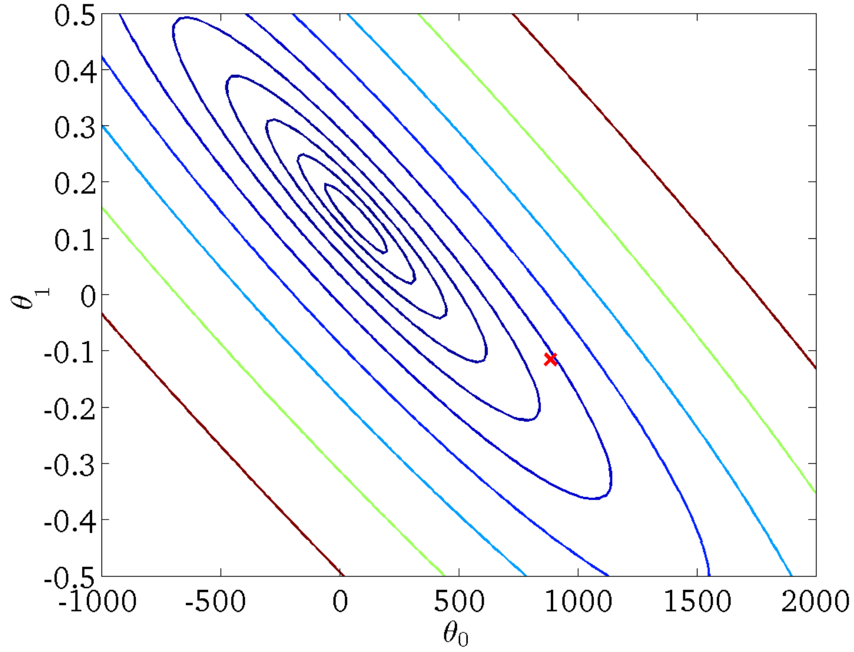
$h_\theta(x)$

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



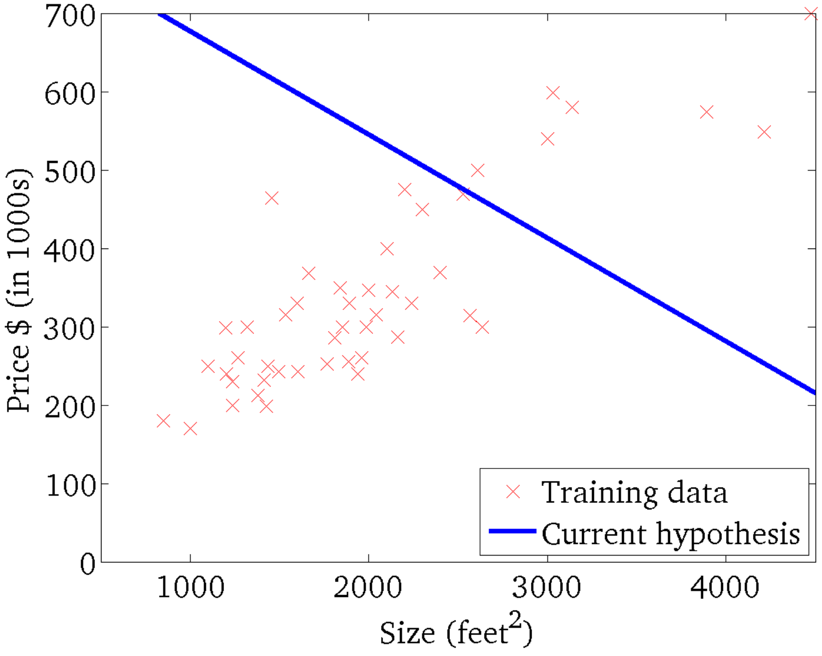
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



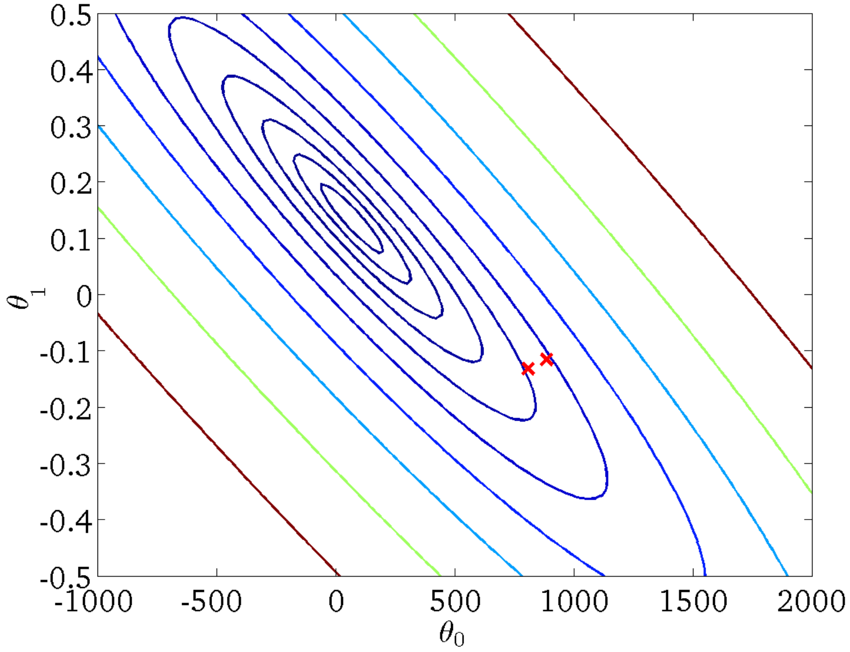
$$h_\theta(x)$$

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



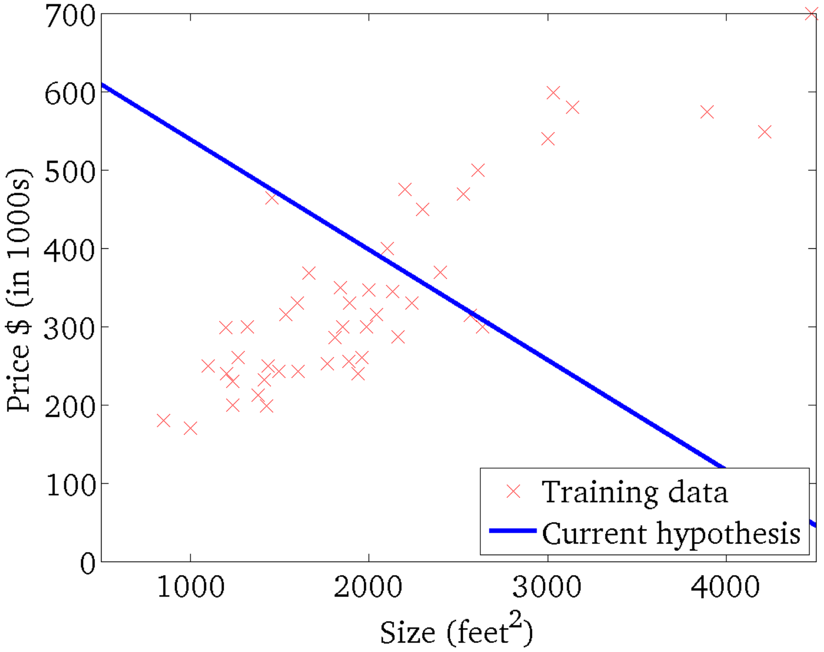
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



