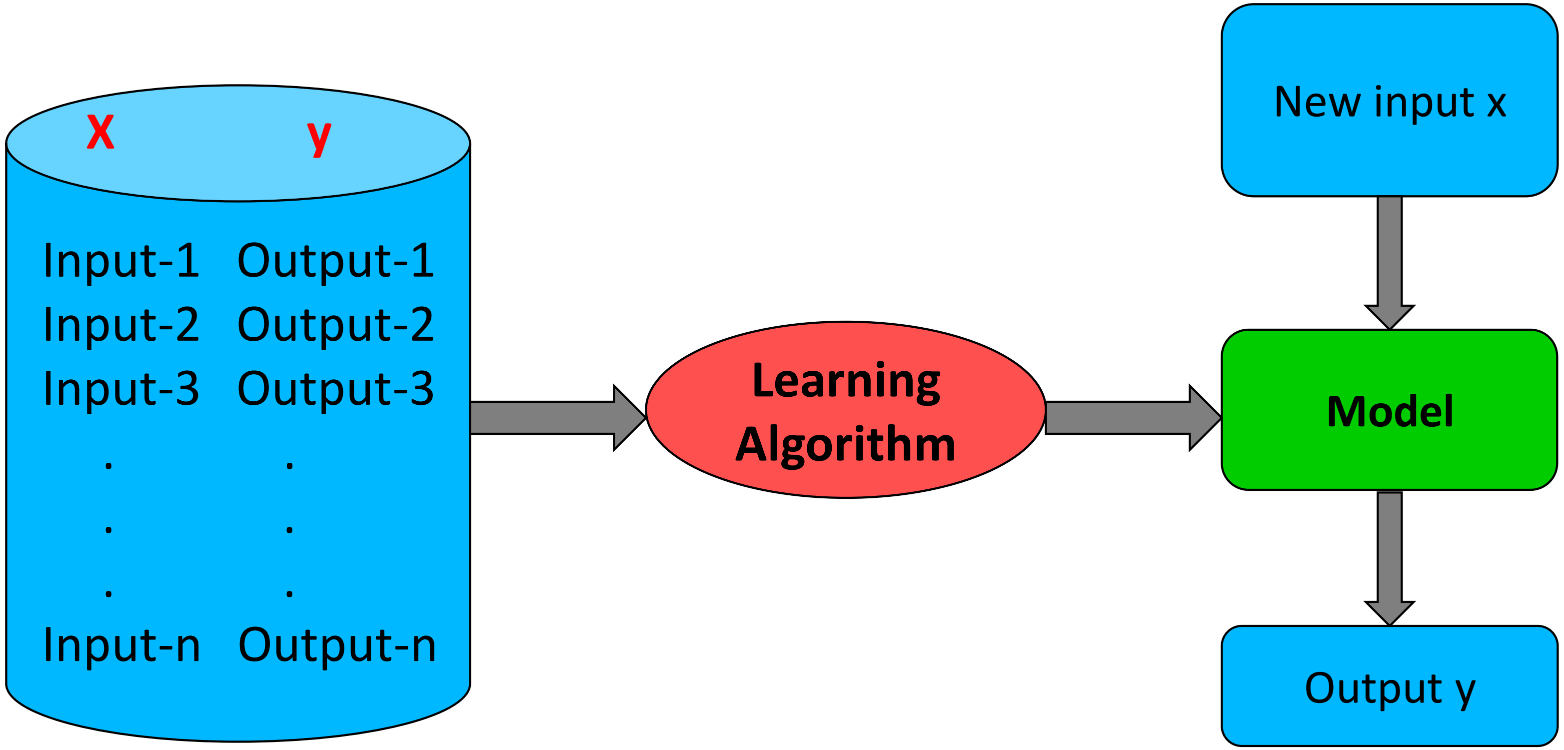
Supervised Learning

Regression



Classification

Supervised Learning

- A model defined with a set of parameters.

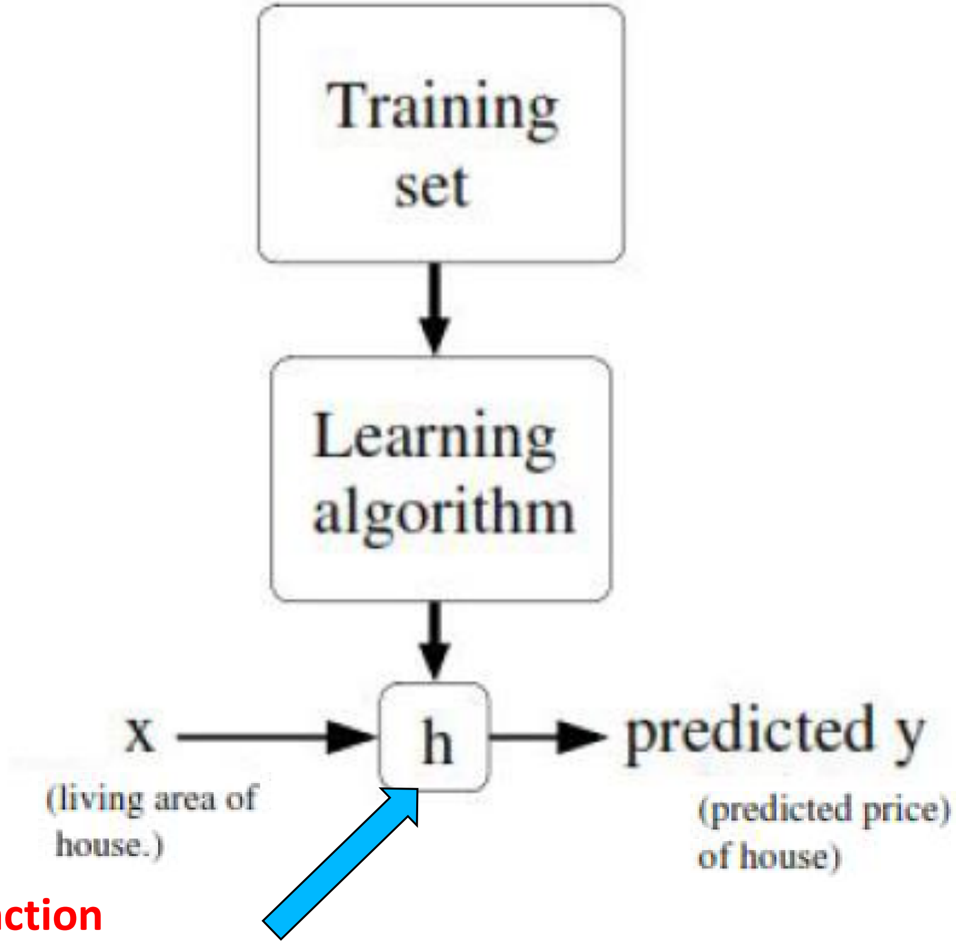
$$y = h(x|\theta)$$

Regression function **OR** Discriminant function

- Where $h(\cdot)$ is the model and θ are its parameters
- Regression : y – number
- Classification: y – class code (e.g. 0/1)

Regression

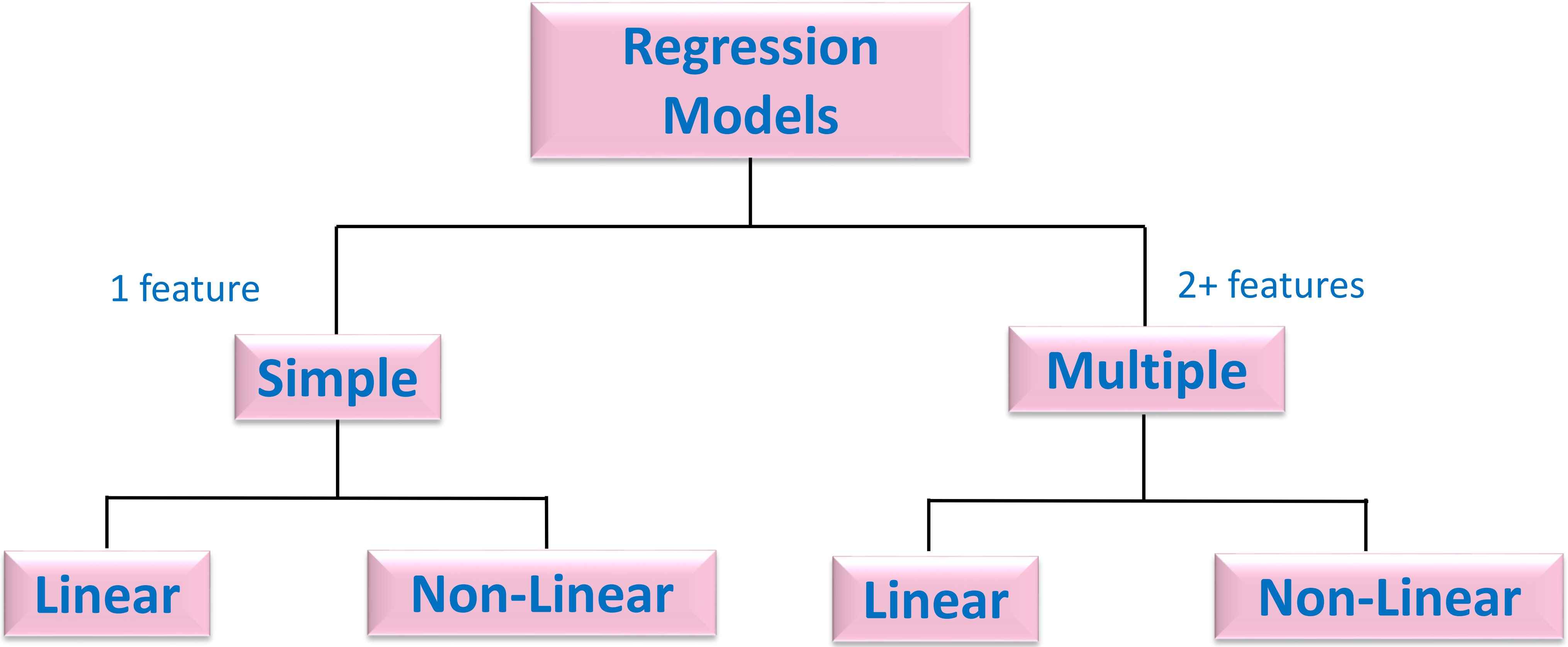
- Supervised Learning tasks where the datasets that are used for predictive / statistical modeling contain continuous labels.



Hypothesis function

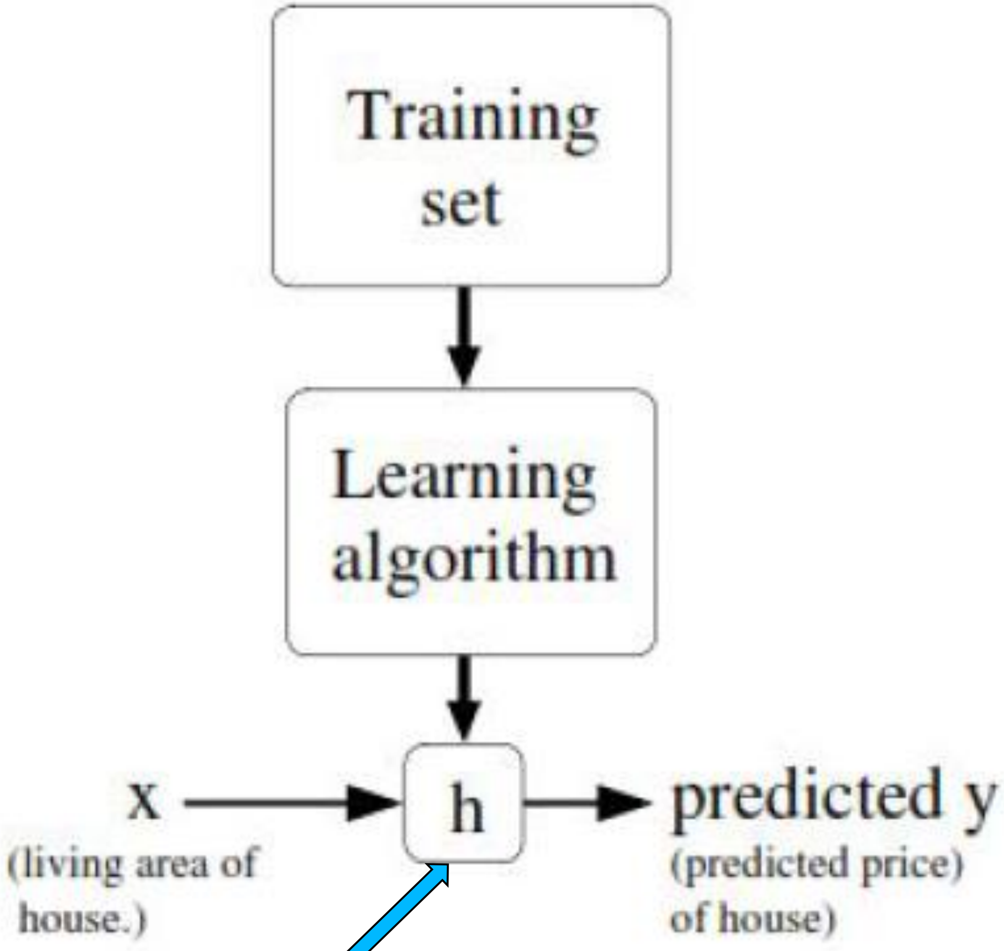
Learn a function *h*(*x*), so that *h*(*x*) is a good predictor for the corresponding value of *y*

Types of Regression Models



Linear Regression

- Find a linear equation for a continuous response variable known as *Y* which will be a function of one or more variables (*X*)



Hypothesis function
→ LINEAR

Linear Regression

- Given an input *x*, compute an output *y*

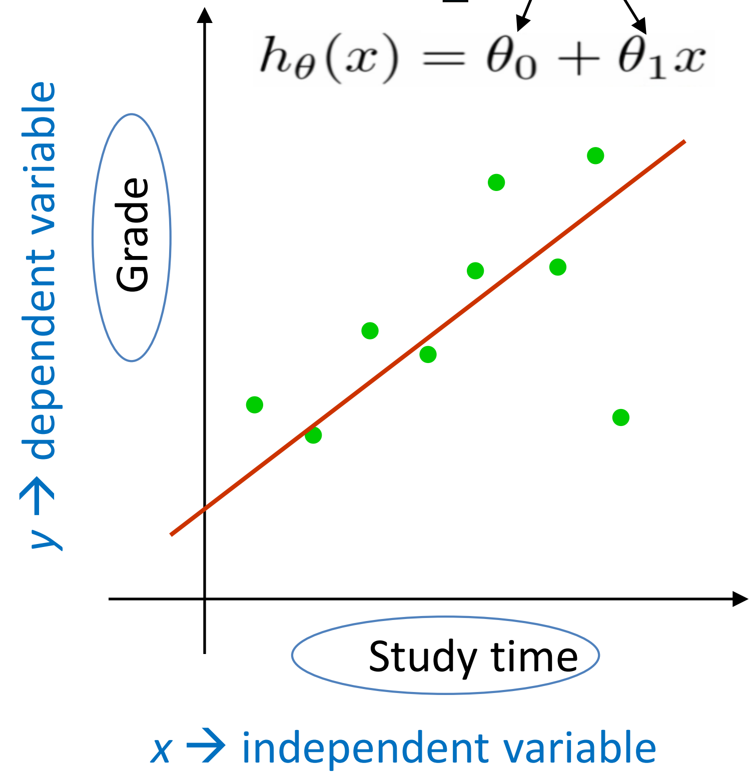
- For example –
  - Predict height from age
  - Predict house price from house area
  - Predict grade from study time

Linear regression with one variable / feature
Univariate Linear Regression

Slope Intercept

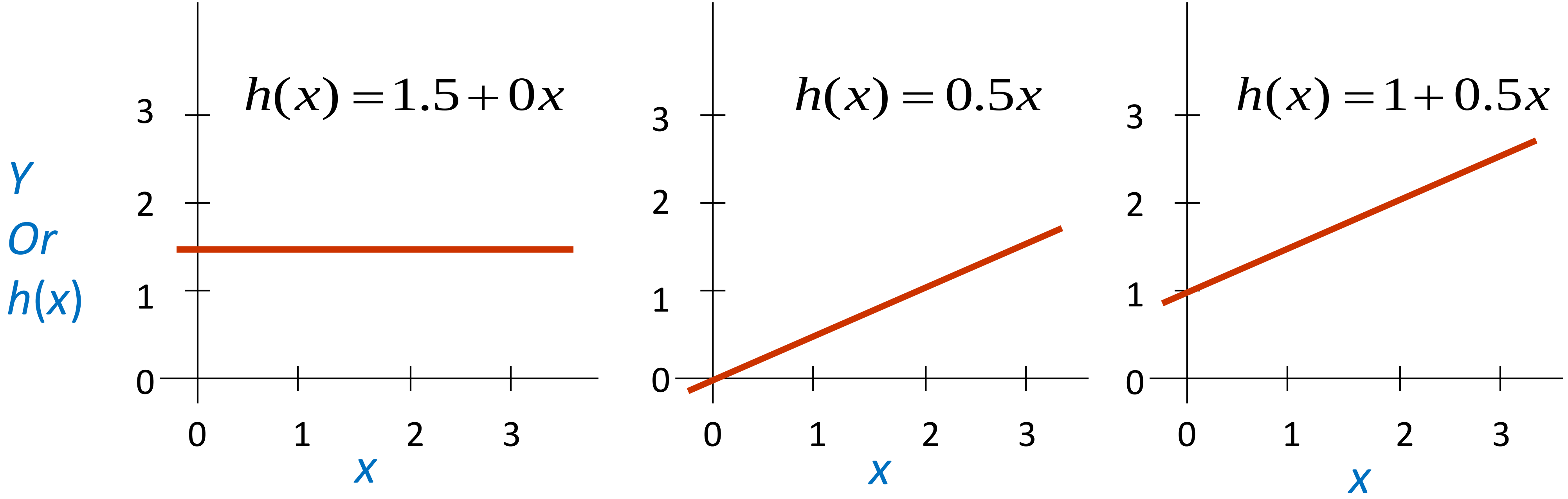
$$y = mx + c$$

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



$h_\theta(x)$ – criterion variable
x – predictor variable
θ_i's - parameters
θ_0 – constant
θ_1 – regression coefficient

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



$\theta_0 = 1.5$

$\theta_1 = 0$

$\theta_0 = 0$

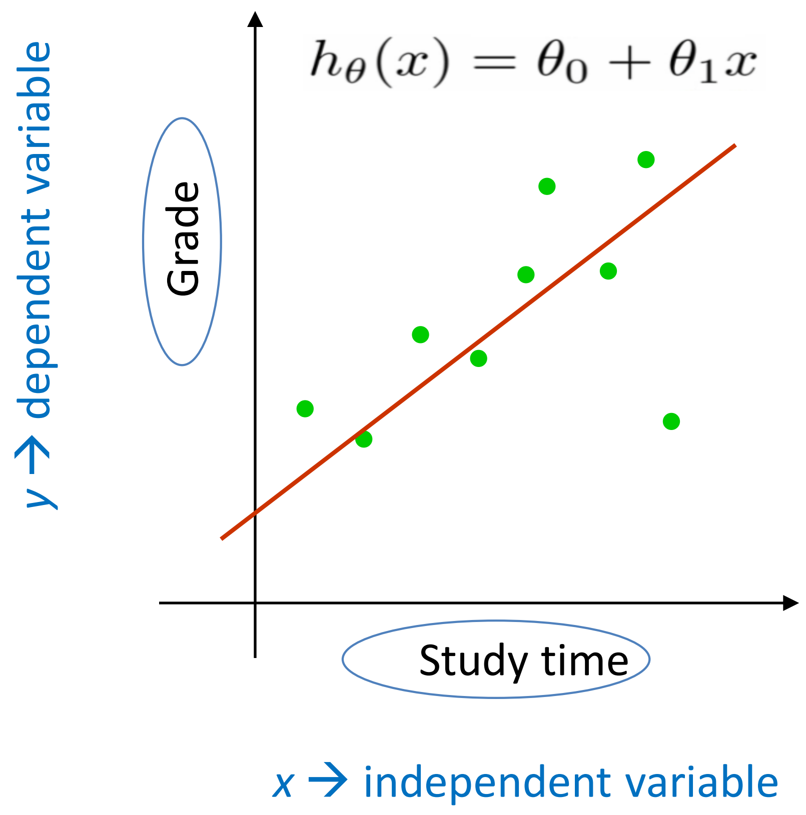
$\theta_1 = 0.5$

$\theta_0 = 1$

$\theta_1 = 0.5$

Linear Regression

- Idea: Choose Θ_0 , Θ_1 so that $h_\Theta(x)$ is close to y for our training example (X, y)



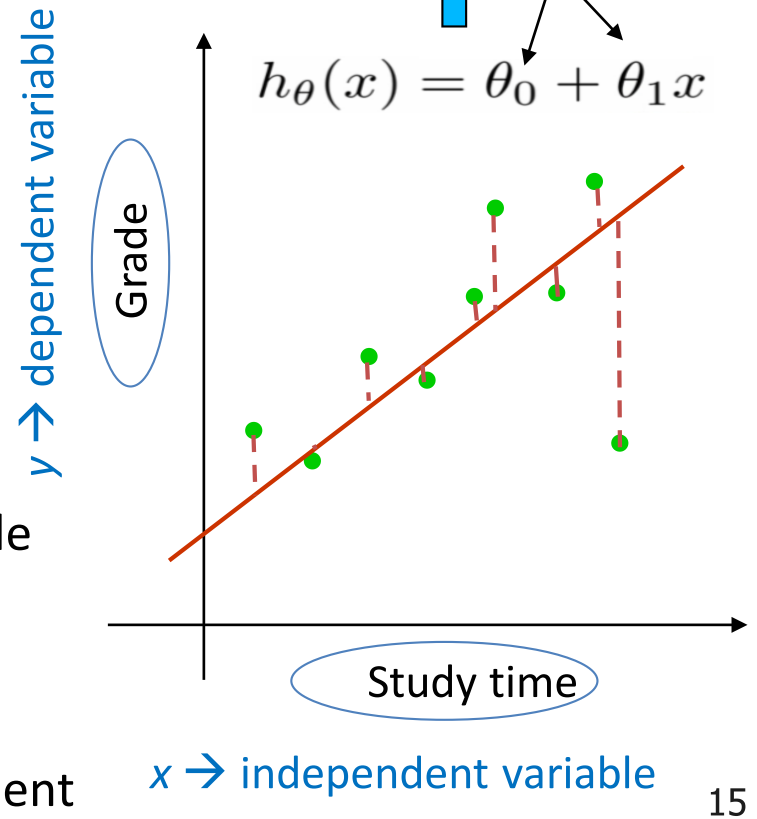
Linear Regression Line

- The least-squares regression line is the unique line such that the sum of squared vertical (y) distances between the data points and the line is the smallest possible.

Slope Intercept

$$y = mx + c$$

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



$h_\theta(x)$ – criterion variable
x – predictor variable
Θ_i's - parameters
Θ_0 – constant
Θ_1 – regression coefficient

Linear Regression Line

- How to compute an error ?
- For given $x^{(i)}$, actual output is $y^{(i)}$ and using regression line (for some Θ_0 and Θ_1), it is $\left(\theta_0 + \theta_1 x^{(i)}\right)$
- Error for $x^{(i)}$

= (predicted $y^{(i)}$ – actual $y^{(i)}$)

= $\left(\left(\theta_0 + \theta_1 x^{(i)}\right) - y^{(i)}\right)$

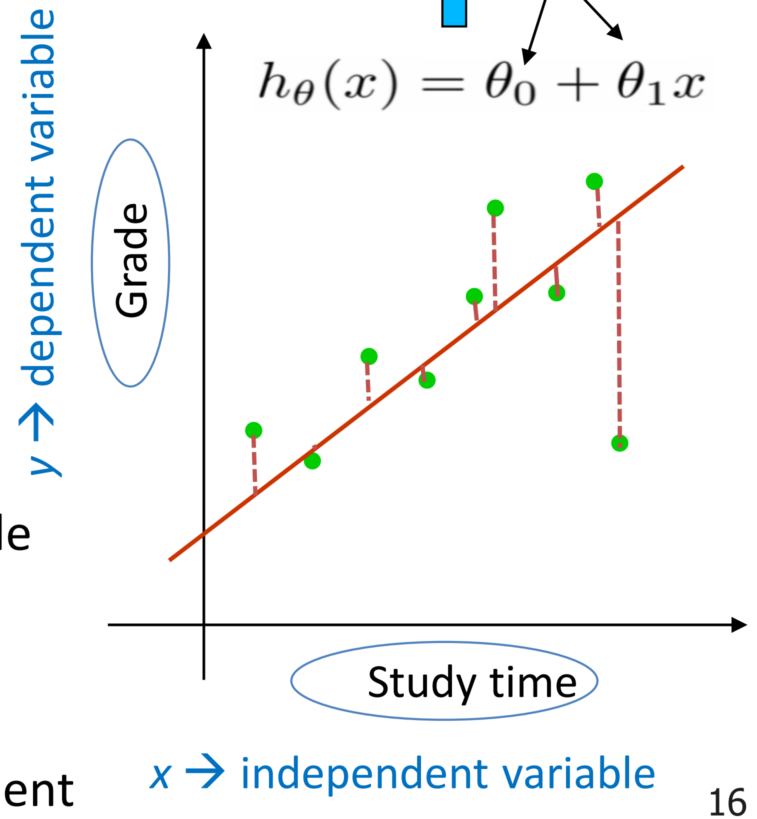
= $\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$

Slope

Intercept

$$y = mx + c$$

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



$h_\Theta(x)$ – criterion variable
x – predictor variable
Θ_i's - parameters
Θ_0 – constant
Θ_1 – regression coefficient

Linear Regression Line

- Sum of squared error

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

Objective Function / Cost Function

Where m = Number of data-points (size of training set)

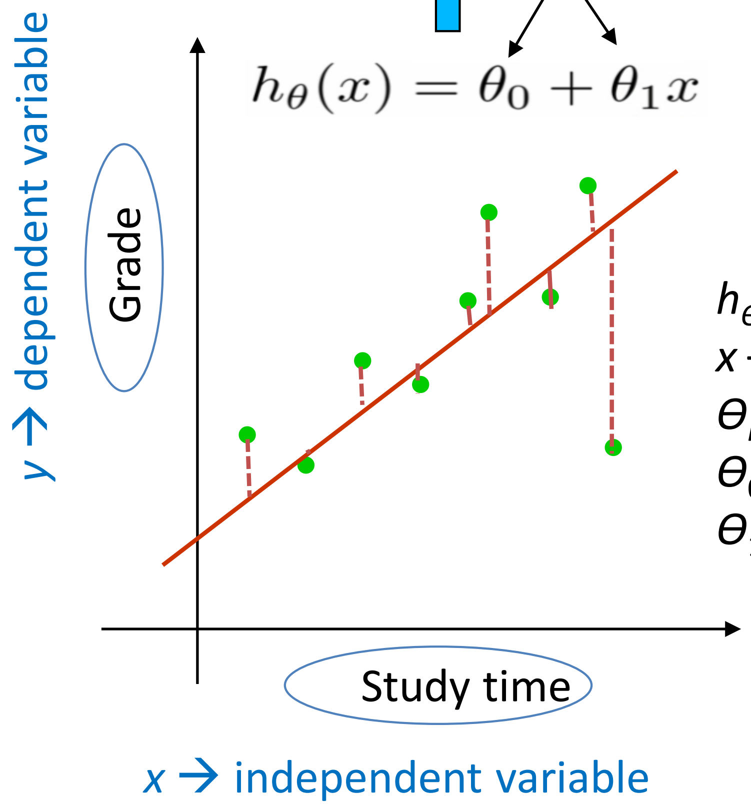
- Goal : Find Θ's such that Error is minimum (Find parameters Θ_0 and Θ_1 to minimize objective function)

➔ Learning

Slope Intercept

$$y = mx + c$$

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



$h_\theta(x)$ – criterion variable
x – predictor variable
Θ_i's - parameters
Θ_0 – constant
Θ_1 – regression coefficient

Exercise

- Consider applying linear regression with the hypothesis as $h_\theta(x) = \theta_0 + \theta_1 x$
- The training data is given as

X	Y
6	7
5	4
10	8
3	4

- The cost function is

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

- What is value of $J(\theta)$, if $\theta = (2,1)$?

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

- Parameters $$\theta_0 \text{ and } \theta_1$$

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

- Goal $$\min_{\theta_0, \theta_1} imize \quad J(\theta_0, \theta_1)$$

- Simplified Illustration with $\theta_0 = 0$

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

$$\Downarrow$$

$$\boxed{h_\theta(x) = \theta_1 x}$$

$$\Downarrow$$

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

$$\Downarrow$$

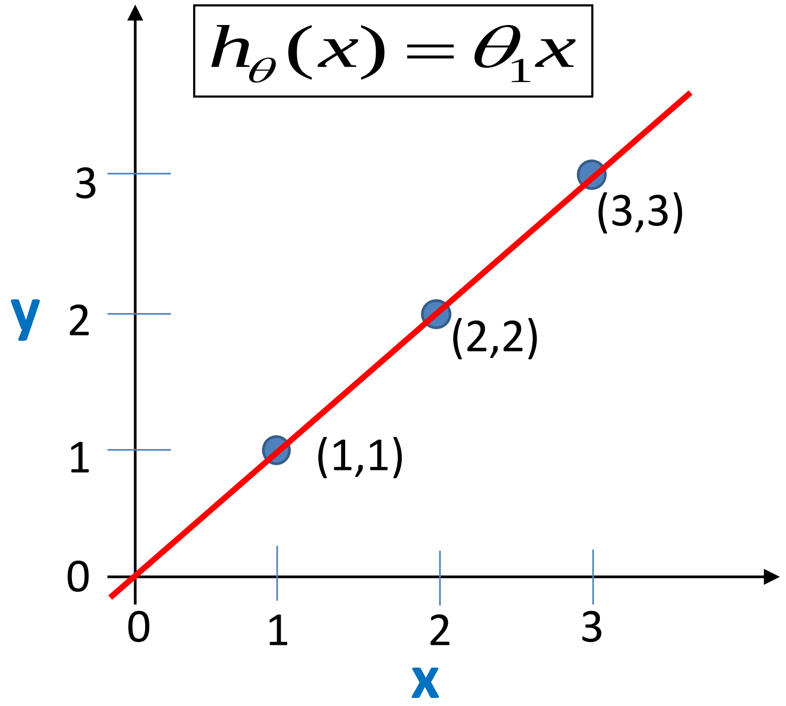
$$\boxed{\underset{\theta_1}{\min imize}\ J(\theta_1)}$$

Understanding Cost Function

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

$\theta_0 = 0$

$\theta_1 = 1$



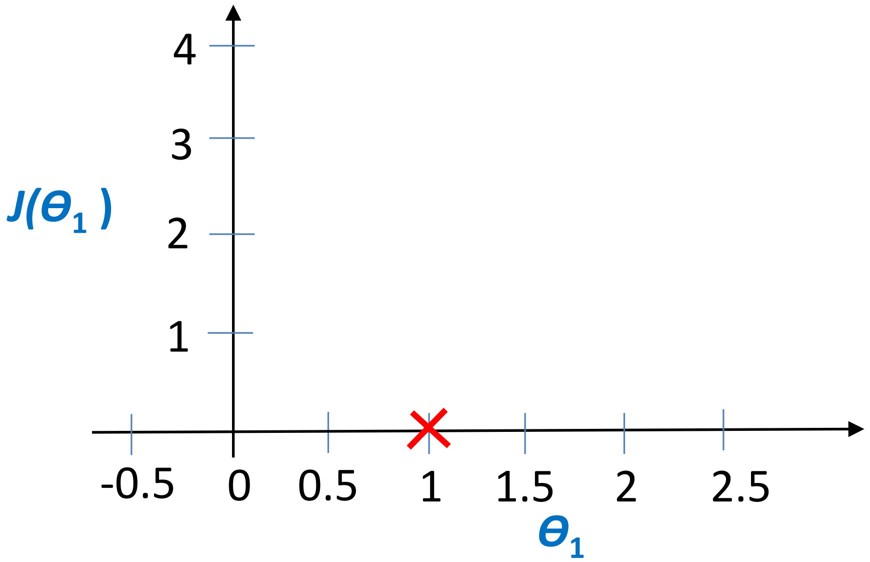
$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$
$$= \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

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

$$\boldsymbol{J(\theta_1) = J(1) = 0}$$

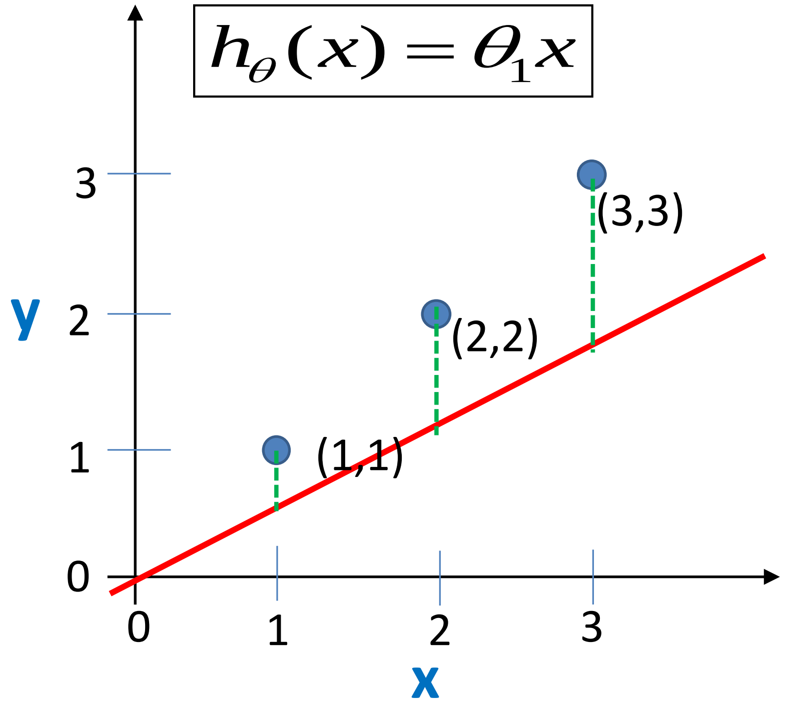
Cost Function $J(\theta_1)$

Function of parameter $\boldsymbol{\theta_1}$



Understanding Cost Function

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



$\theta_0 = 0$

$\theta_1 = 0.5$

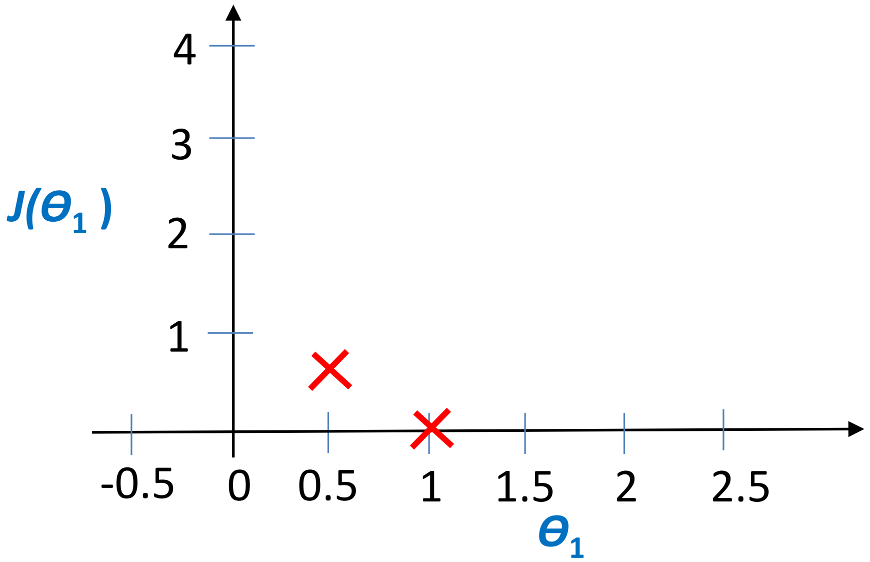
$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2*3}((0.5*1-1)^2 + (0.5*2-2)^2 + (0.5*3-3)^2)$$
$$= \frac{1}{2*3}((-0.5)^2 + (-1)^2 + (-1.5)^2) = \frac{3.5}{6} \approx 0.58$$

$\boldsymbol{J(\theta_1) = J(0.5) = 0.58}$

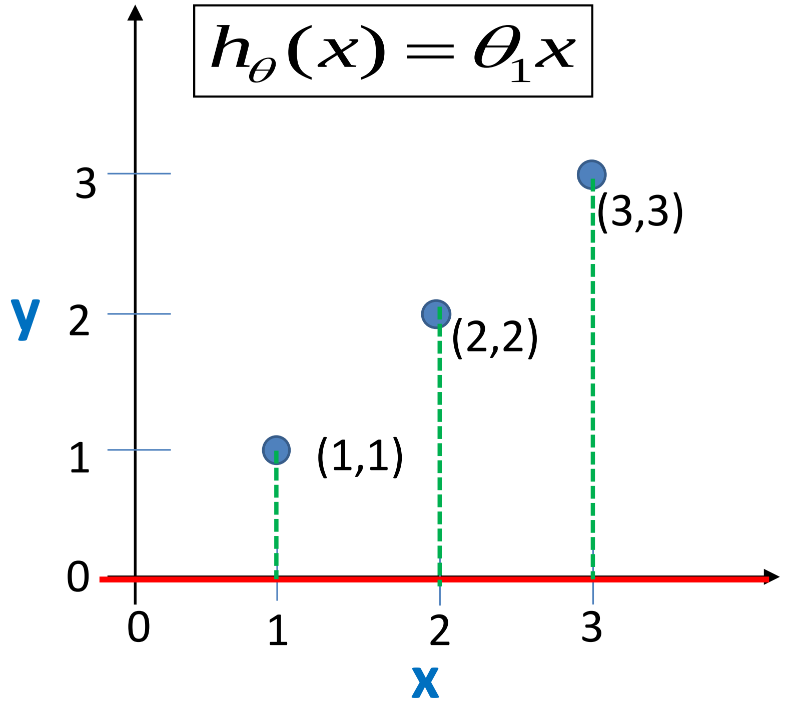
Cost Function $J(\theta_1)$

Function of parameter $\boldsymbol{\theta_1}$



Understanding Cost Function

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



$\theta_0 = 0$

$\theta_1 = 0$

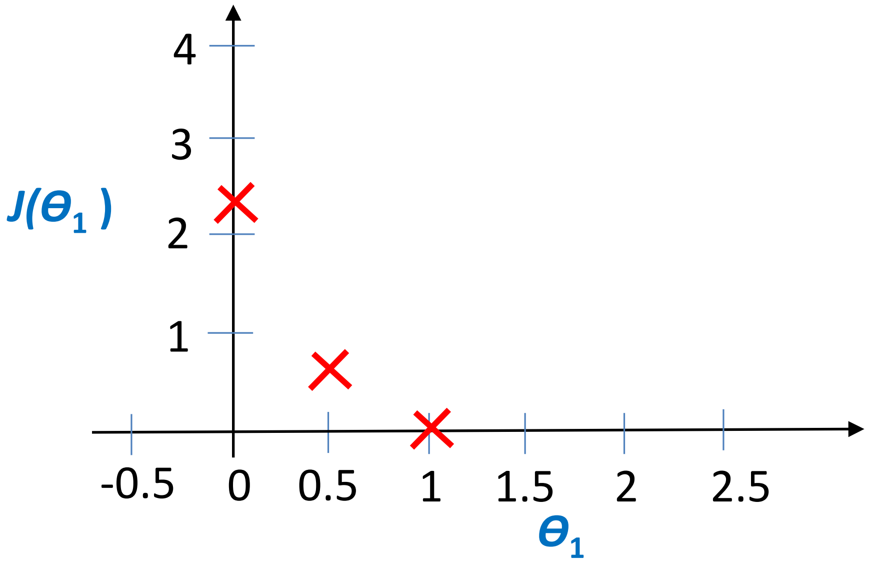
$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2*3}((0*1-1)^2 + (0*2-2)^2 + (0*3-3)^2)$$
$$= \frac{1}{2*3}((-1)^2 + (-2)^2 + (-3)^2) = \frac{14}{6} \approx 2.33$$