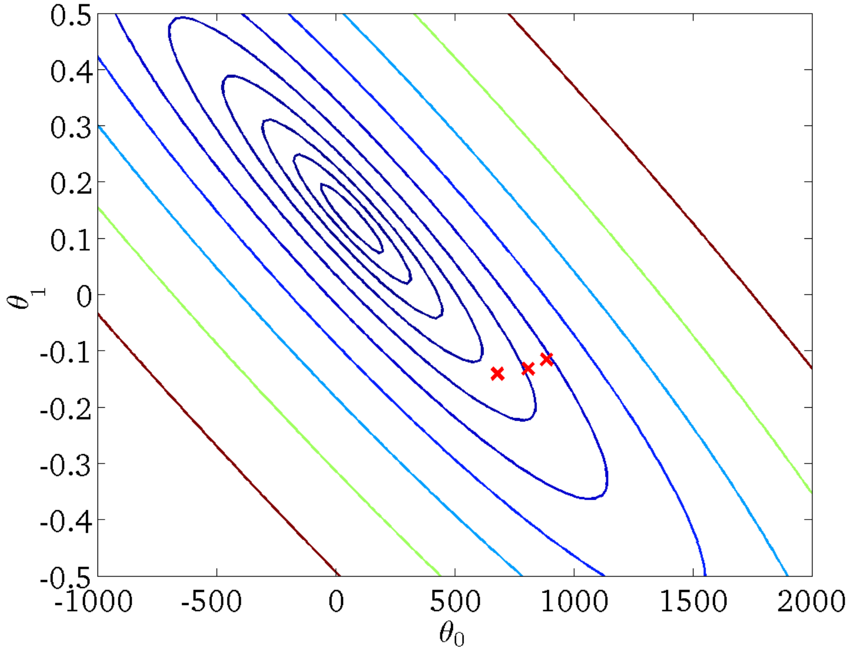
$$h_\theta(x)$$

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



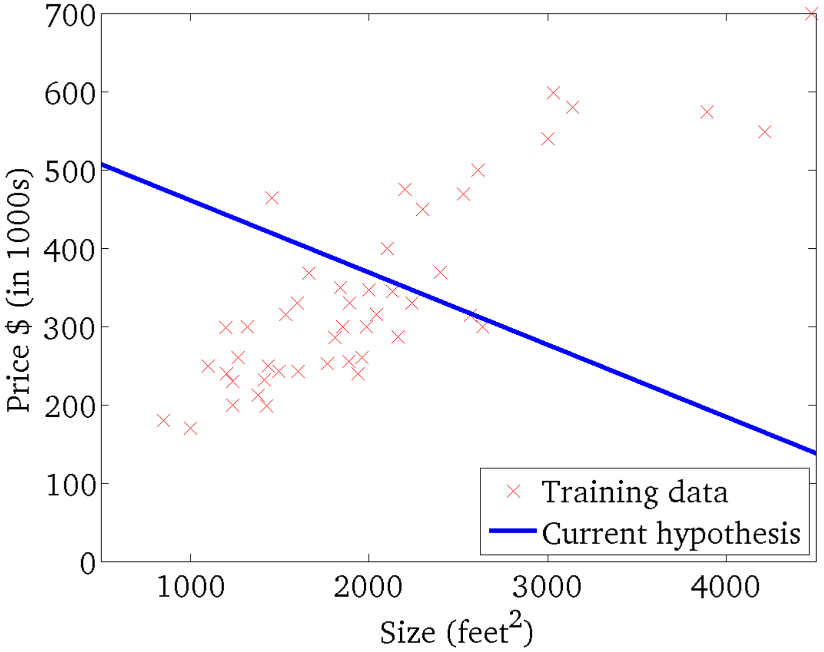
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



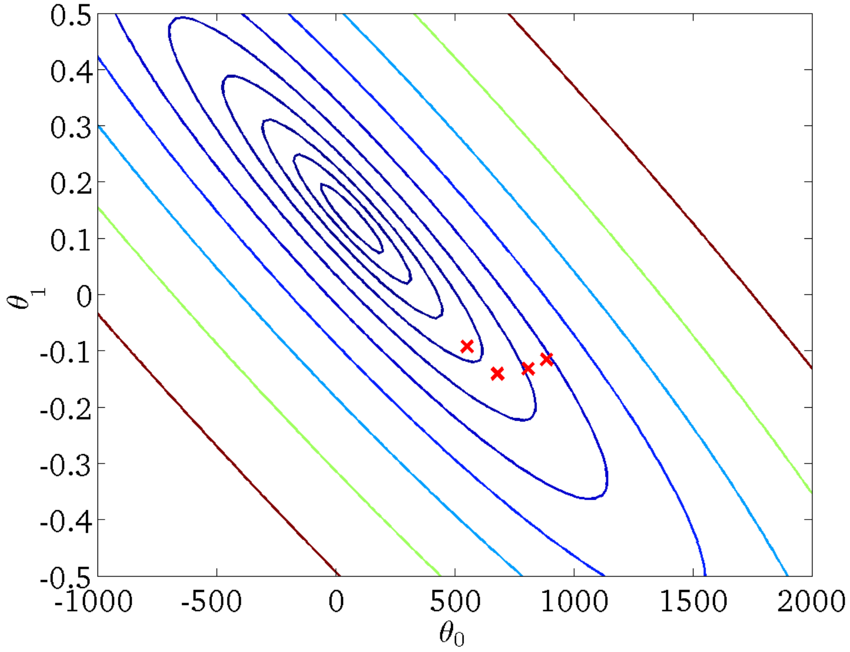
$h_\theta(x)$

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



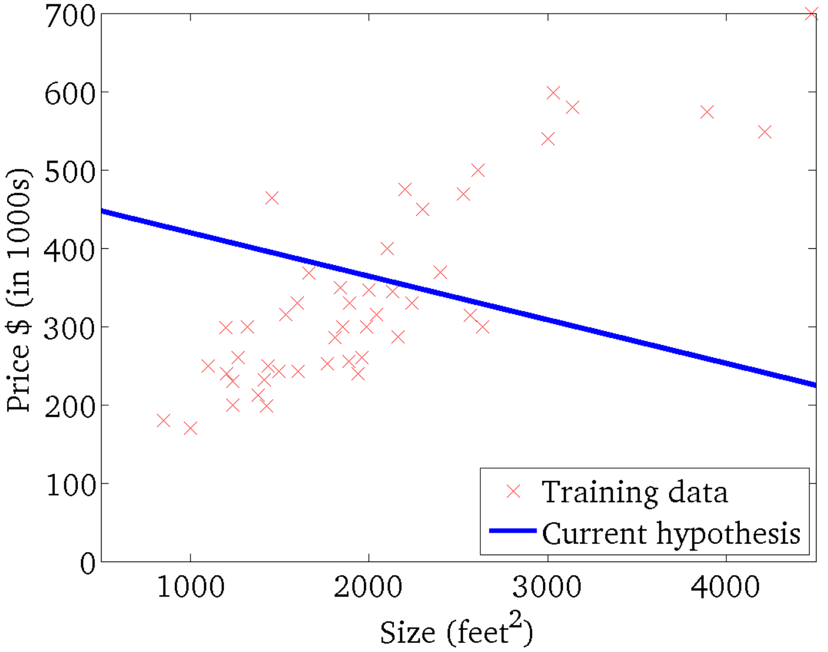
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



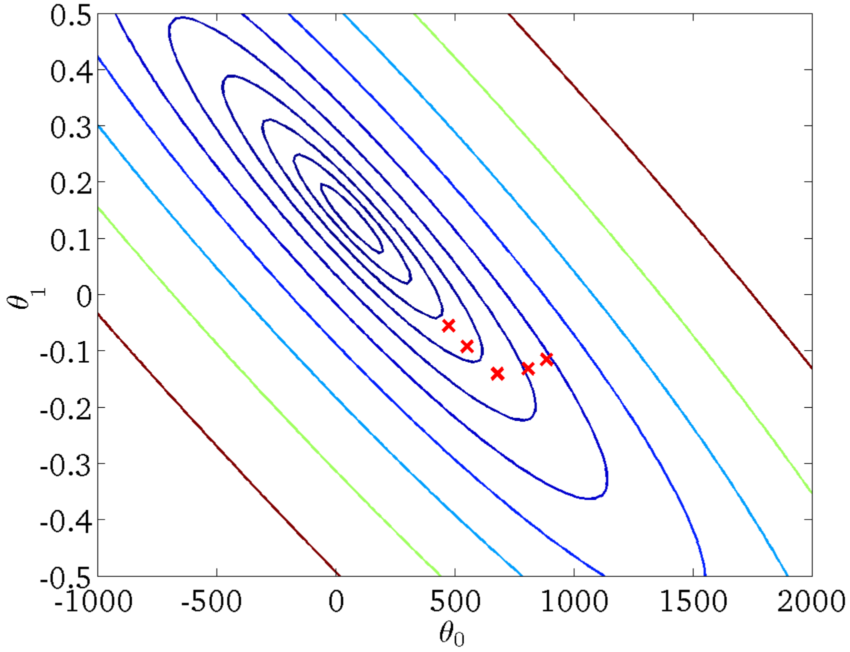
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

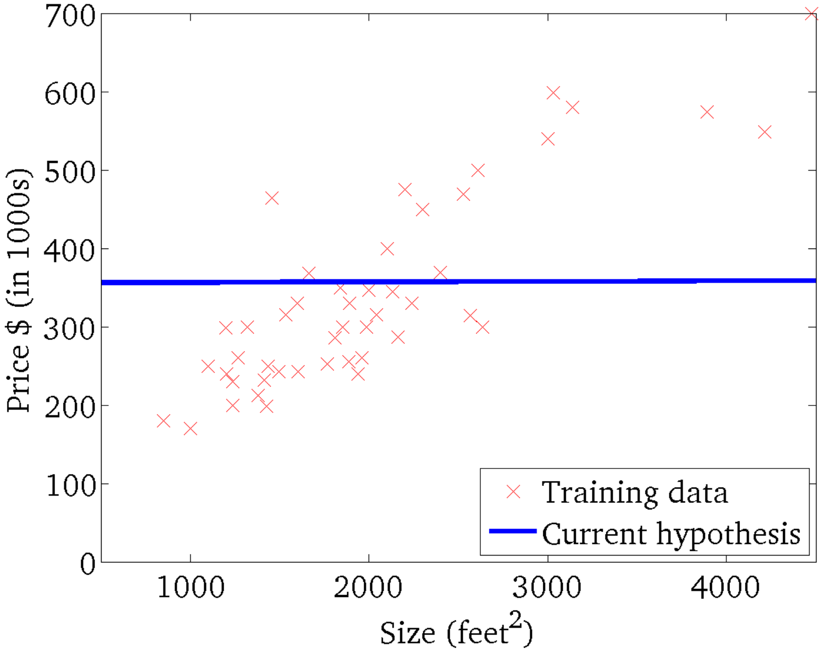
(function of the parameters θ_0, θ_1)