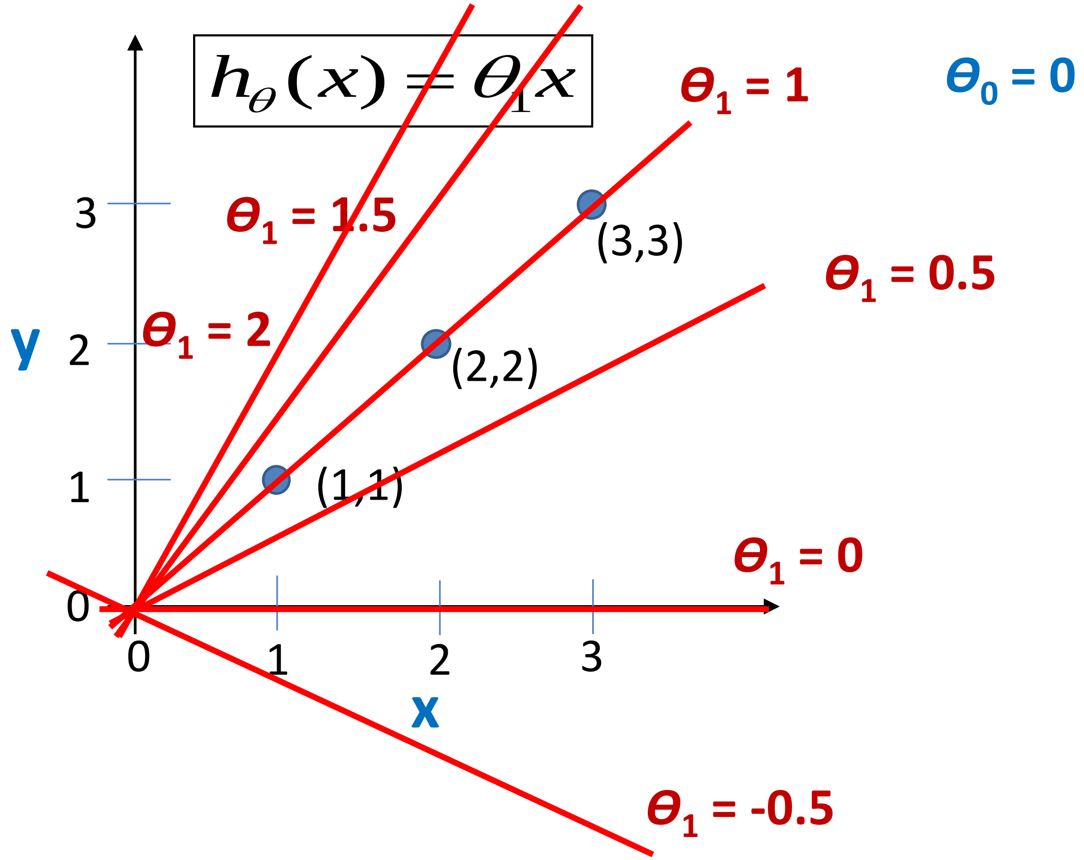
$\boldsymbol{J(\theta_1) = J(0) = 2.33}$

Cost Function $J(\theta_1)$

Function of parameter $\boldsymbol{\theta_1}$



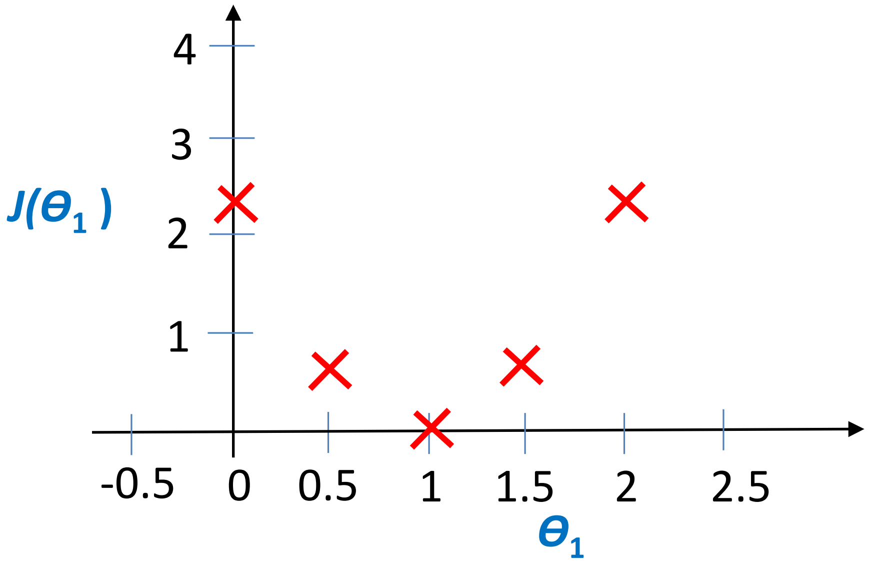
Understanding Cost Function



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

Cost Function $J(\theta_1)$

Function of parameter $\boldsymbol{\theta_1}$



Cost Optimization

Finding $\boldsymbol{\theta_1}$ to minimize $J(\boldsymbol{\theta_1})$

$\boxed{\theta_1 = 1}$

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

- Parameters $$\boxed{\theta_0 \text{ and } \theta_1}$$

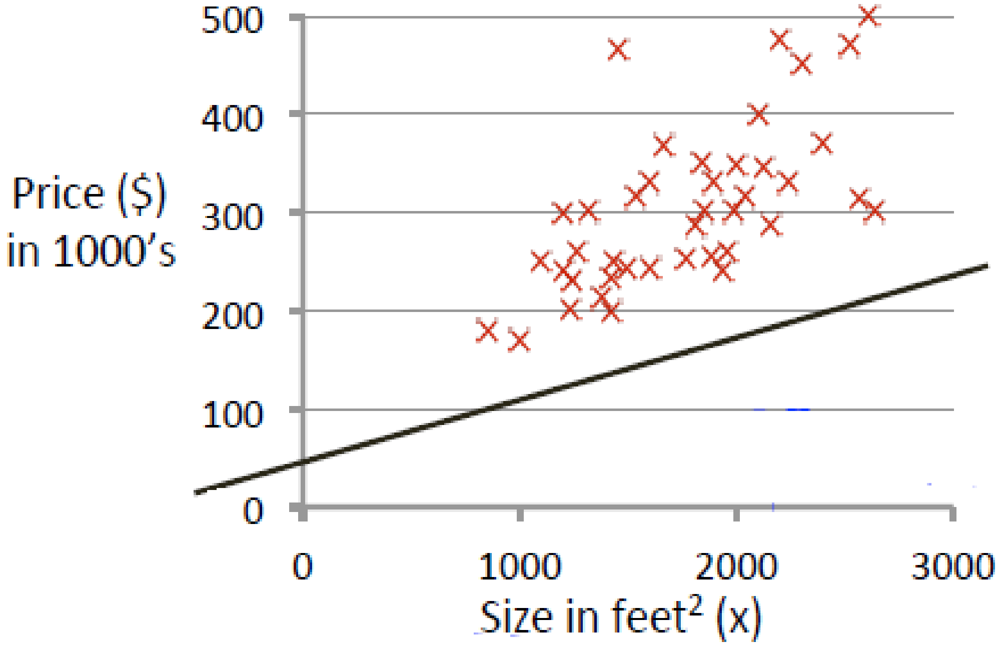
- Cost Function $$\boxed{J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2}$$

- Goal $$\boxed{\min_{\theta_0, \theta_1} imize \quad J(\theta_0, \theta_1)}$$

Understanding Cost Function

$h_\theta(x)$

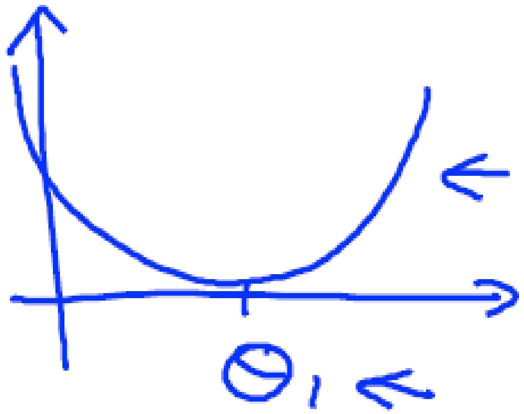
(for fixed θ_0, θ_1, this is a function of x)



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

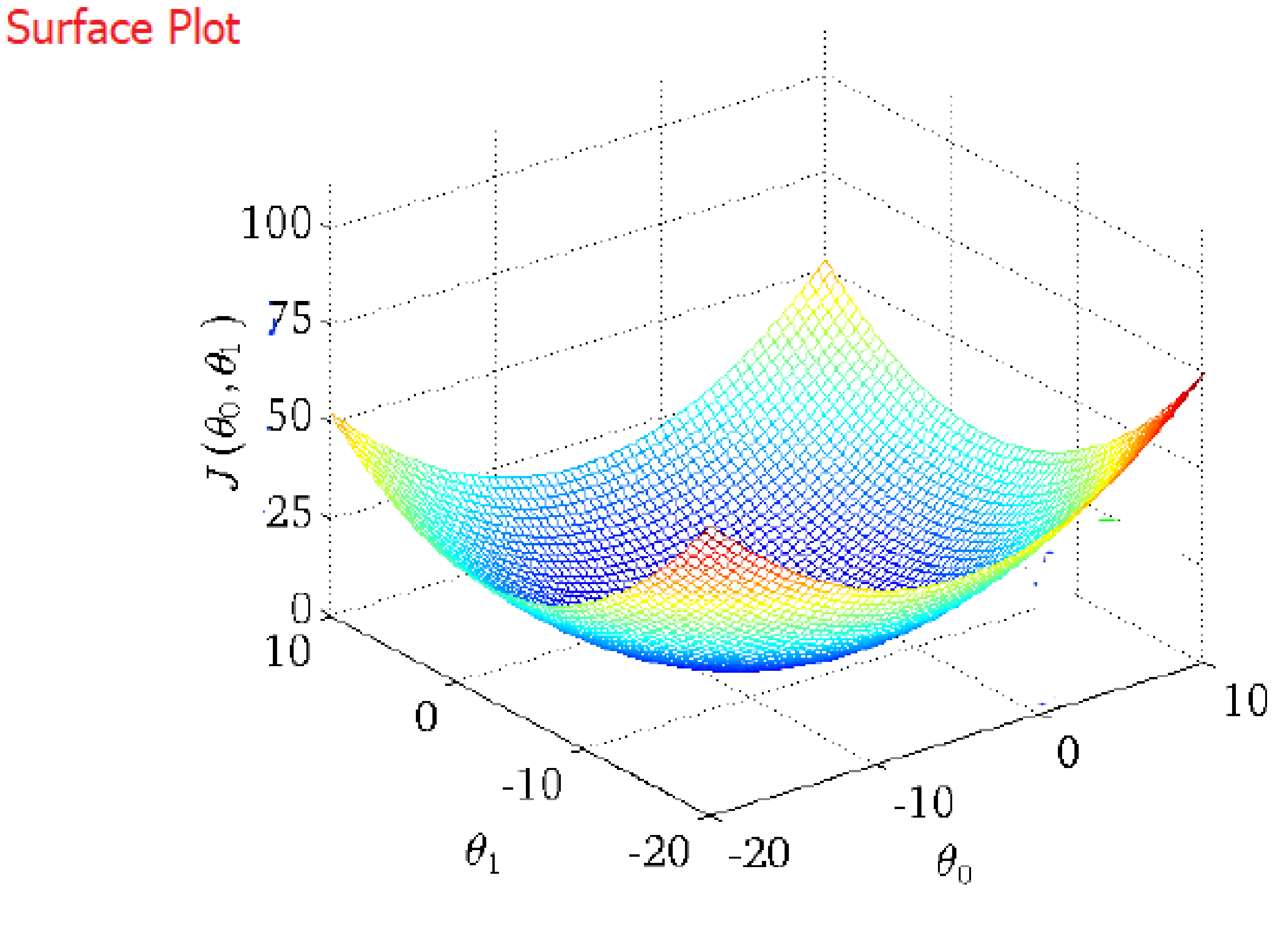
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



$\boxed{\theta_0, \theta_1}$

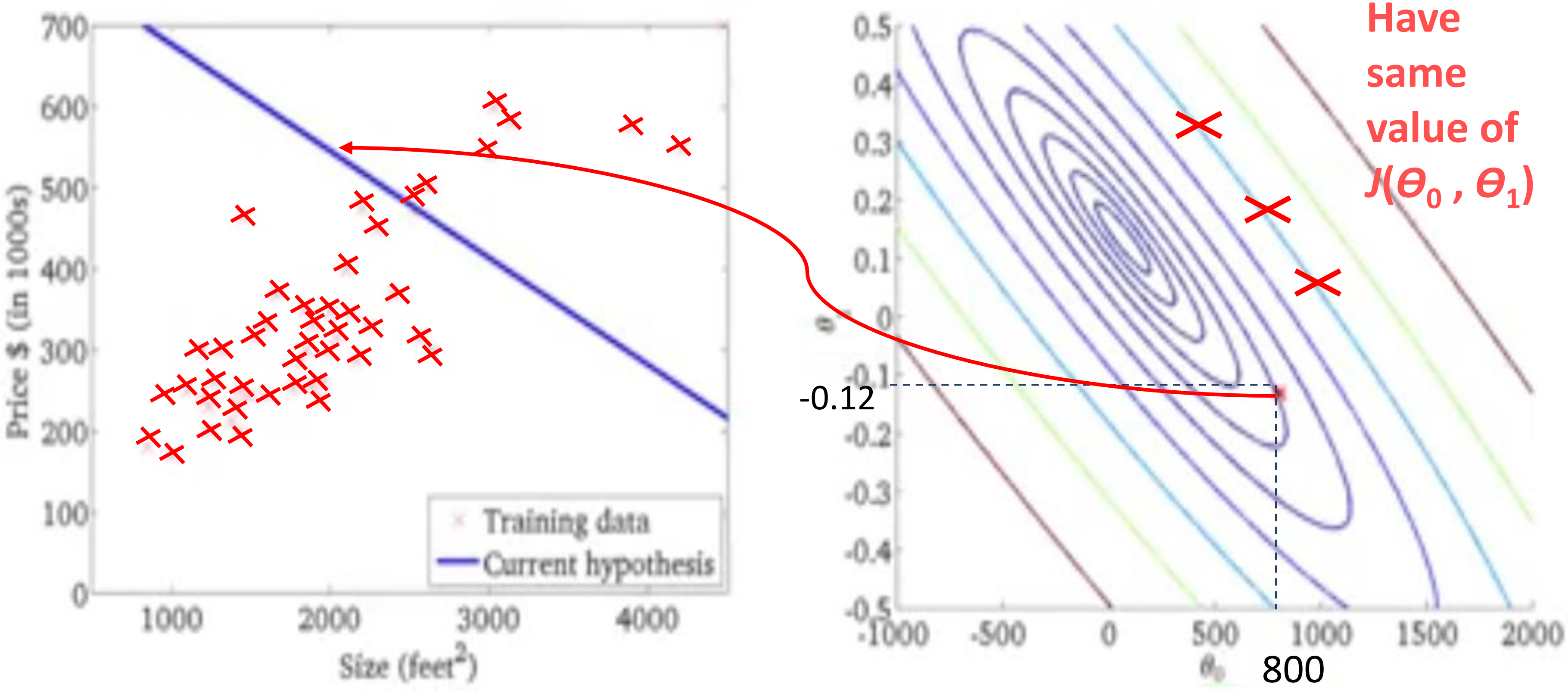
Understanding Cost Function



Understanding Cost Function

$h_\theta(x)$ | $J(\theta_0, \theta_1)$

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



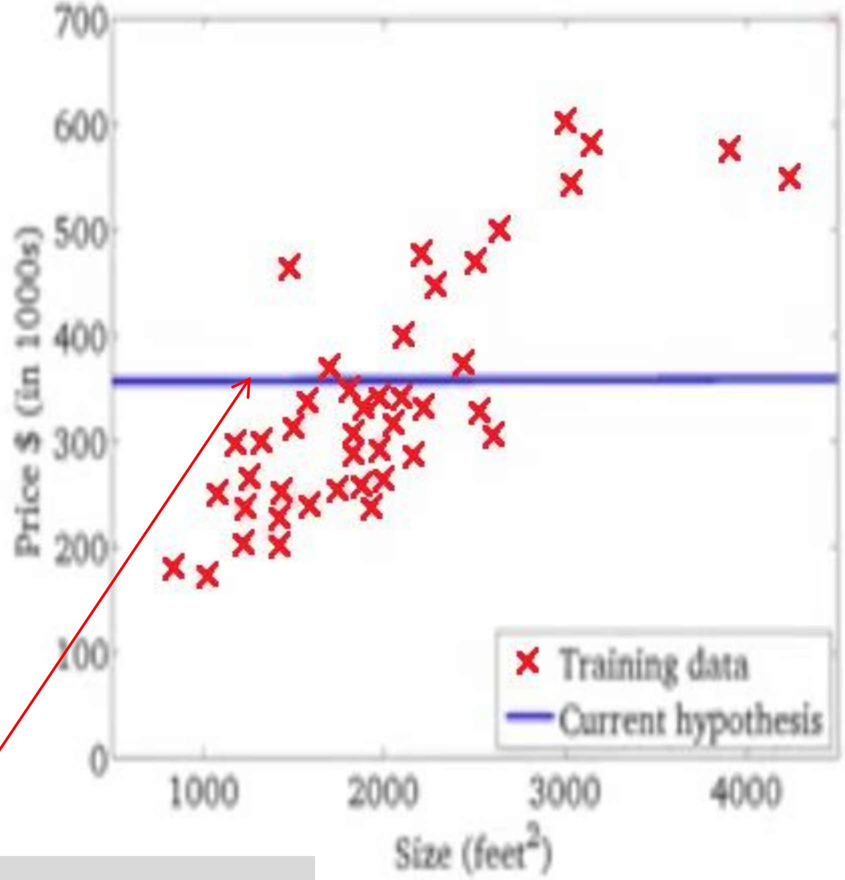
Contour Plot / Contour Figure

Elliptical contours show the set of points which takes on the same value of $J(\theta_0, \theta_1)$

Understanding Cost Function

$h_\theta(x)$

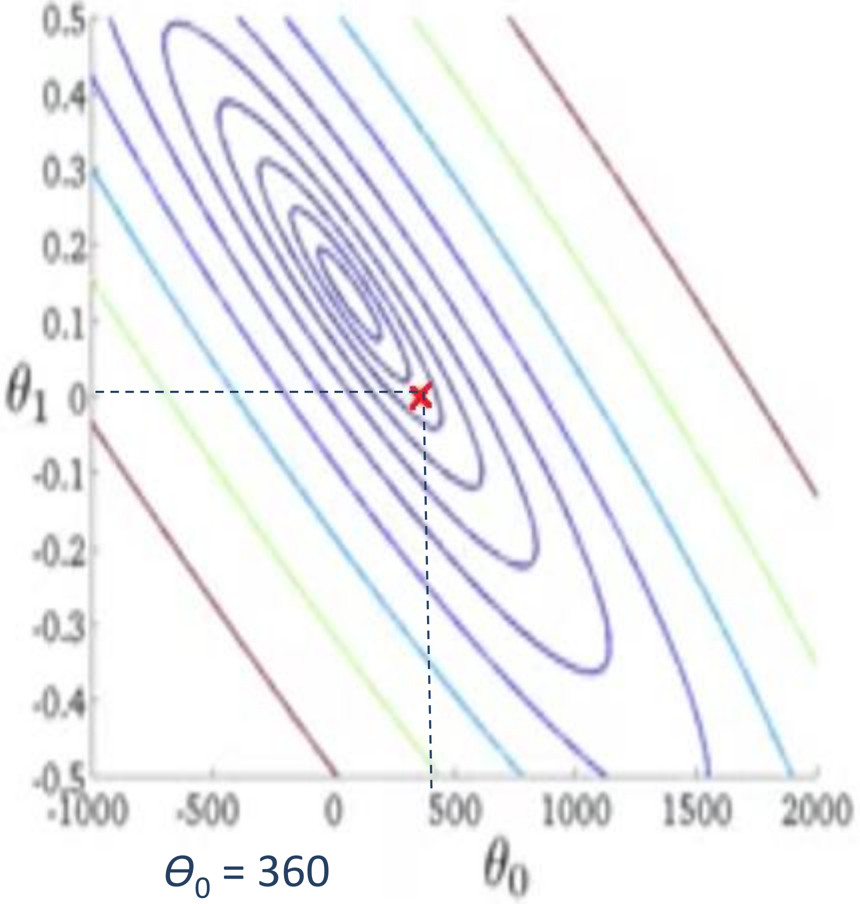
(for fixed θ_0, θ_1, this is a function of x)



$$h_\theta(x) = 360 + 0 * x$$

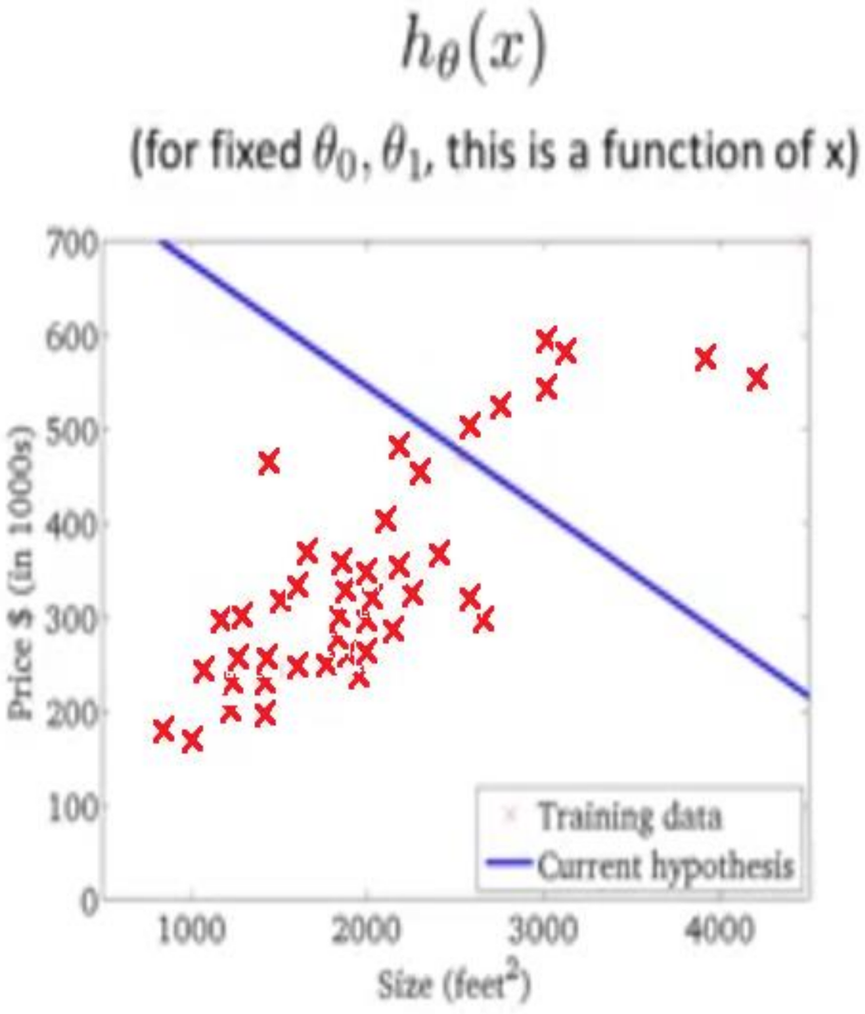
$J(\theta_0, \theta_1)$

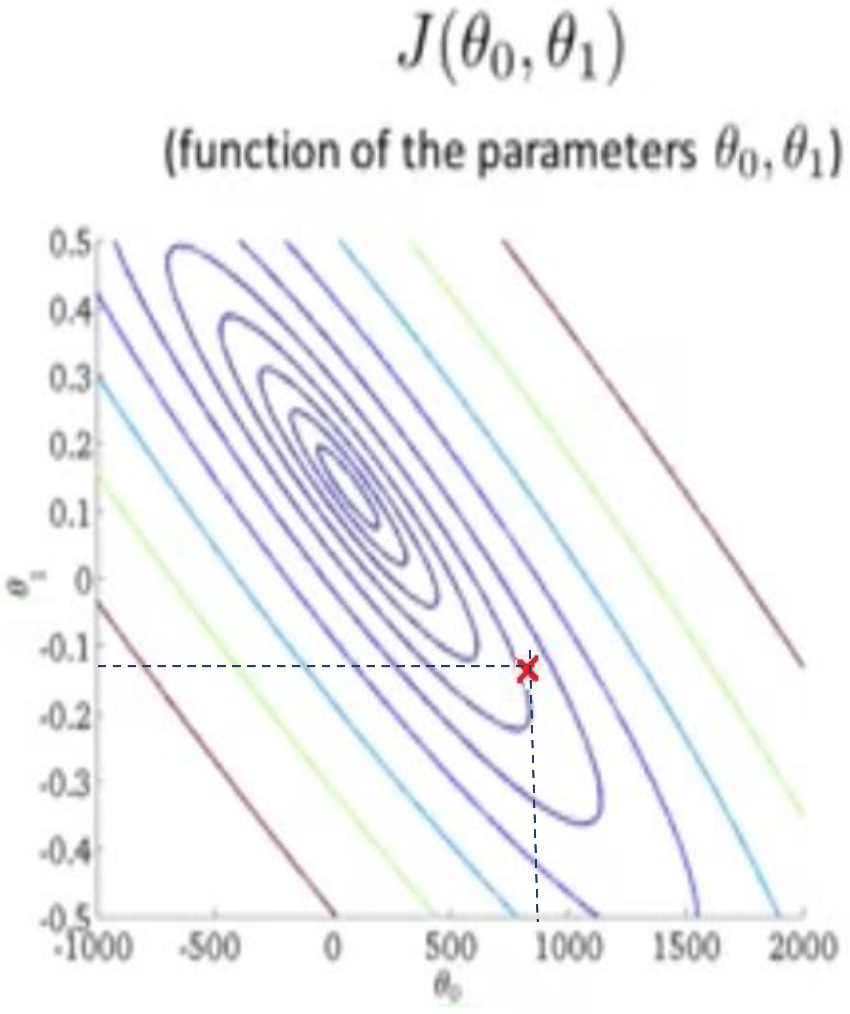
(function of the parameters θ_0, θ_1)



$\theta_0 = 360$
$\theta_1 = 0$

Understanding Cost Function

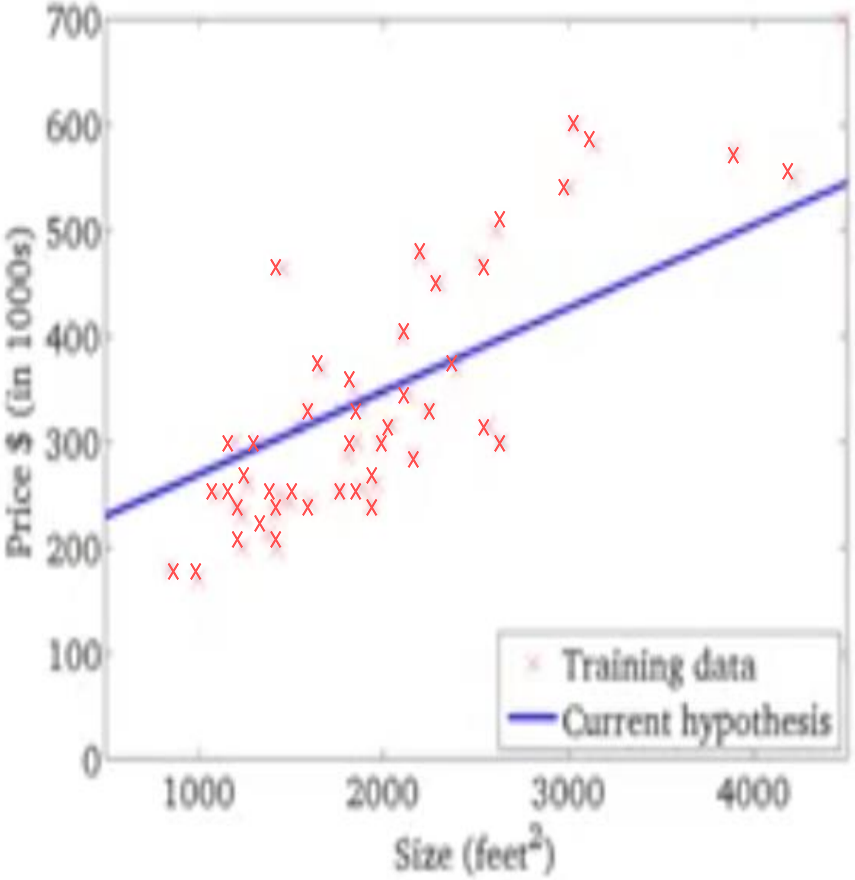




Understanding Cost Function

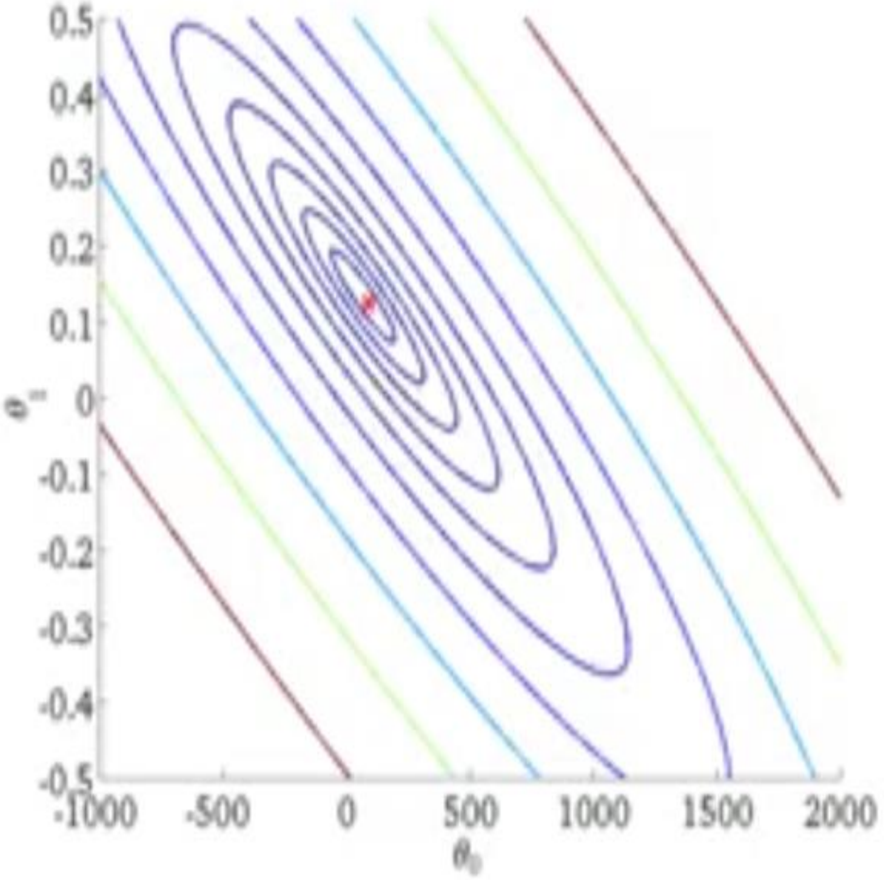
$h_\theta(x)$

(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)



Slide from Andrew Ng's Coursera course on Machine Learning

Unit 2

Gradient Descent

Machine Learning – Regression and Decision Trees

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

- Parameters $$\theta_0 \text{ and } \theta_1$$

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

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

Gradient Descent

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

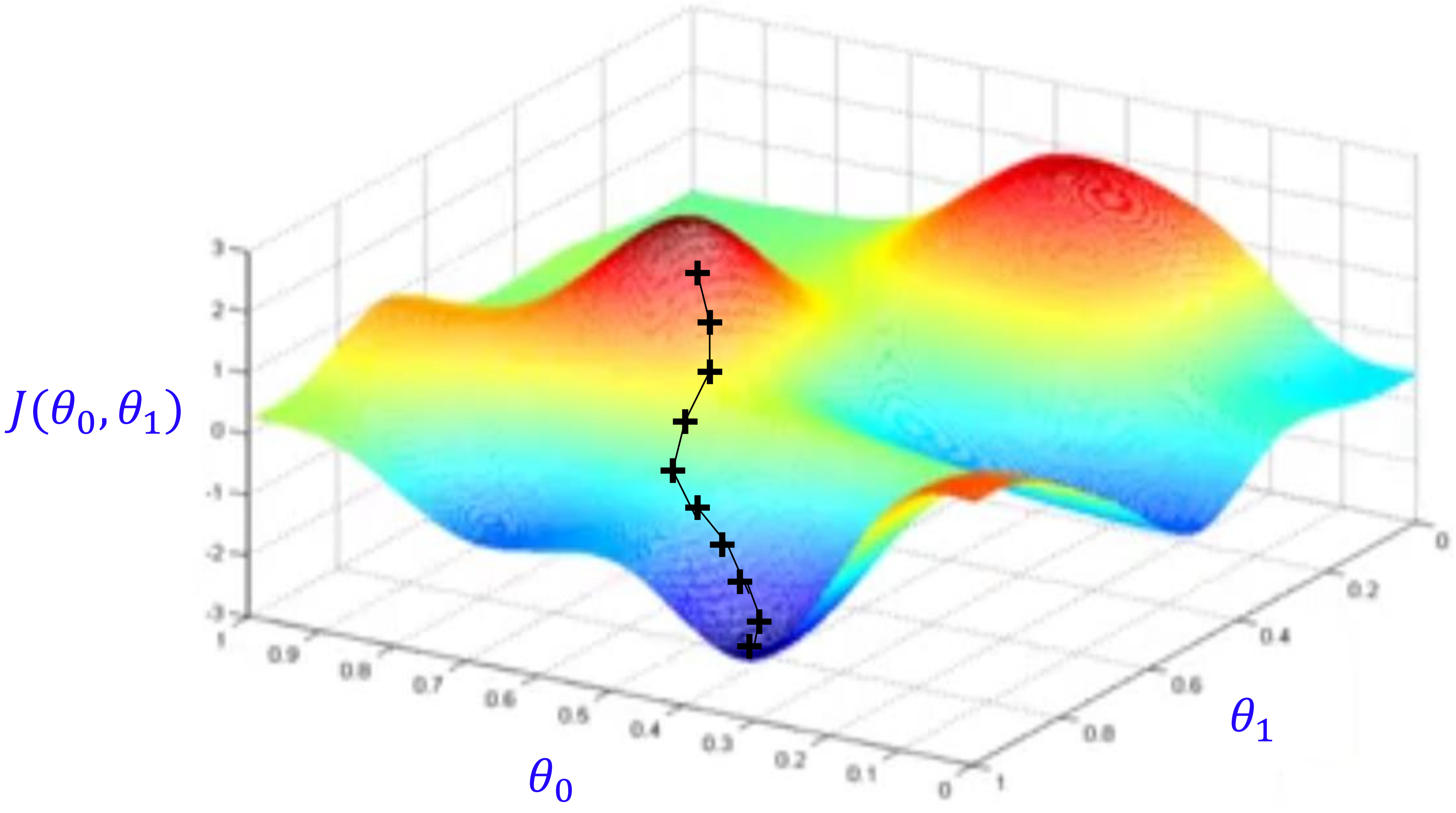
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

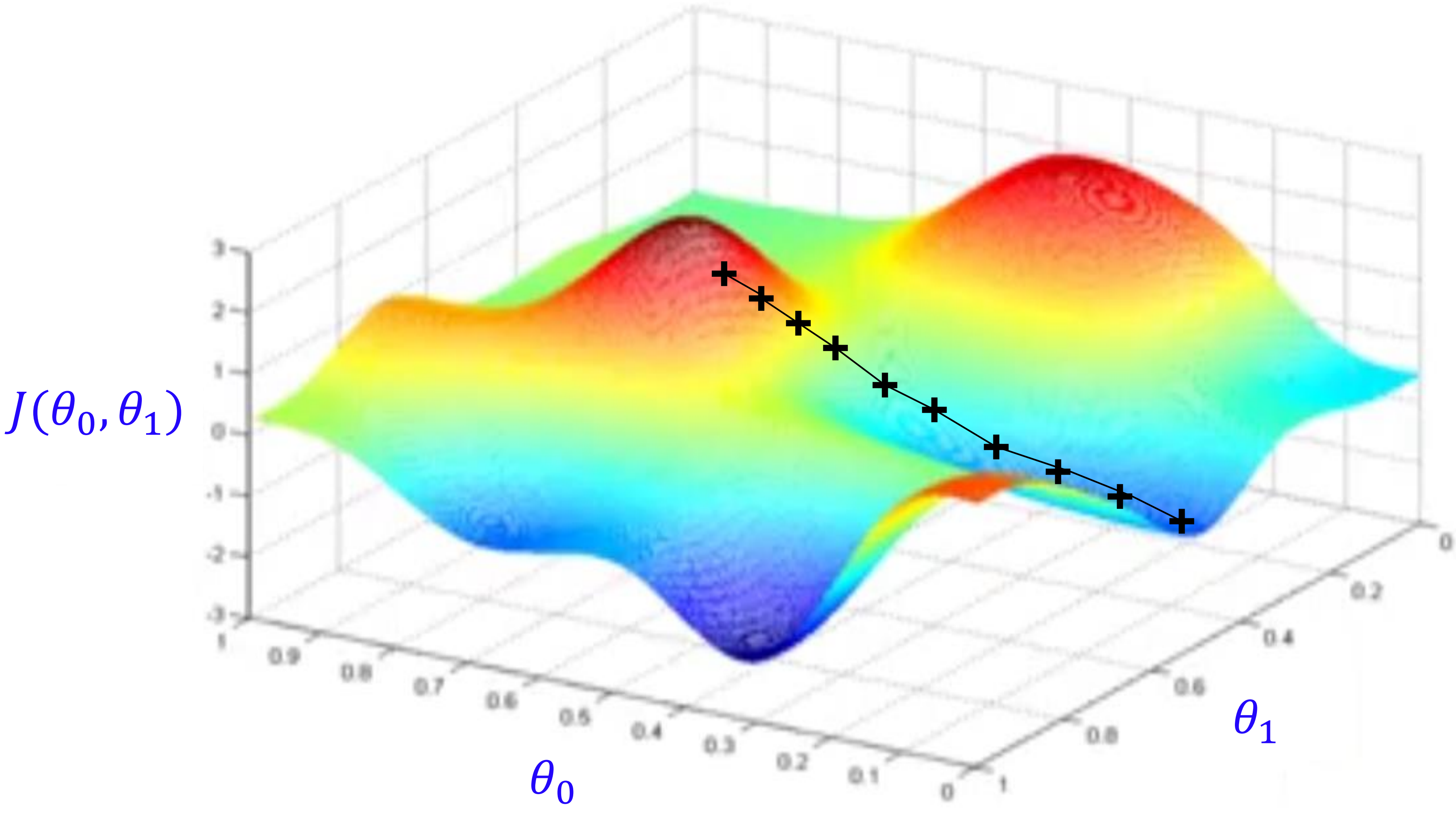
Outline:

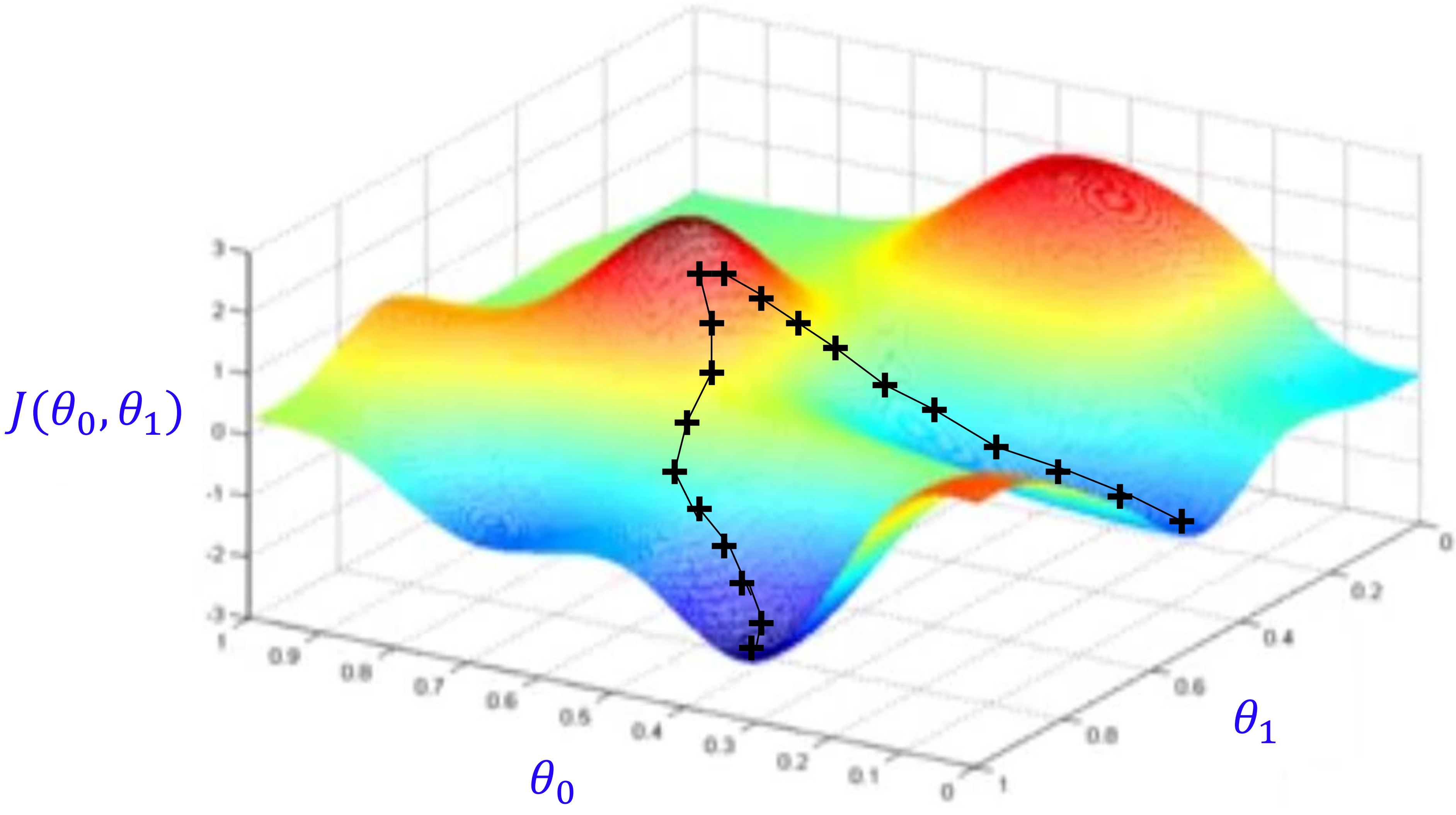
- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

  until we hopefully end up at a minimum

Slide from Andrew Ng's Coursera course on Machine Learning


$J(\theta_0, \theta_1)$
θ_0
θ_1


$J(\theta_0, \theta_1)$
θ_0
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
0
0.2
0.4
0.6
0.8
1
θ_0
θ_1

Unit 2

19

Gradient Descent for Linear Regression

Machine Learning – Regression and Decision Trees

Gradient Descent Algorithm

repeat until convergence {

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

}

Learning Rate

Derivative

Simultaneous update of θ_0, θ_1

Correct Implementation: Simultaneous Update

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

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

$$\theta_0 := temp0$$

$$\theta_1 := temp1$$

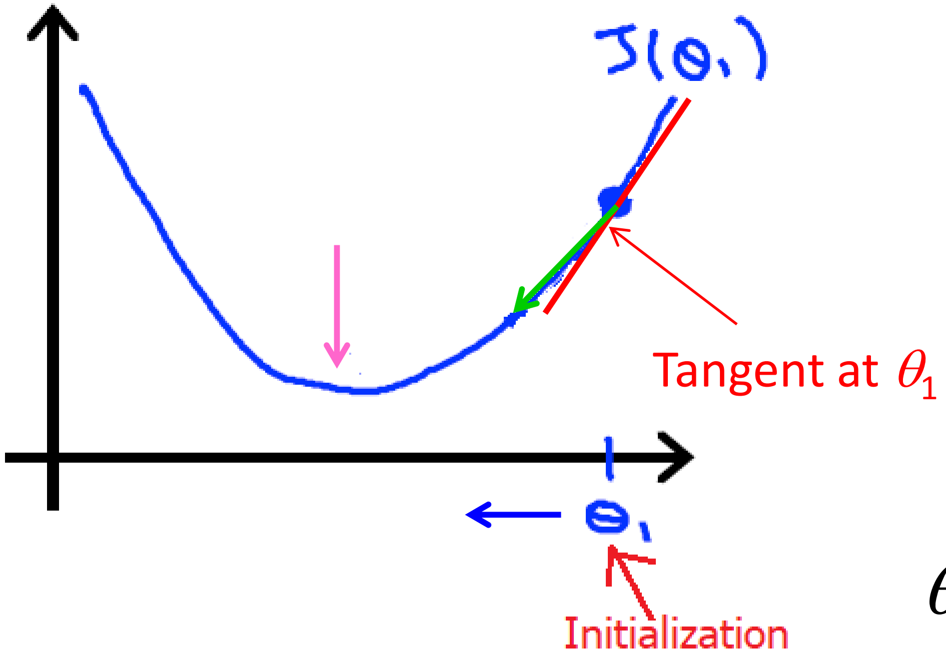
Incorrect Implementation

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

$$\theta_0 := temp0$$

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

$$\theta_1 := temp1$$

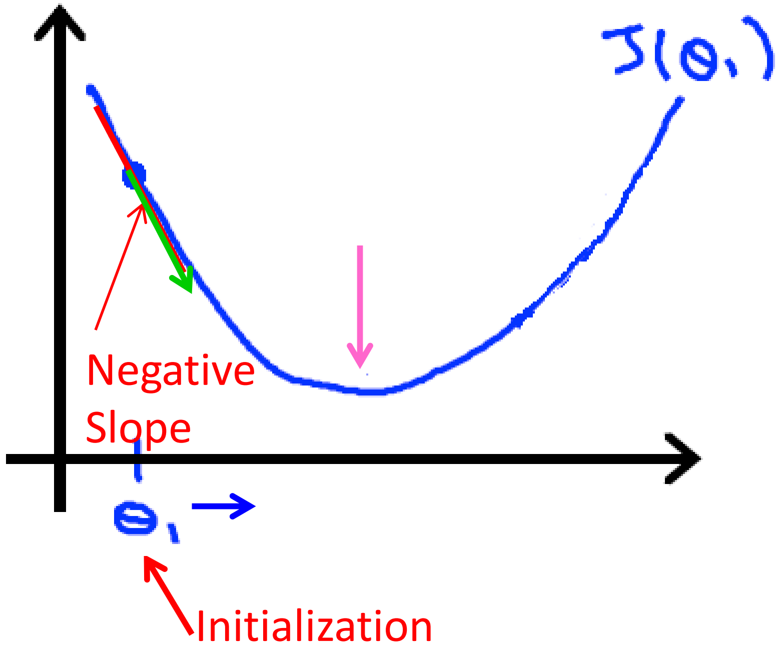


$$\theta_1 := \theta_1 - \alpha \boxed{\frac{d}{d\theta_1} J(\theta_1)}$$

Slope of Tangent at θ_1

≥0

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



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

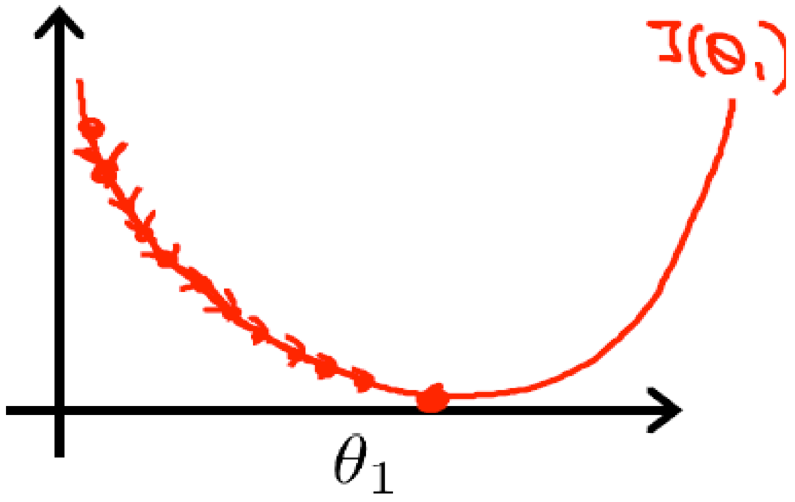
≤0

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

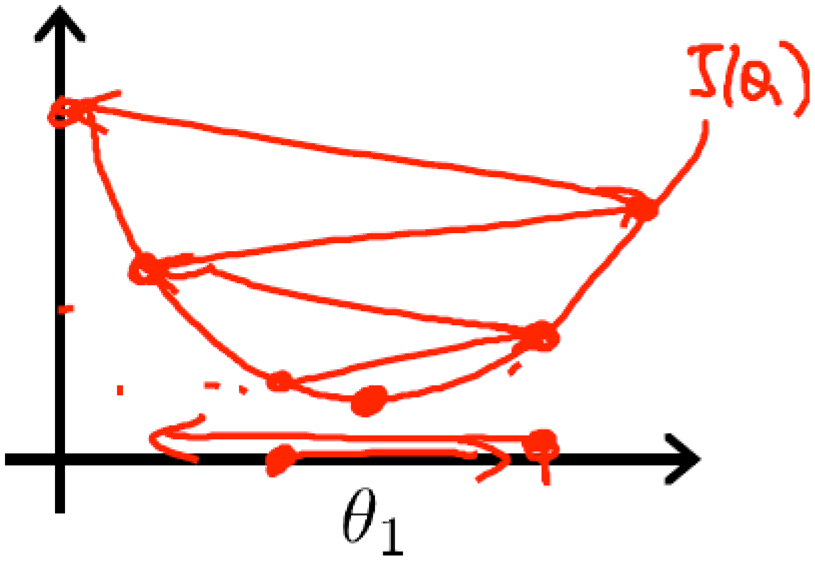
The Learning Rate

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

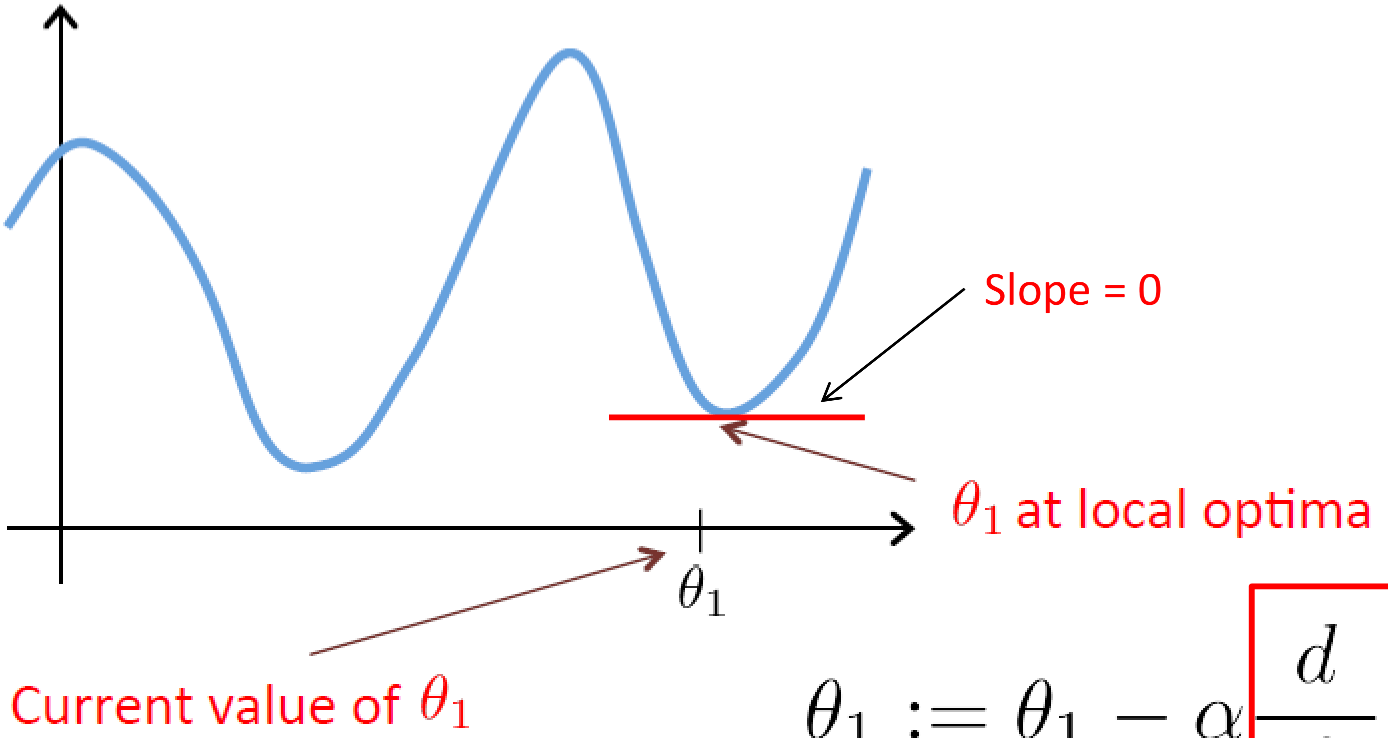
- If α is **too small**, gradient descent can be slow



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



- What if θ_1 is already at local minimum ?



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

= 0

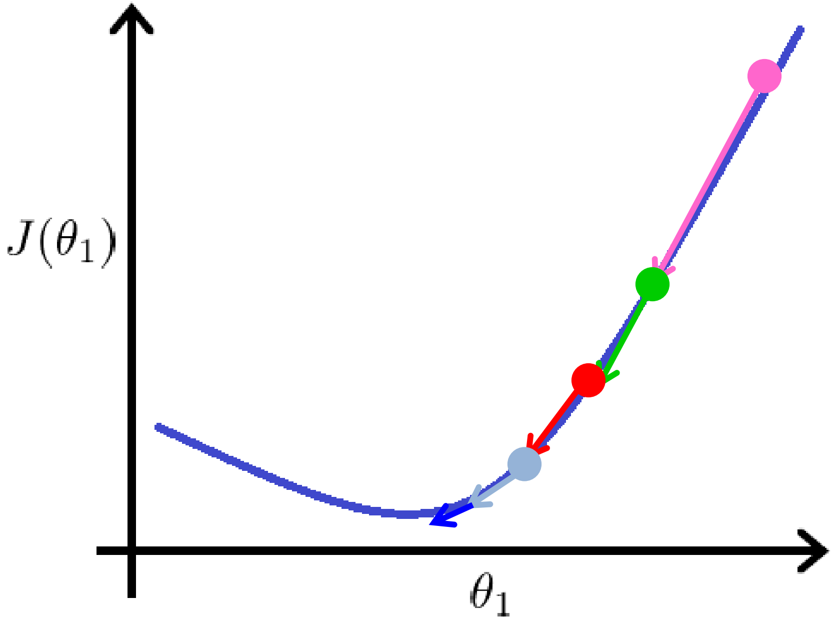
$$\theta_1 := \theta_1 - \alpha * 0$$

$$\theta_1 := \theta_1$$ ➔ No change

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 = 0 and j = 1)