Andrew Ng

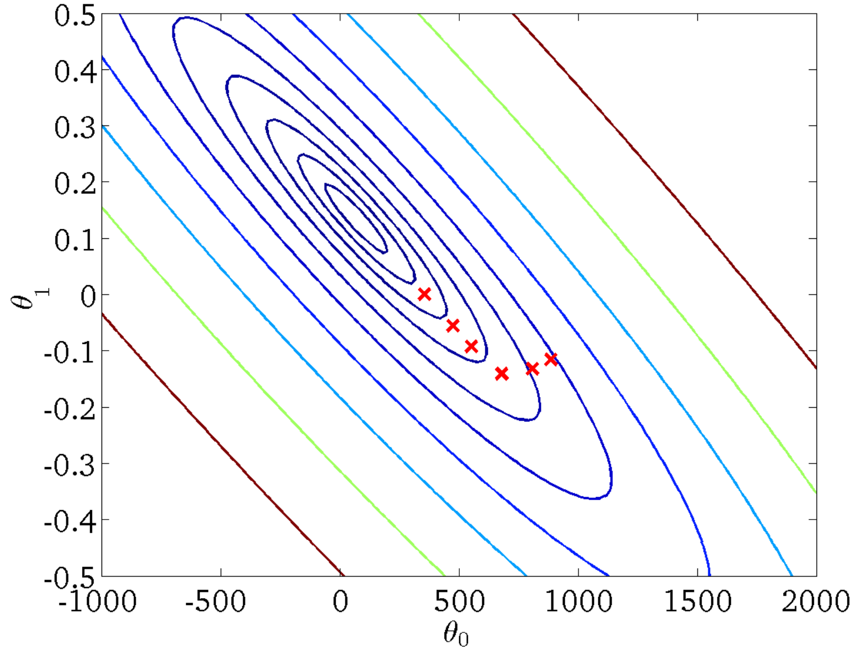
$h_\theta(x)$

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



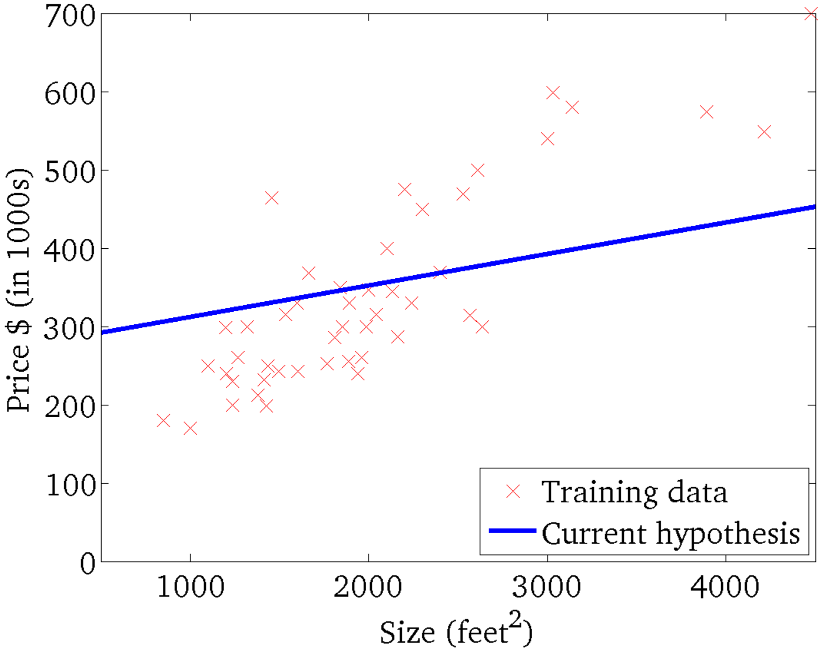
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



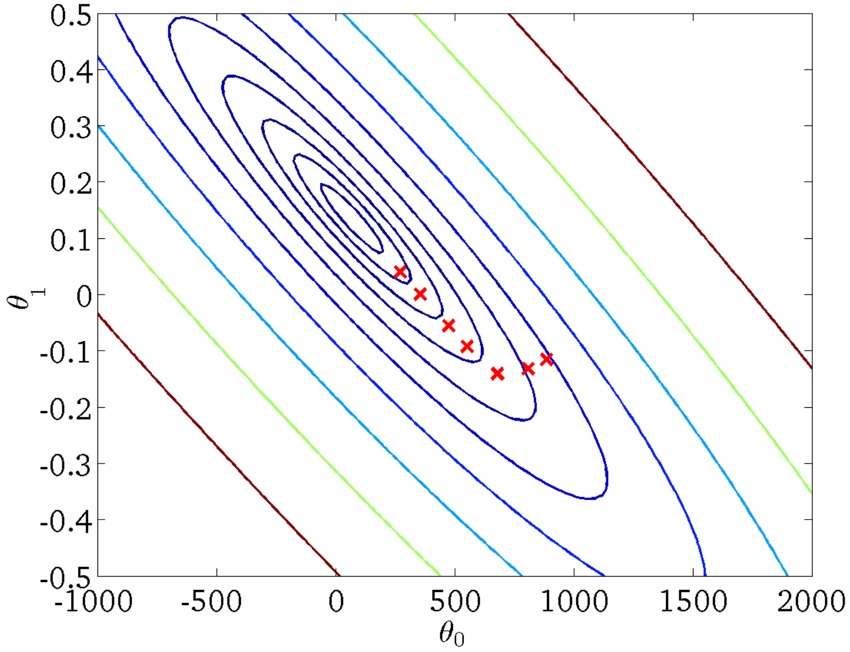
$$h_\theta(x)$$

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



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

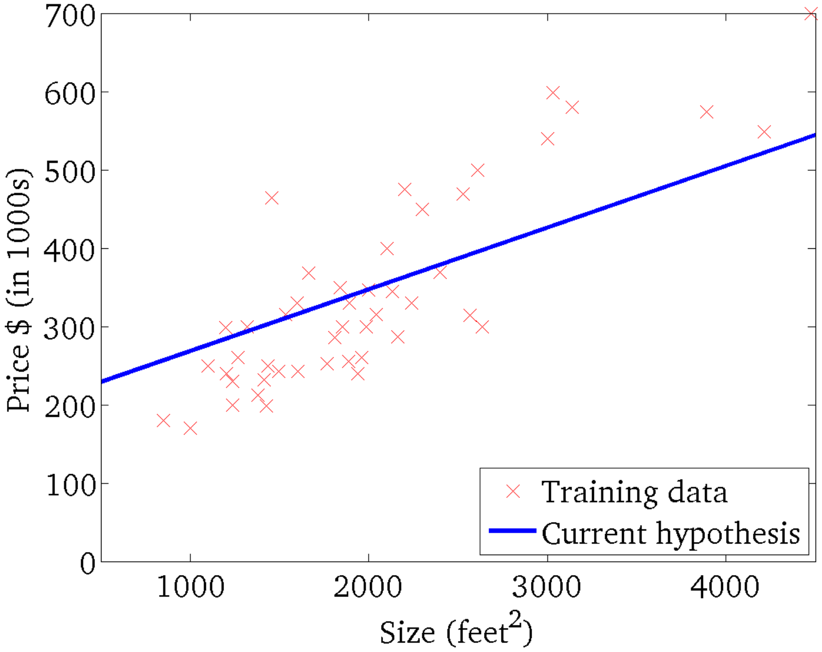
(function of the parameters θ_0, θ_1)



Andrew Ng

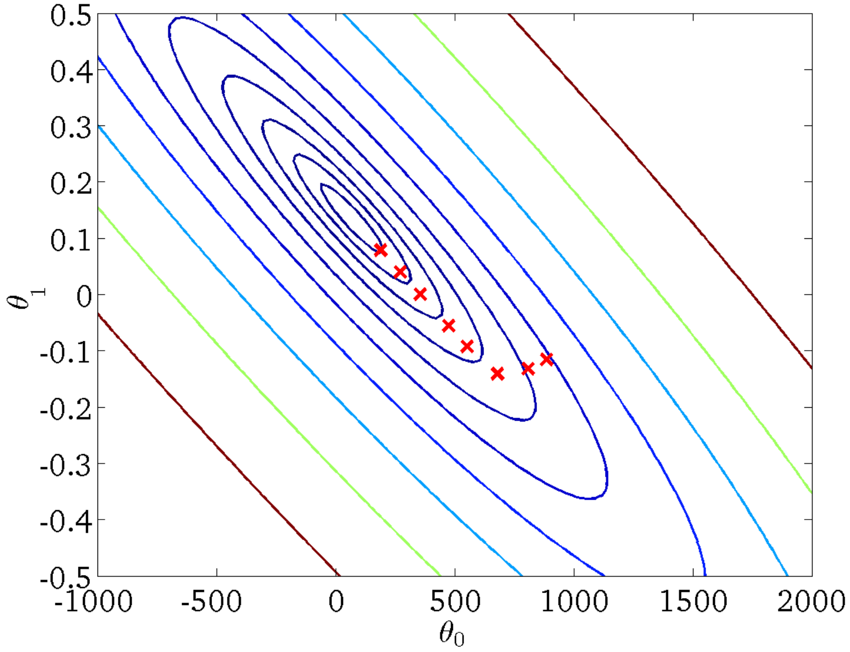
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

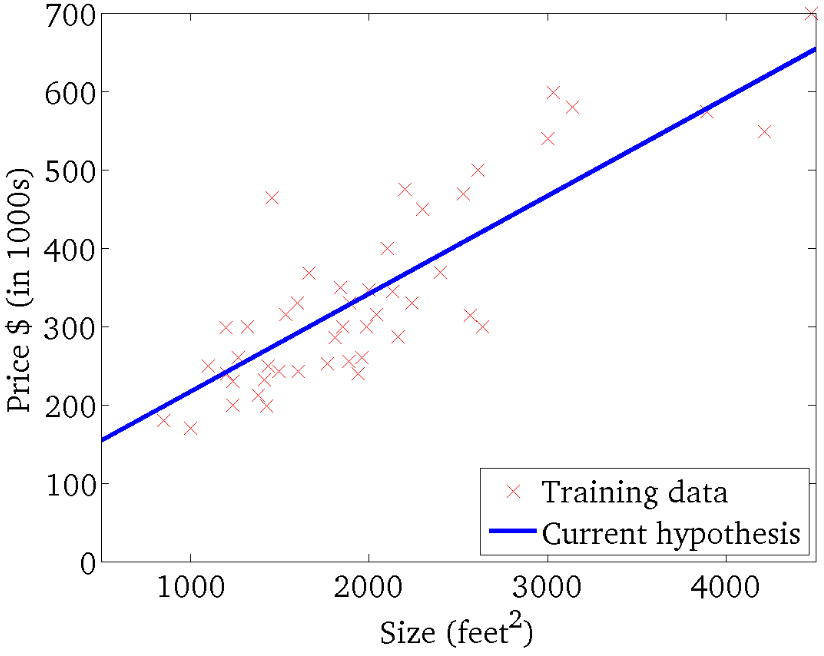
(function of the parameters θ_0, θ_1)



Andrew Ng

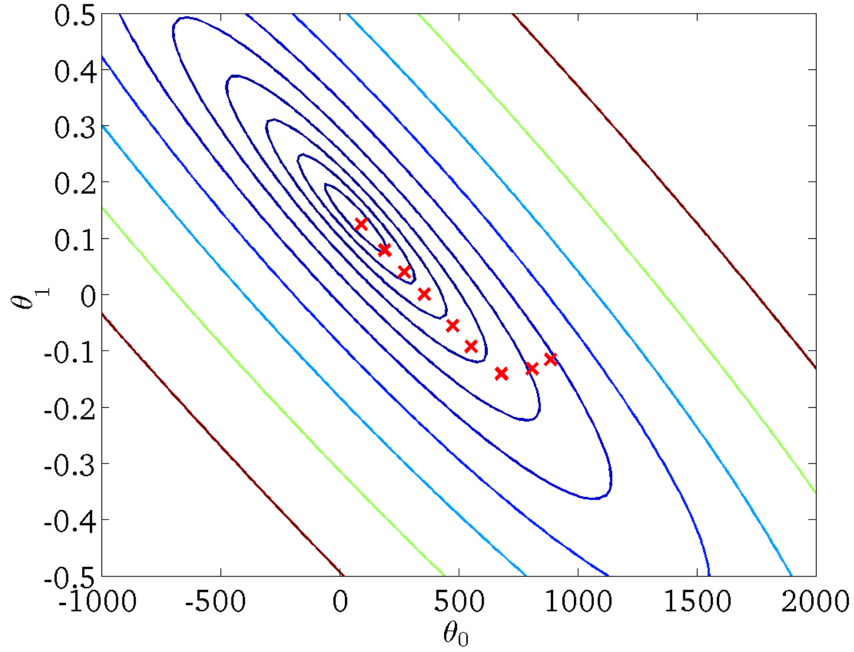
$h_\theta(x)$

(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)

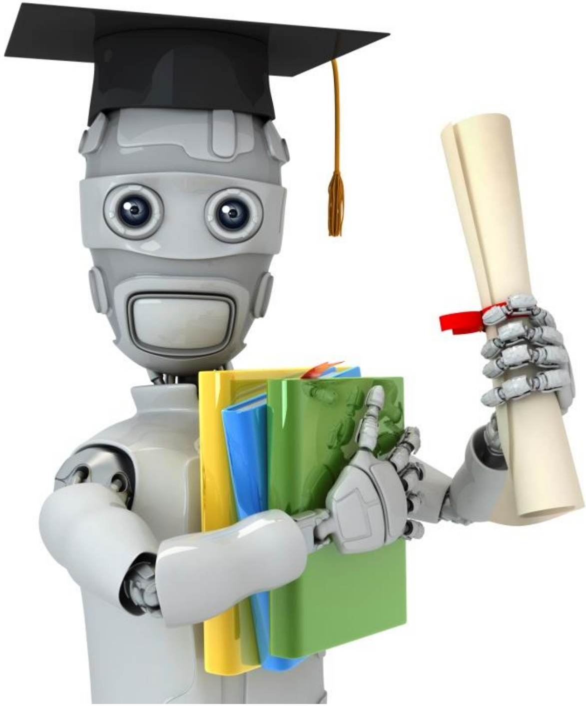


"Batch" Gradient Descent

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

"Batch" means $\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$

Questions?



Machine Learning

Linear Regression with multiple variables

Multiple features

Multiple features (variables).

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

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.

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

Hypothesis:

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

Multivariate Linear Model

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

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

Multivariate Linear Model

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

$$= \theta^T x$$

$$= \begin{bmatrix}\theta_0 \theta_1 \theta_2 \ldots \theta_n\end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad x_0 = 1$$

Cost function:

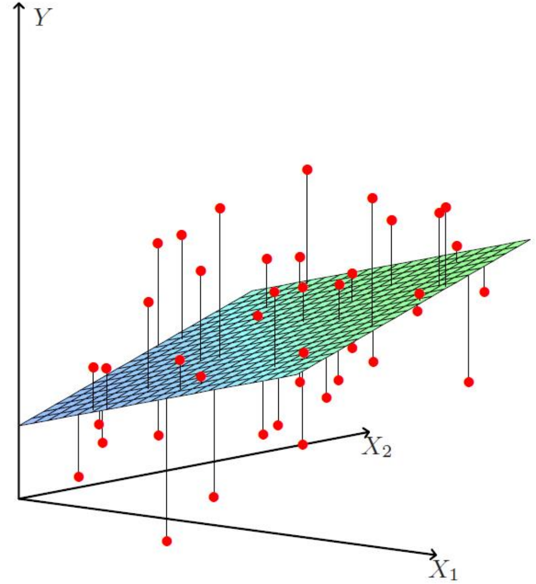
$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

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:

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



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 $\{$

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

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

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

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

$\}$

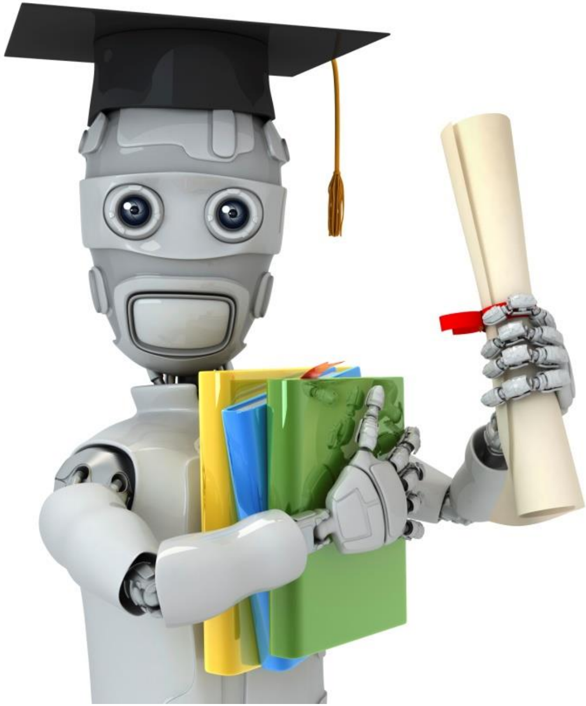
$x_0^{(i)} = 1$

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

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

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

$\ldots$



Machine Learning

Linear Regression with multiple variables

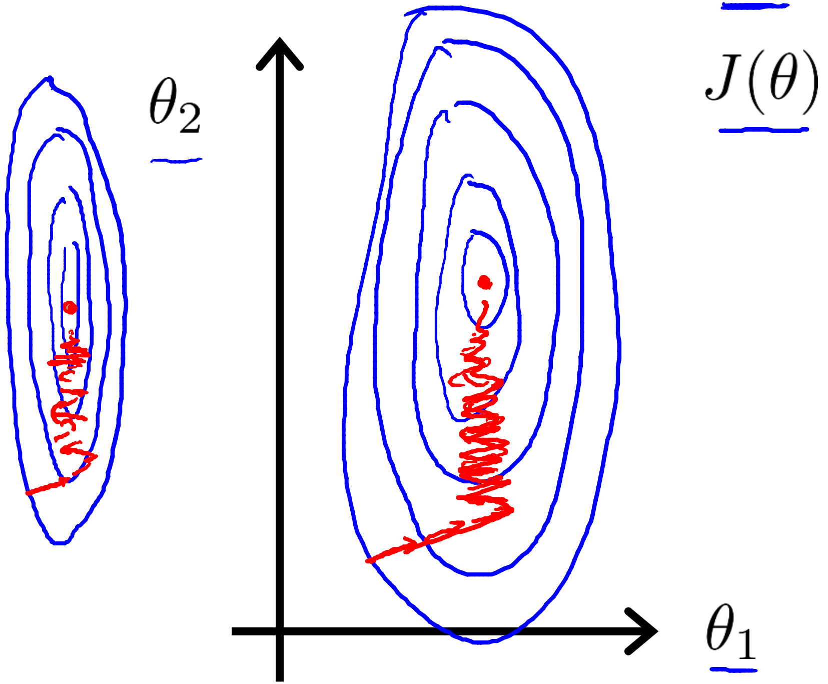
Gradient descent in practice I: Feature Scaling practice II: Learning Rate

Feature Scaling

Idea: Make sure features are on a similar scale. May be $-1 \le x_i \le 1$

E.g. x_1 = size (0-2000 feet^2)

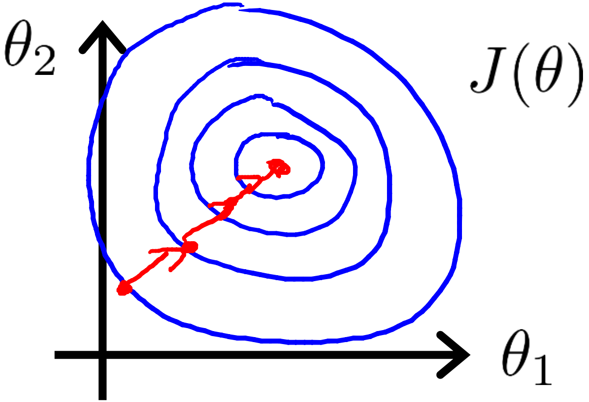
x_2 = number of bedrooms (1-5)



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

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

$0 \le x_1 \le 1 \qquad 0 \le x_2 \le 1$



Andrew Ng

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. $$x_1 = \frac{size - 1000}{2000}$$

$$x_2 = \frac{\#bedrooms - 2}{5}$$

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

$$x_i^{(j)} = \frac{x_i^{(j)} - \mu_i}{S_i}$$

S_i ranges (max - min) or standard deviation

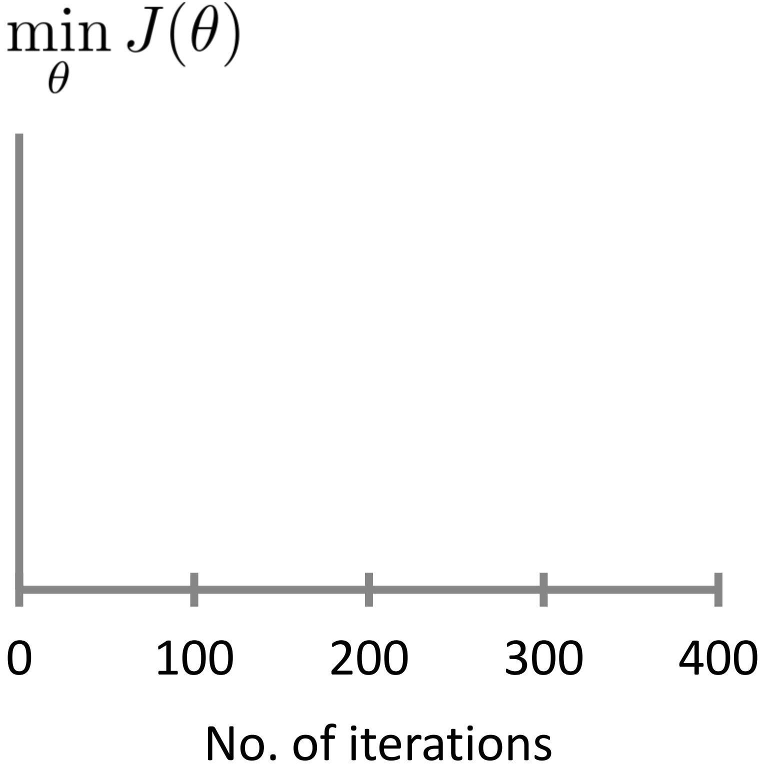
Learning rate of gradient descent

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

- “Debugging”: How to make sure gradient descent is working correctly.

- How to choose learning rate α.