}

Linear Regression Model

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

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

Apply Gradient Descent to minimize cost function of linear regression

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

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

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

$$j = 1 :$$

Gradient Descent Algorithm for Linear Regression

repeat until convergence {

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

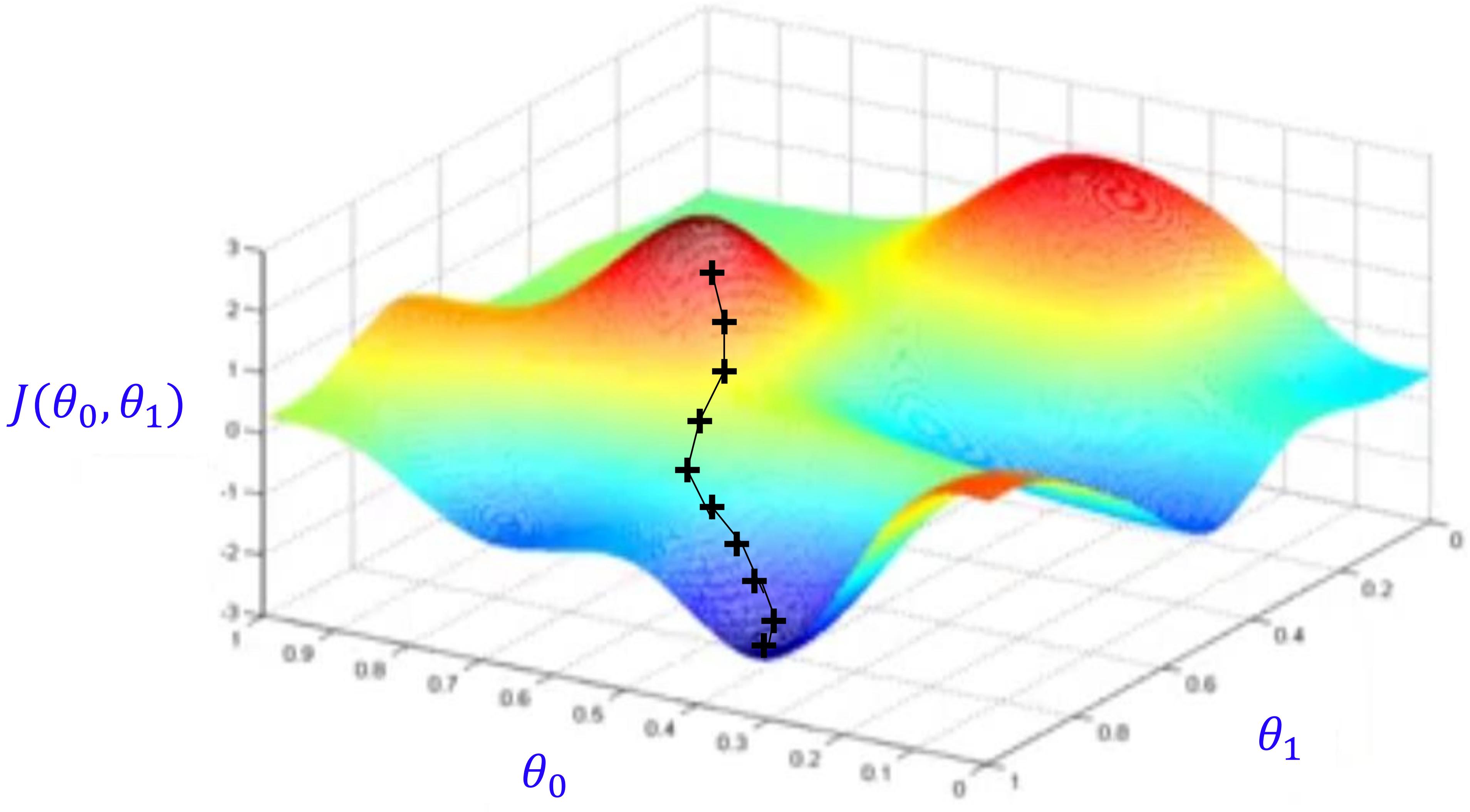
$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

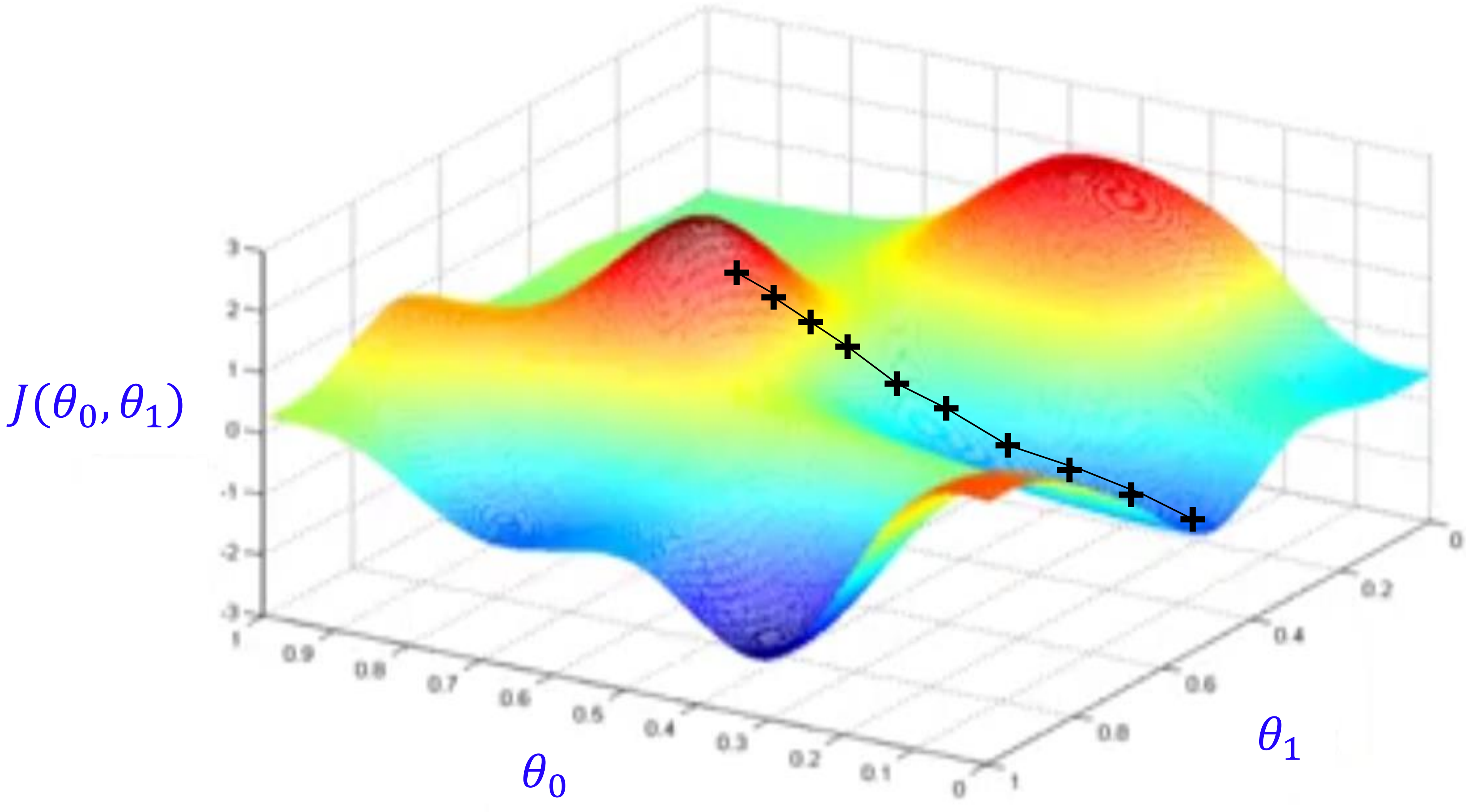
Update θ_0 and θ_1 simultaneously

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

}

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

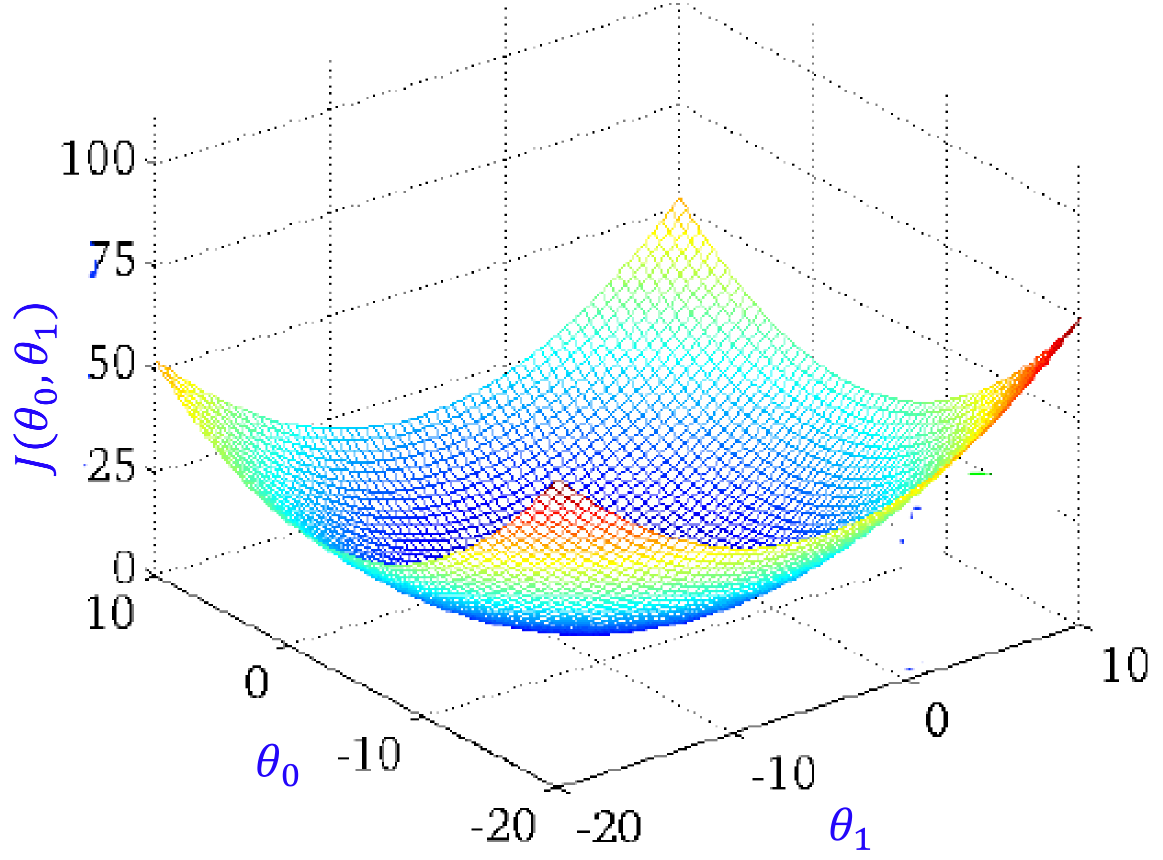

$J(\theta_0, \theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


$J(\theta_0, \theta_1)$
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

Surface Plot

Convex Function

Bowl-shaped Function

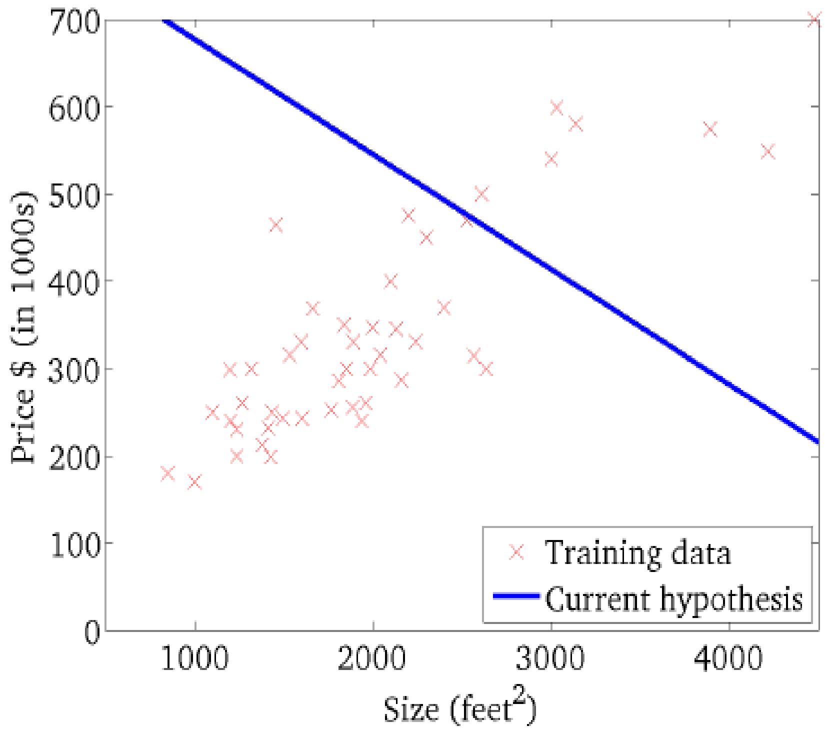


- **Cost function of linear regression has always one global minima and not the local minima.**

Slide from Andrew Ng's Coursera course on Machine Learning

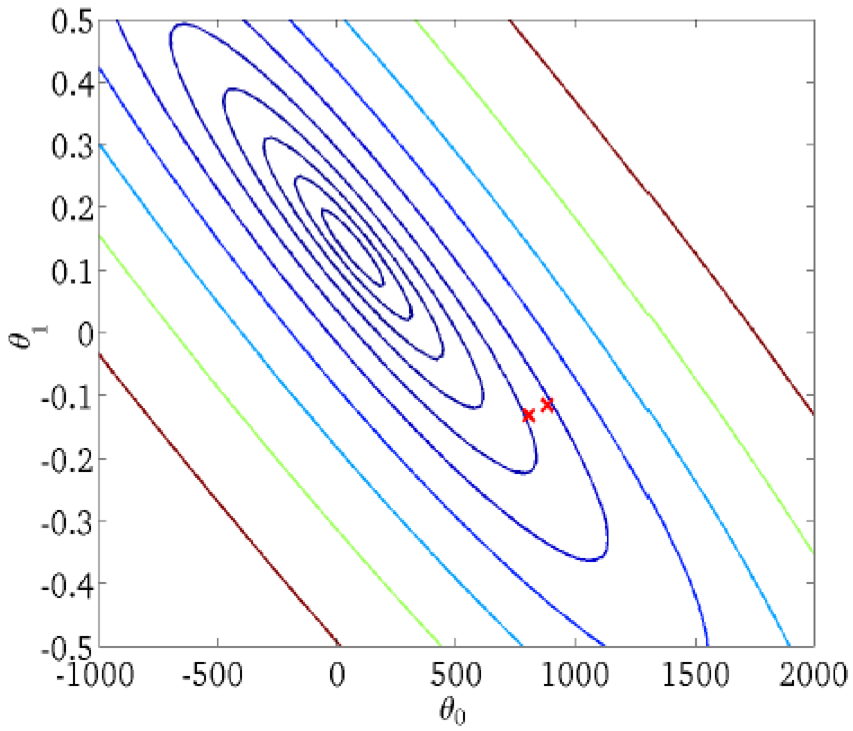
$$h_\theta(x)$$

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



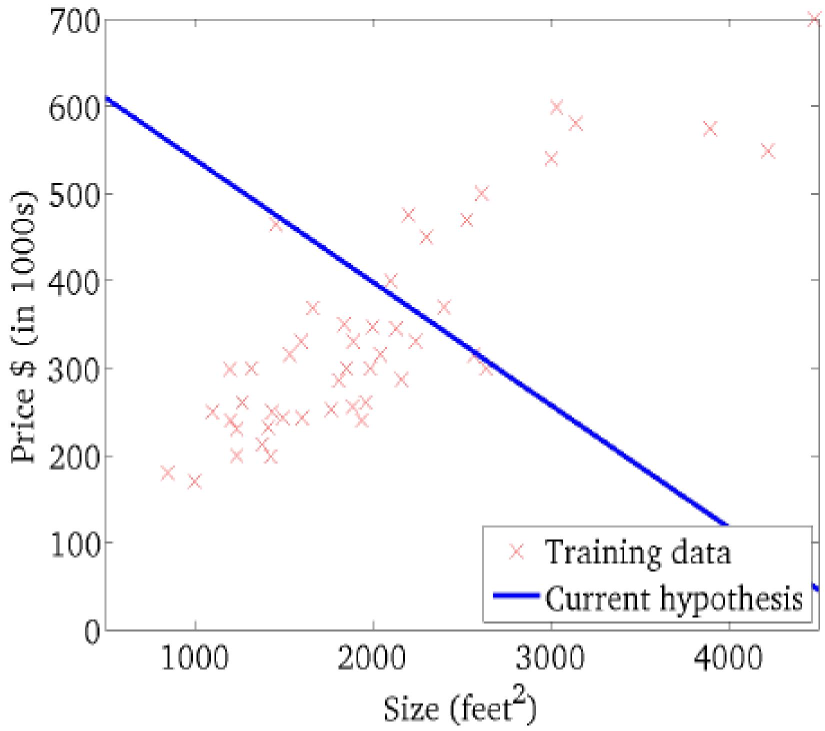
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



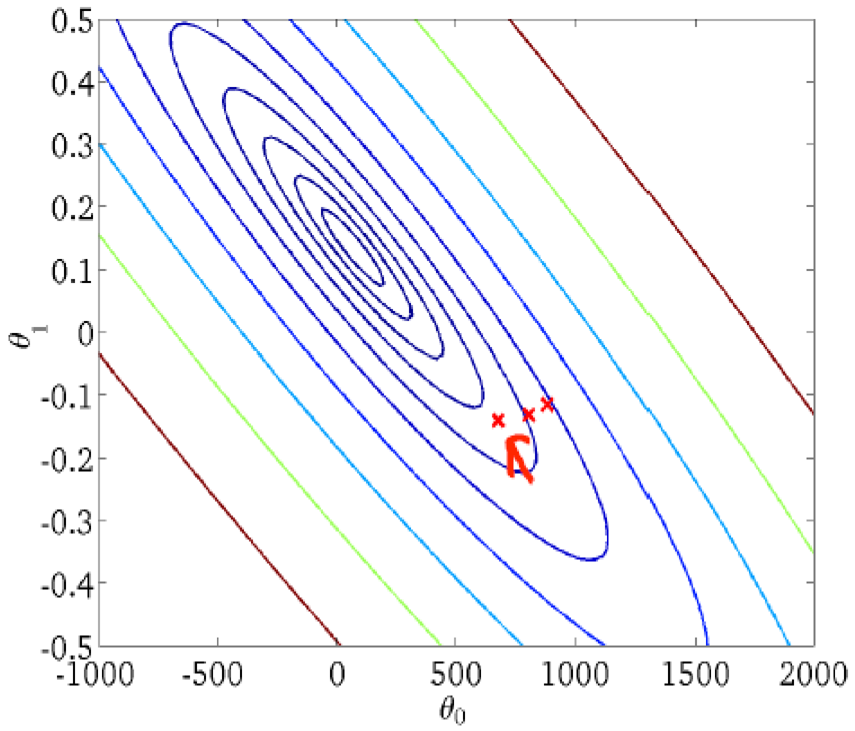
$$h_\theta(x)$$

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



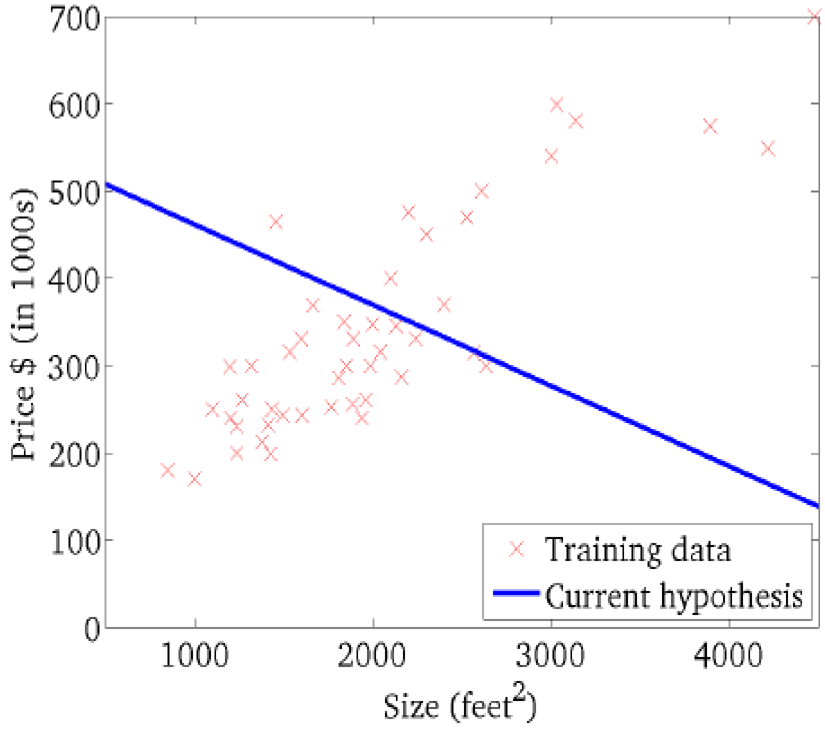
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