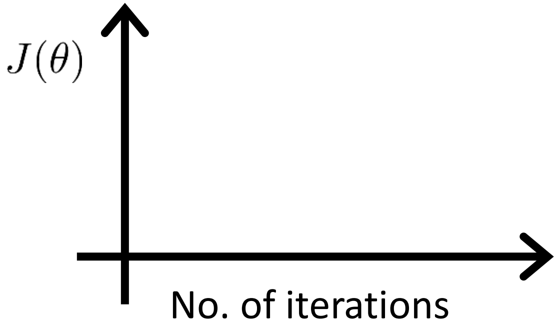
Making sure gradient descent is working correctly.



Example automatic convergence test:

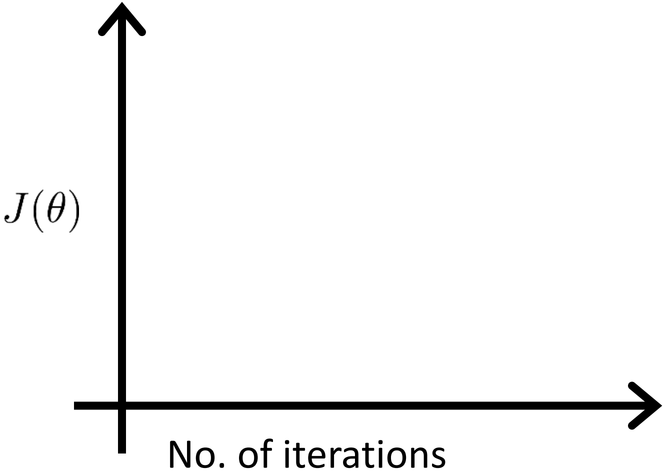
Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

Making sure gradient descent is working correctly.



Gradient descent not working.

Use smaller α.



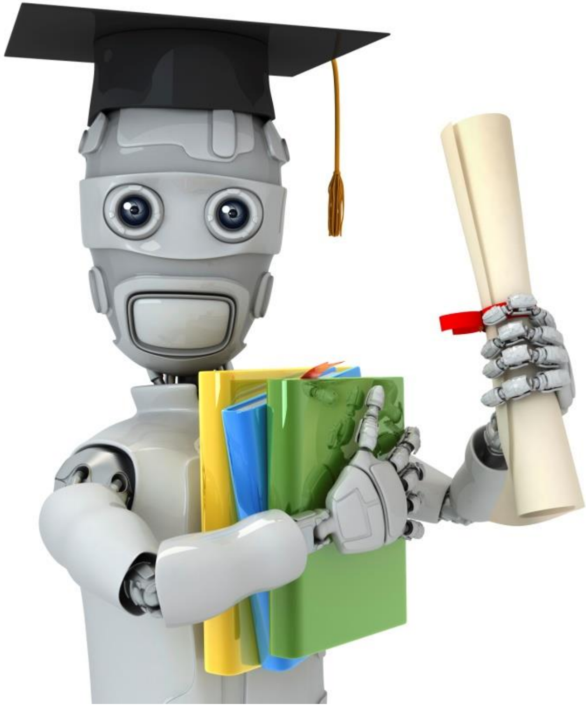
- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

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

$$\dots, 0.001, \qquad , 0.01, \qquad , 0.1, \qquad , 1, \dots$$



Machine Learning

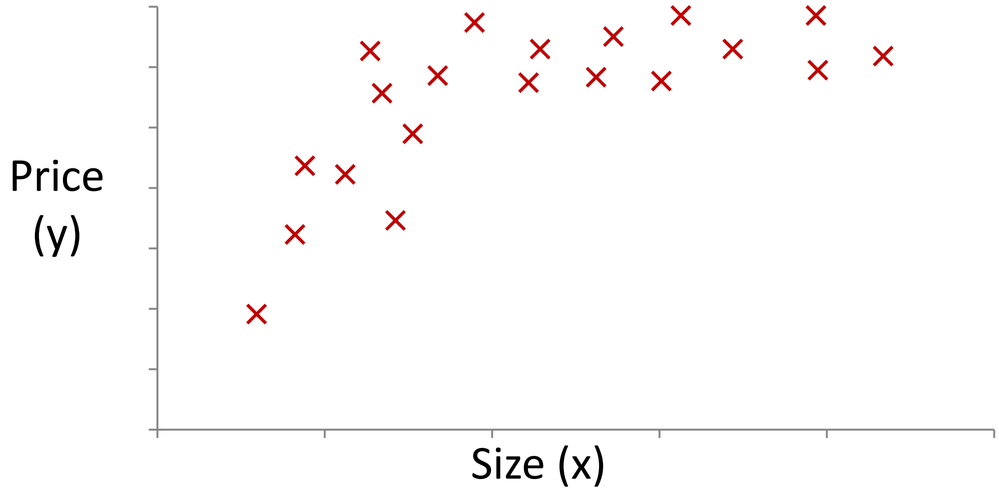
Linear Regression with multiple variables

Features and polynomial regression

Housing prices prediction

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

Polynomial regression



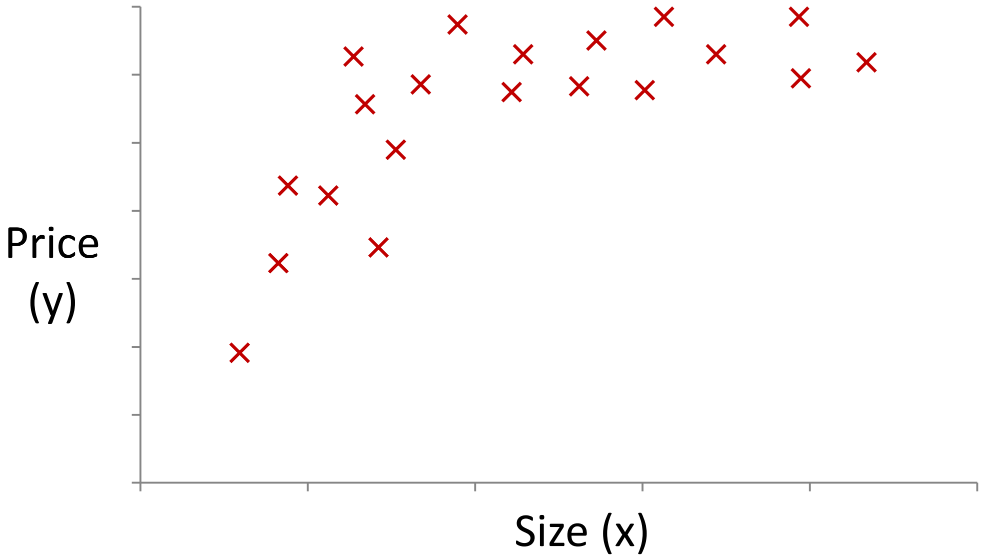
$$\theta_0 + \theta_1 x + \theta_2 x^2$$

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

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3 \end{aligned}$$

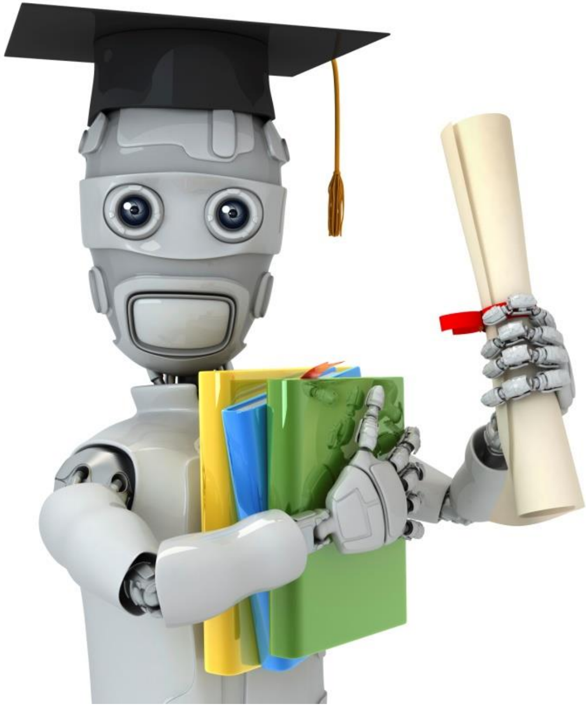
$$x_1 = (size)$$
$$x_2 = (size)^2$$
$$x_3 = (size)^3$$

Choice of features



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

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

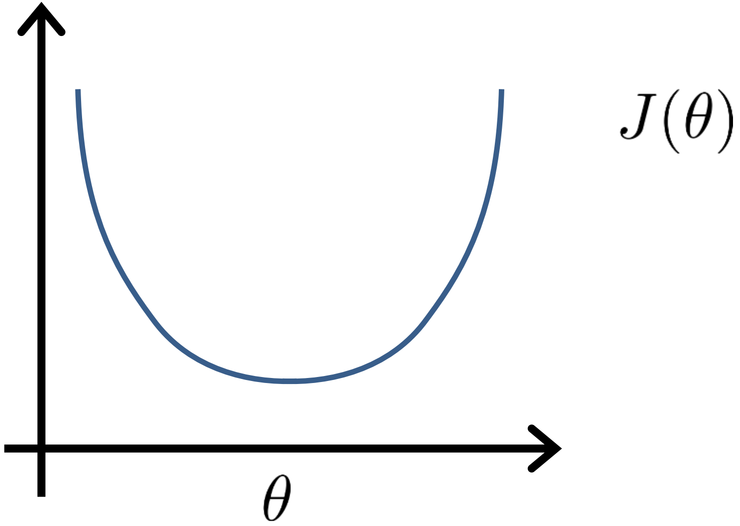


Machine Learning

Linear Regression with multiple variables

Normal equation

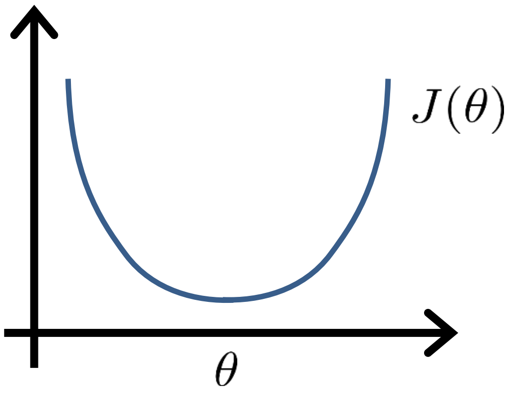
Gradient Descent



Normal equation: Method to solve for θ analytically.

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

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



$\theta \in \mathbb{R}^{n+1}$ $\qquad J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots = 0 \qquad \text{(for every } j\text{)}$$

Solve for $\theta_0, \theta_1, \ldots, \theta_n$

$$\theta \in \mathbb{R}^{n+1} \quad J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

(for every j)

$$= \frac{1}{2m} \sum_{i=1}^{m} (\theta_j x_j^{(i)} - y_j^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} (\theta_j^2 (x_j^{(i)})^2 - 2\theta_j x_j^{(i)} + (y_j^{(i)})^2)$$

$$\frac{\partial J(\theta_j)}{\partial \theta_j} = \frac{1}{2m} \sum_{i=1}^{m} (2\theta_j (x_j^{(i)})^2 - 2x_j^{(i)} y_j^{(i)})$$

$$= \frac{1}{m} \sum_{i=1}^{m} (\theta_j (x_j^{(i)})^2 - x_j^{(i)} y_j^{(i)})$$

$$\frac{\partial J(\theta_j)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (\theta_j (x_j^{(i)})^2 - x_j^{(i)} y_j^{(i)}) = 0$$

$$\sum_{i=1}^{m} \theta_j x_j^{(i)} x_j^{(i)} = \sum_{i=1}^{m} x_j^{(i)} y_j^{(i)}$$

(for every j) matrix form can be used

$$X^T X \theta = X^T y$$

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

Examples: $m = 4$.

x_0	Size (feet^2) 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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

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

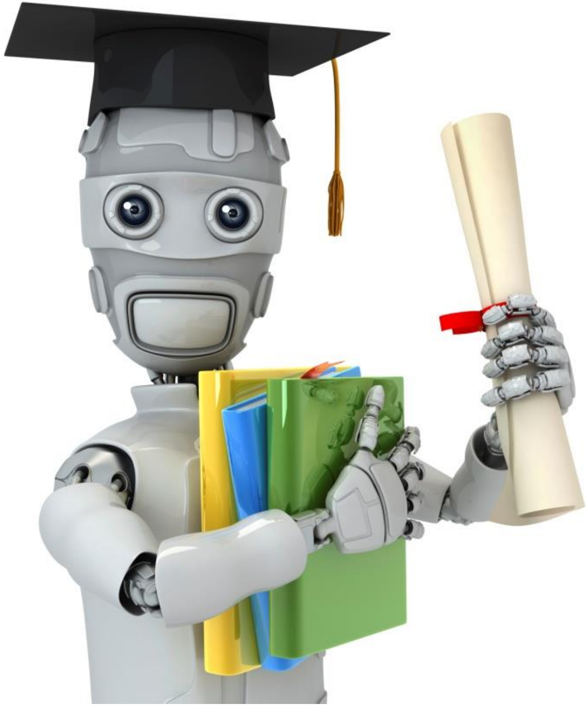
m **examples** $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ **;** n **features.**

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

E.g. If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

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

Gradient Descent	Normal Equation
• Need to choose α. • Needs many iterations. • Works well even when n is large.	• No need to choose α. • Don't need to iterate. • Need to compute $(X^TX)^{-1}$ • Slow if n is very large.



Machine Learning

Linear Regression with multiple variables

Normal equation and non-invertibility (optional)

Normal equation

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

- What if $X^T X$ is non-invertible? (singular/ degenerate)
- MATLAB: `inv(X'*X)*X'*y`

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

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