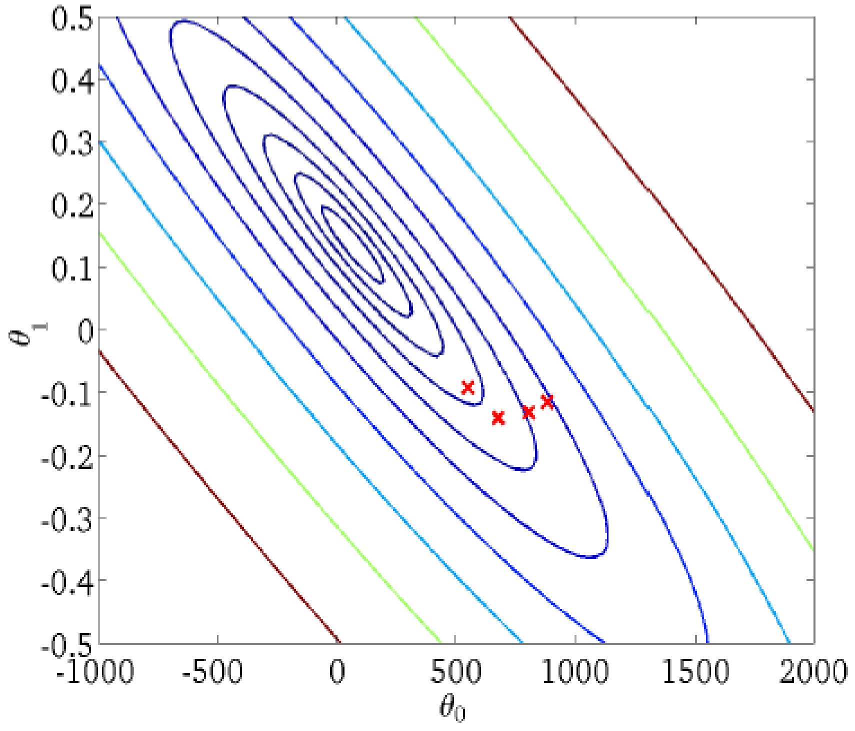
$h_\theta(x)$

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



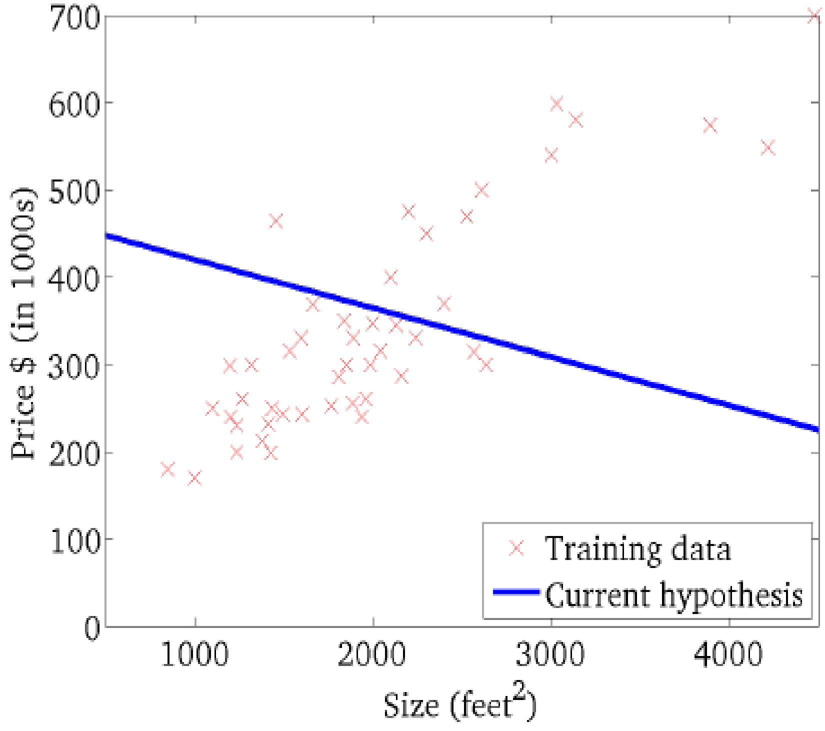
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



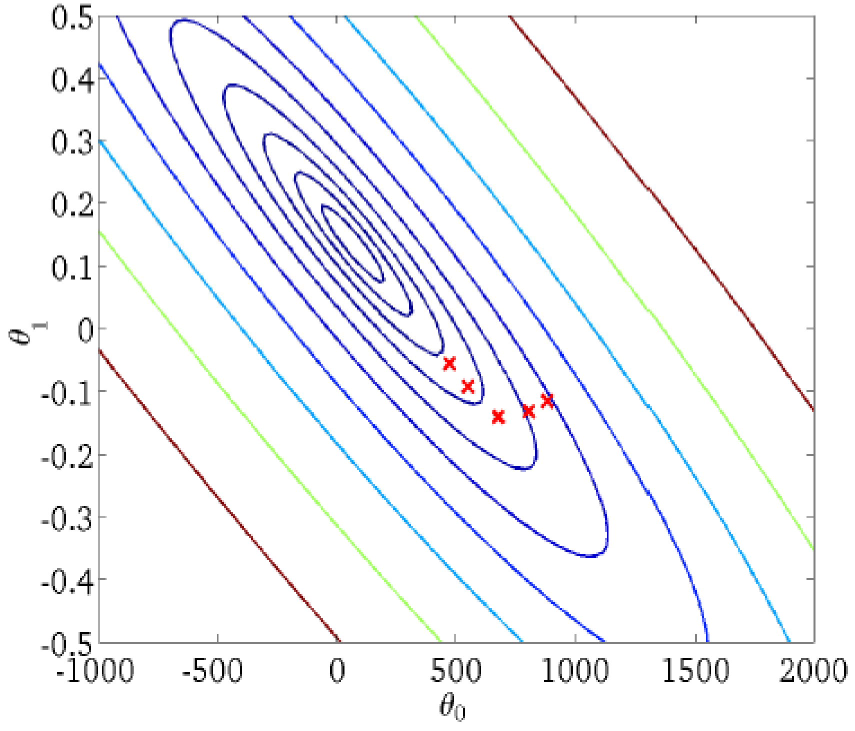
$h_\theta(x)$

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



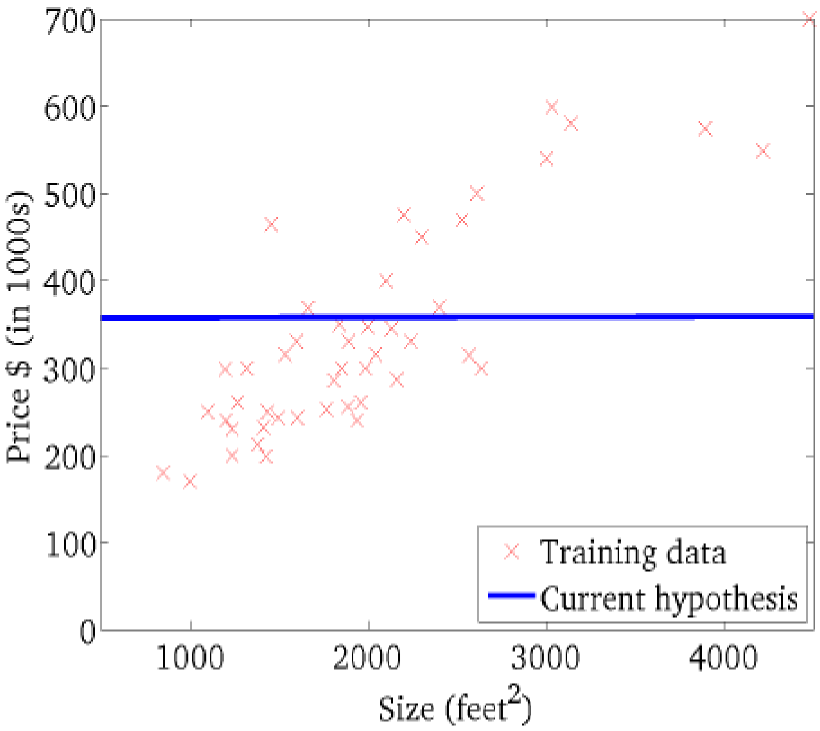
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



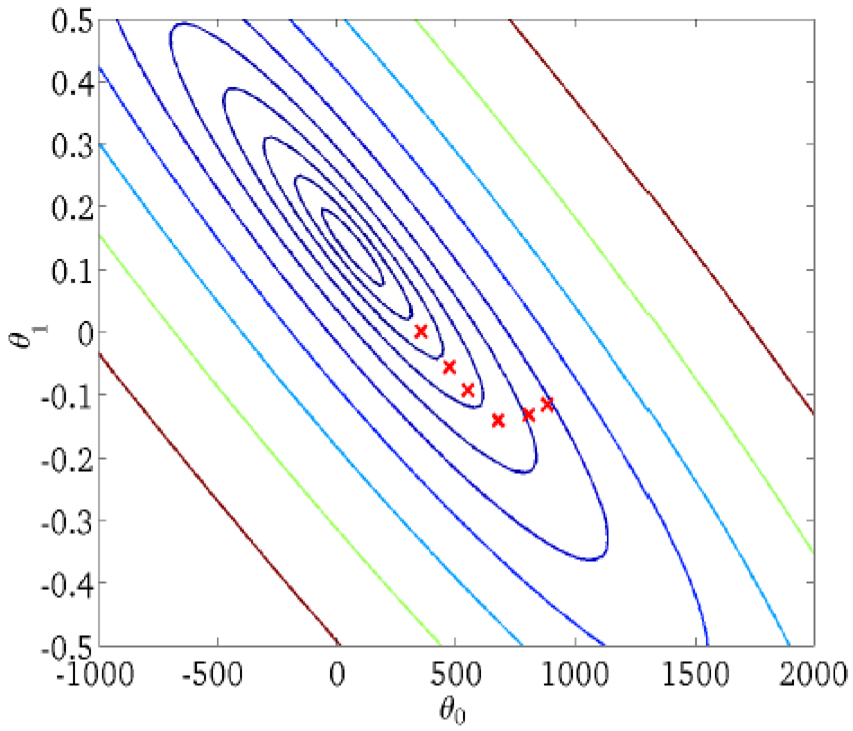
$$h_\theta(x)$$

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



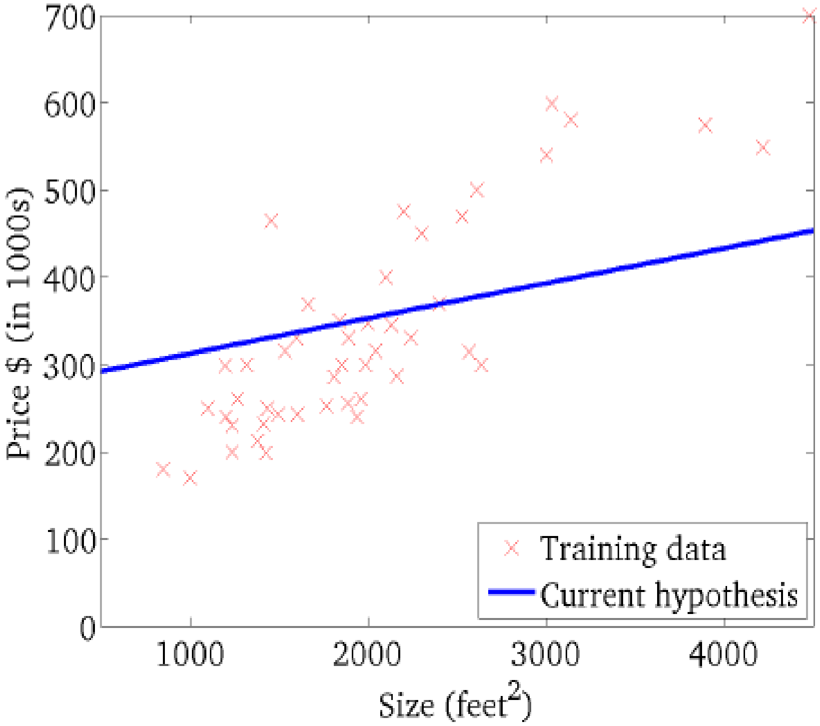
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



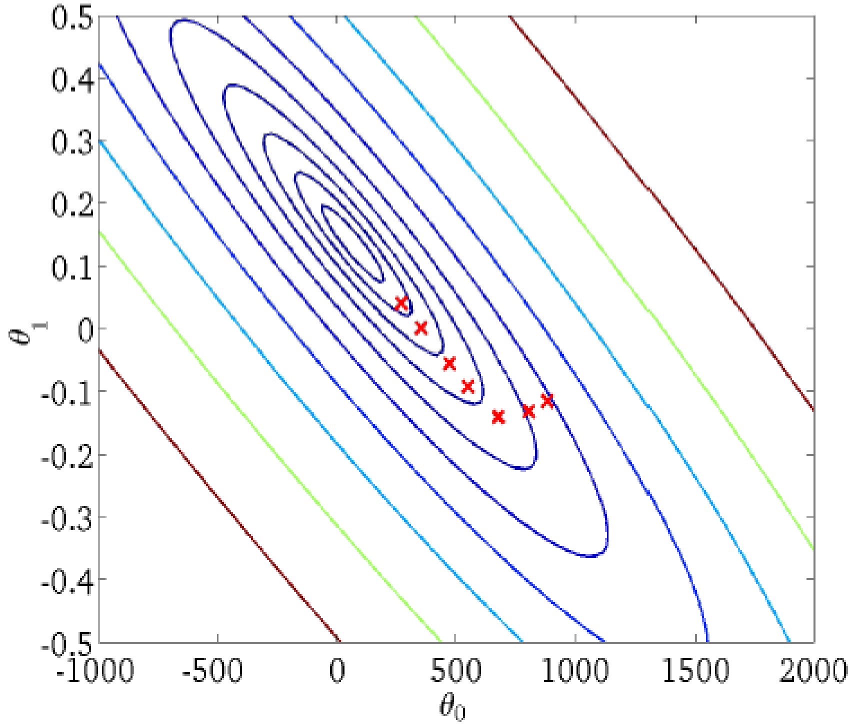
$$h_\theta(x)$$

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



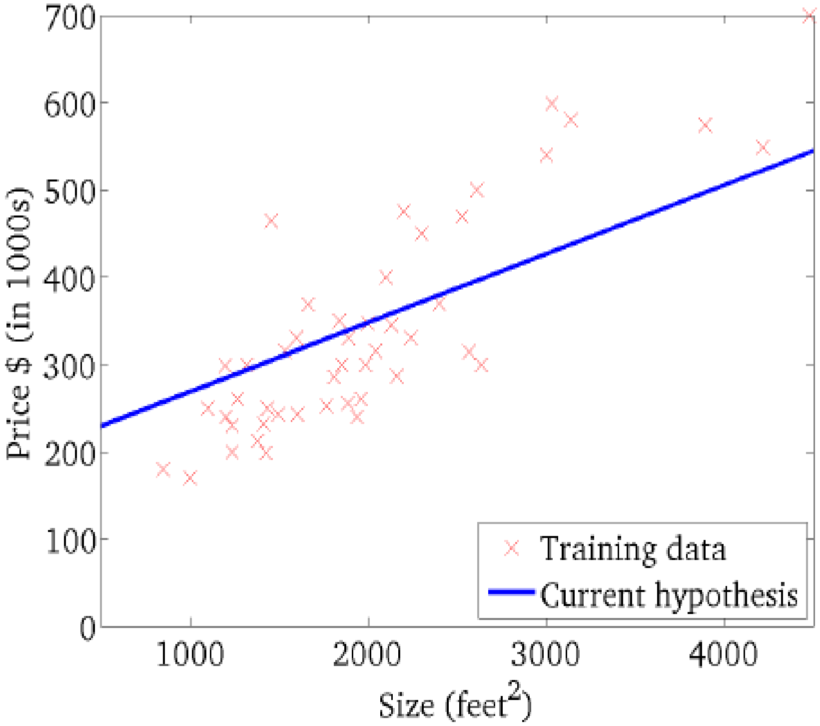
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



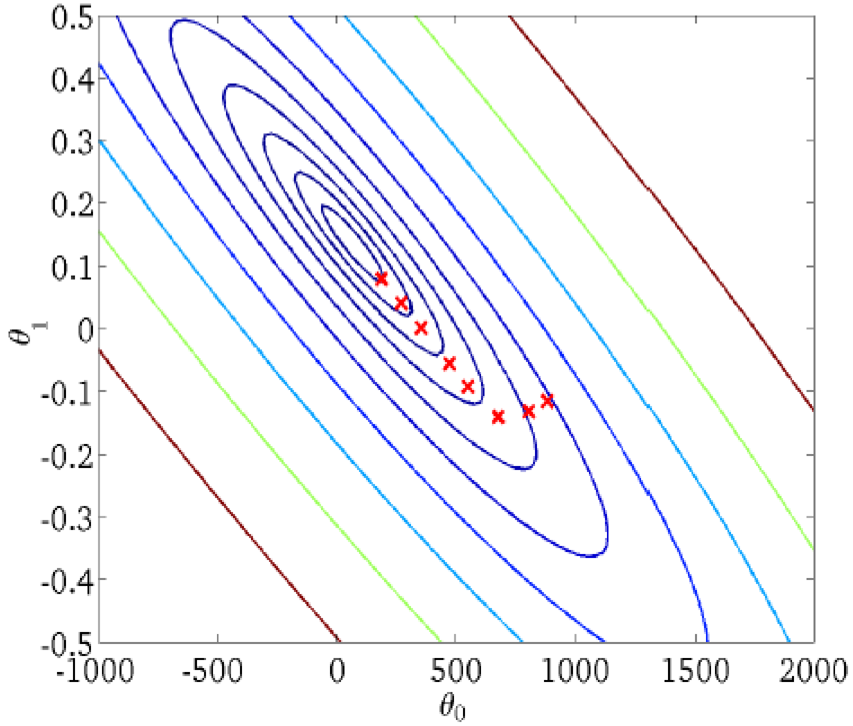
$$h_\theta(x)$$

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



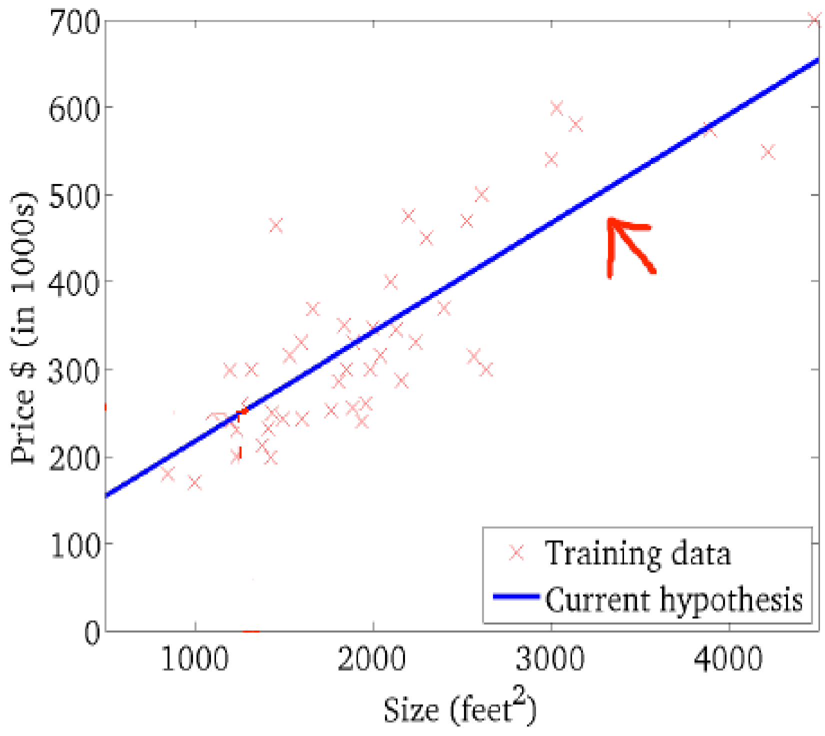
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



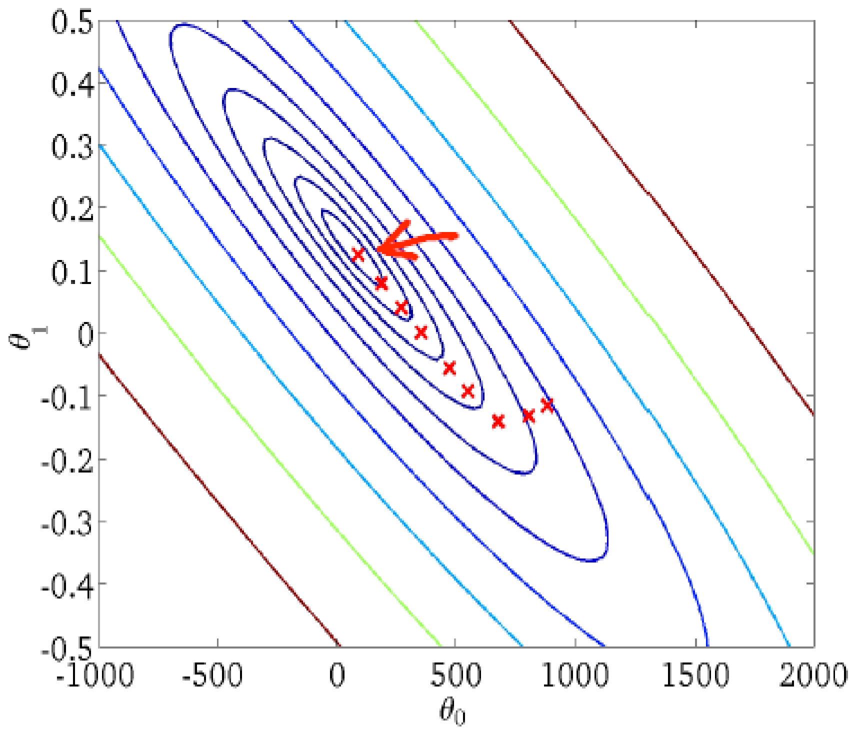
$$h_\theta(x)$$

(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 and Stochastic Gradient Descent

- GD is an iterative algorithm, that starts from a random point on the function and travels down its slope in steps until it reaches the lowest point of that function
- Batch GD
  - "Batch": Each step of gradient descent uses all the training examples.
- Stochastic GD
  - Randomly pick one data point from whole dataset at each iteration → reduce computations
- Mini-batch GD
  - Sample small number of data points at each step

Multiple (Multivariable) Linear Regression

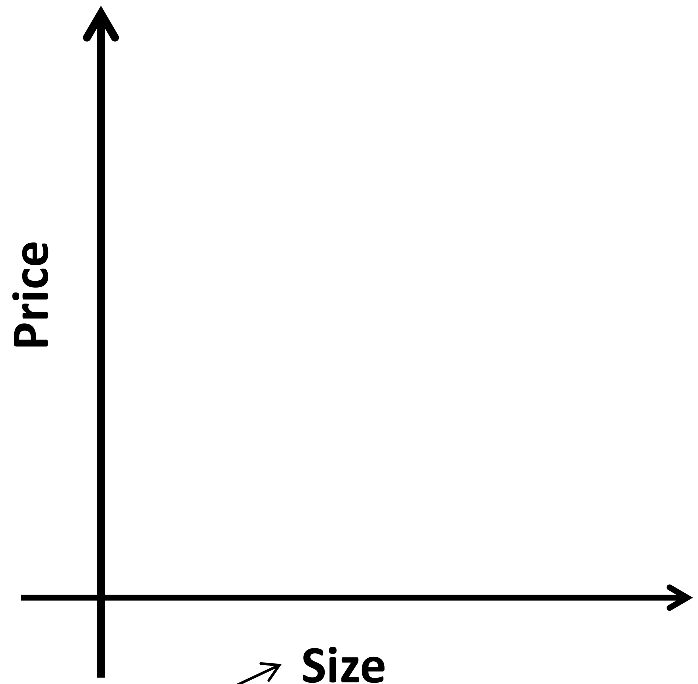
- Simple Regression
  - Only one feature or independent variable

- Multiple Regression
  - Multiple (More than 1) features or independent variables

Example – Simple Regression

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

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



Single Variable / feature

Example – Multiple Regression

- m = Number of Examples
- n = Number of features
- X = Feature Vector for training example
- $x_j^{(i)}$ = value of feature *j* in *i*th training example

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

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

Credit: Andrew Ng's Coursera course on Machine Learning

- Simple Regression

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

- Multiple Regression

$$\boxed{h_\theta(\mathbf{X}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ...}$$

x_1, x_2, x_3, x_4, … are features of a single example under consideration

- In general,

$$h_\theta(\mathbf{X}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n$$

- For convenience of notations, let x_0=1

$$h_\theta(\mathbf{X}) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n$$

$$\mathbf{X} = \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ x_n \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ \theta_n \end{bmatrix}$$

$$\boxed{h_\theta(\mathbf{X}) = \boldsymbol{\theta}^T \mathbf{X}}$$

n+1 column vectors

- Hypothesis

=1

$$h_\theta(\mathbf{X}) = \boldsymbol{\theta}^T\mathbf{X} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots.. + \theta_n x_n$$

n+1 dimensional vector

- Parameters

$$\theta_0, \theta_1, \theta_2, \ldots, \theta_n \qquad \boldsymbol{\theta}$$

n+1 dimensional vector

- Cost Function

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

- Goal

$$\min_{\boldsymbol{\theta}} imize \quad J(\boldsymbol{\theta})$$

- Gradient Descent

Repeat {

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

}

Simultaneous update for every $j = 0,1,2,\ldots,n$

Gradient Descent Algorithm

Simple Linear Regression

repeat until convergence {

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

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

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

}

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

Update θ_0 and θ_1 simultaneously

Multiple Linear Regression

repeat until convergence {

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

}

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

Simultaneously update θ_j for every $j = 0,1,2,\ldots,n$

Unit 2

Practical Tips for Linear Regression

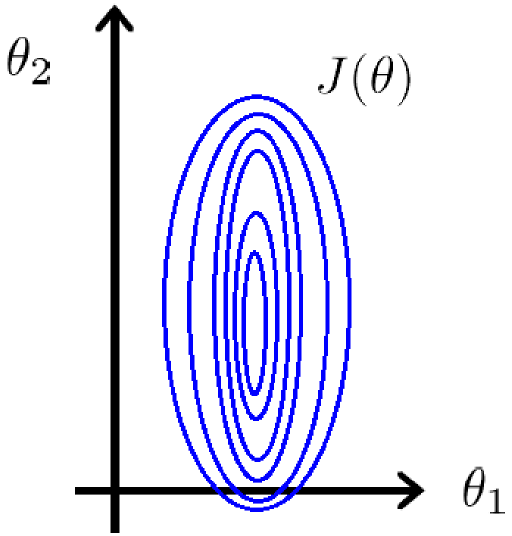
Machine Learning – Regression and Decision Trees

Some Practical Tips

- **Feature Scaling**
  - Features should be in similar scale (i.e. they should have similar range of values) → For quick convergence

E.g. x_1 = size (0 – 2000 feet^2)

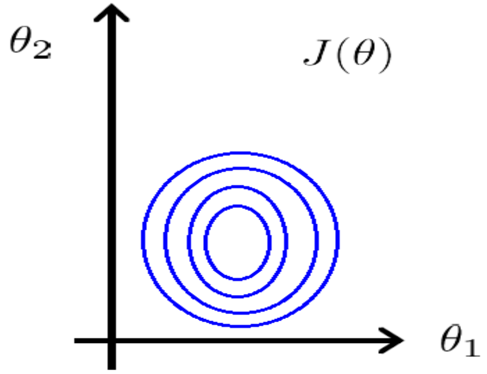
x_2 = number of bedrooms (1-5)



Method 1: Divide the feature value by maximum in the range

$$x_1 = \frac{size(feet^2)}{2000} \qquad 0 \le x_1 \le 1$$

$$x_2 = \frac{number\ of\ bedrooms}{5} \qquad 0 \le x_2 \le 1$$



Credit: Andrew Ng's Coursera course on Machine Learning

Some Practical Tips

- Generally, get every feature approximately in the range -1 $\leq x_i \leq$ 1

- Need not be exactly in the same range but at least closer to one another

Some Practical Tips

- **Feature Scaling**
  - **Method 2: Mean Normalization**

Replace x_i with $(x_i - \mu_i)$ → make the features to have approximately zero mean (Do not apply to $x_0 = 1$)

Average value of x_i in training set

E.g.

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

Assuming average size of house = 1000 feet^2

$$x_2 = \frac{number\ of\ bedrooms - 2}{5}$$

-0.5 ≤ x_1, x_2 ≤ 0.5 (approx.)

Generalization:

$$x_i = \frac{x_i - \mu_i}{s_i}$$

Average value of x_i in training set

Range of values i.e. $\max(x_i) - \min(x_i)$ in training set OR standard deviation

Some Practical Tips

- **Learning Rate (α)**

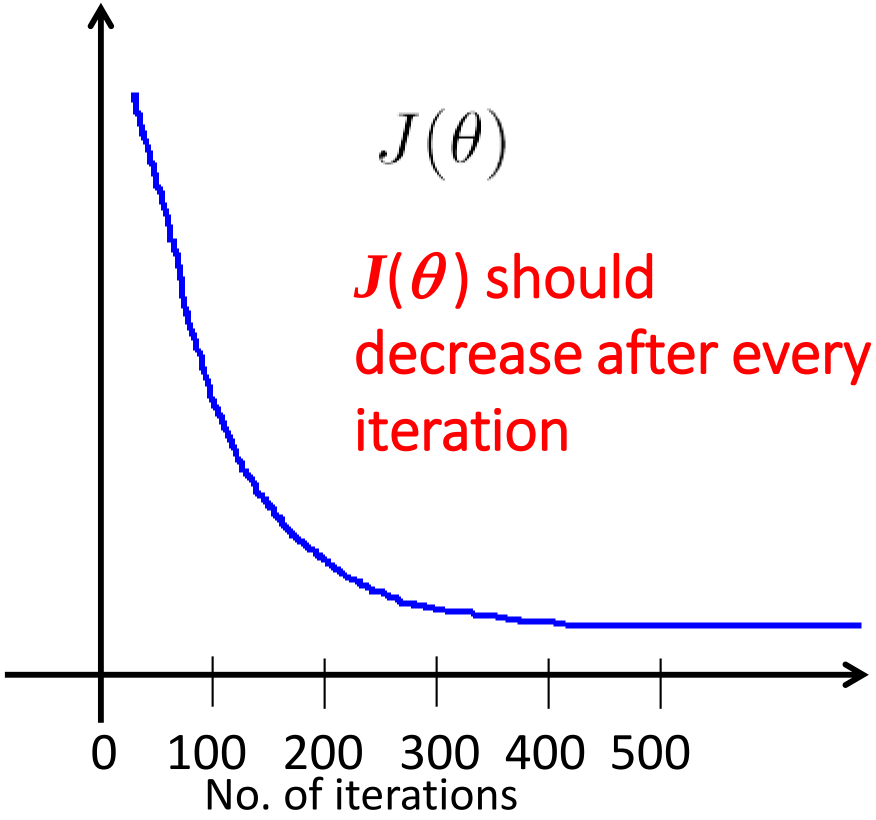
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

  - How to make sure that gradient descent is working correctly ?
  - How to choose learning rate ?

Some Practical Tips

- **How to make sure that gradient descent is working correctly ?**

$$\min_{\theta} J(\theta)$$



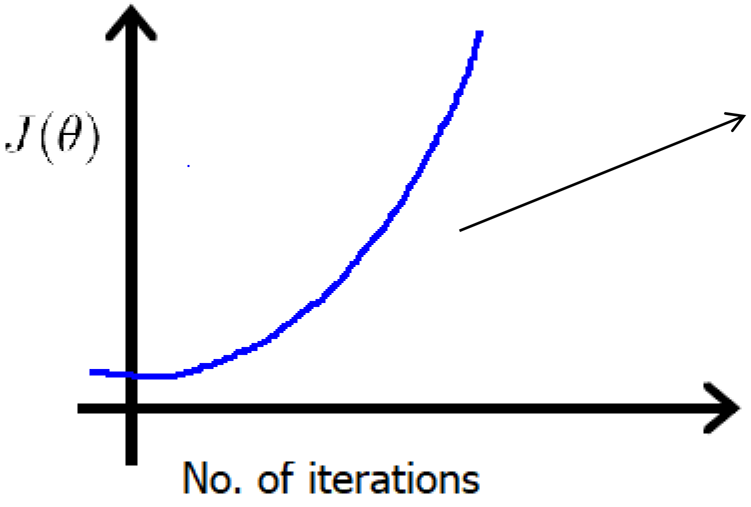
$J(\theta)$ should decrease after every iteration

Automatic Convergence Test

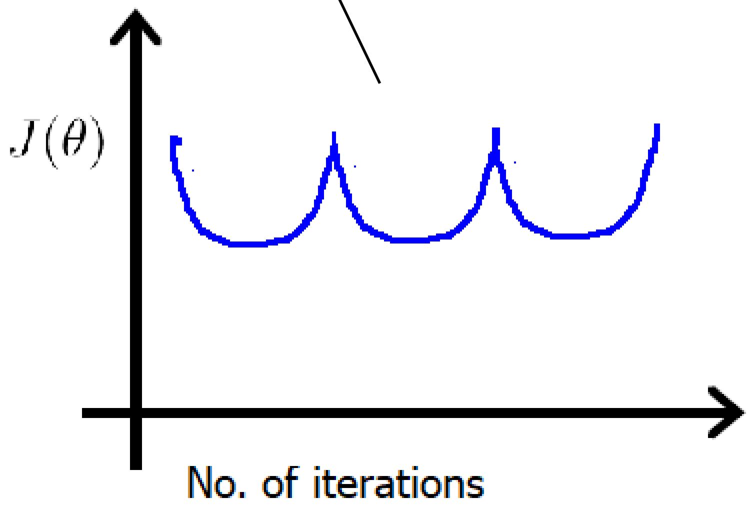
Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration. (ε)

Some Practical Tips

- **How to make sure that gradient descent is not 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

Some Practical Tips

- **Learning Rate (α) : Summary**

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

  - 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 the following

    0.001, 0.01, 0.1, ,1

Unit 2

Polynomial Regression, Normal Equation

Machine Learning – Regression and Decision Trees

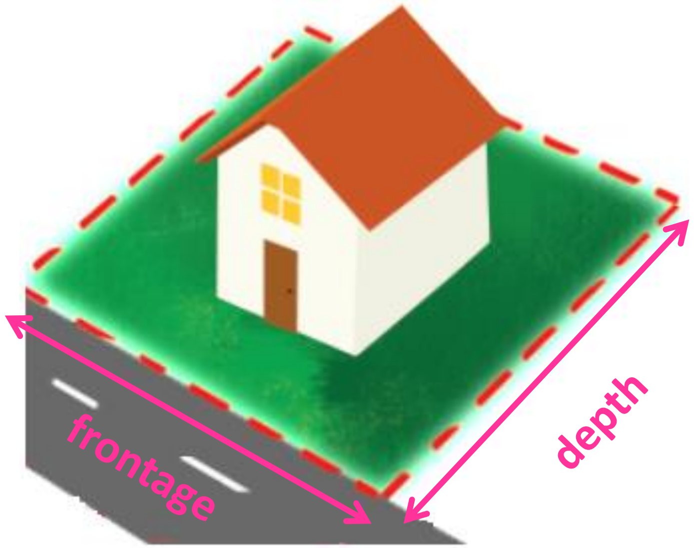
Creating our own features

- Example – Housing price prediction

$$h_\theta(\mathbf{X}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

frontage

depth



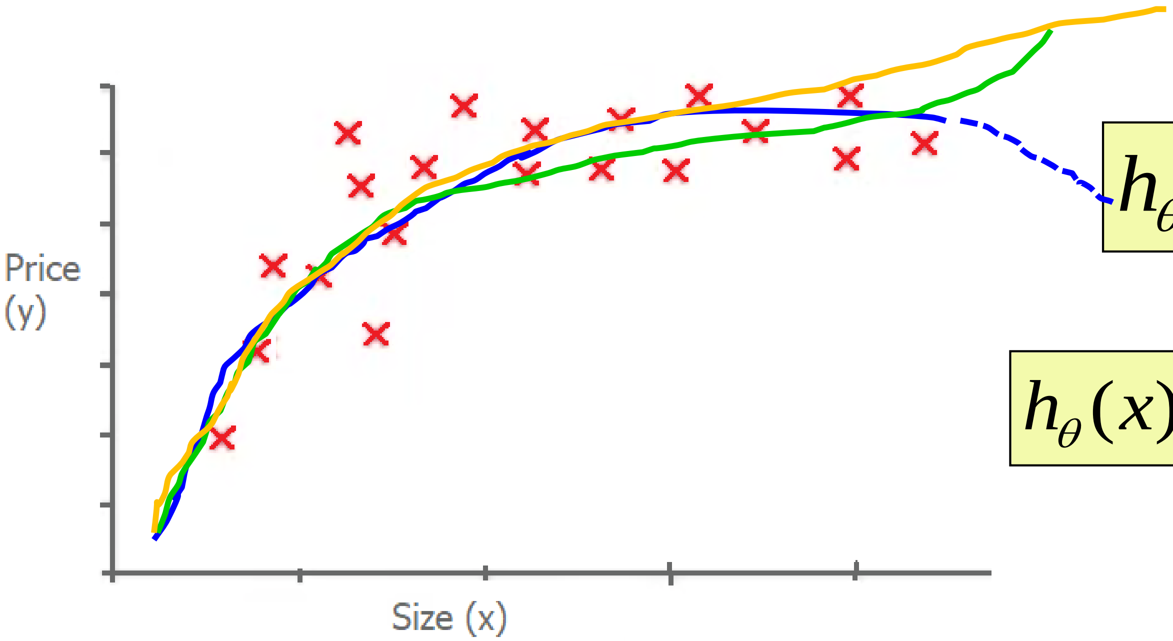
Create New feature of our own

Area = frontage * depth

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

Depending on the insight of the problem one may create new features from existing ones

Polynomial Regression



Quadratic model

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

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

Cubic model

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

$$= \theta_0 + \theta_1 (\underline{size}) + \theta_2 (\underline{size})^2 + \theta_3 (\underline{size})^3$$

Feature Scaling becomes important

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

Normal Equation

- Method to solve for θ analytically (Alternative for Gradient Descent)
- Example: Number of features, n = 4

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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 4 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

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

m x (n+1) matrix

m dimensional vector

Normal Equation

- Generalization: m examples are given as $(X^{(1)}, y^{(1)}), (X^{(2)}, y^{(2)}), \ldots. (X^{(m)}, y^{(m)})$ ----- n features per vector

$$\mathbf{X}^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ . \\ . \\ x_n^{(i)} \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} ------(\mathbf{X}^{(1)})^T ------ \\ ------(\mathbf{X}^{(2)})^T ------ \\ . \\ . \\ ------(\mathbf{X}^{(m)})^T ------ \end{bmatrix}$$

Design Matrix

m x (n+1) matrix

Example

$$\mathbf{X}^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ . & . \\ . & . \\ 1 & x_1^{(m)} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ . \\ . \\ y^{(m)} \end{bmatrix}$$

m x 2 matrix

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

m training examples, n features per example

Gradient Descent	Normal Equation
Need to choose α	No need to choose α
Needs many iterations	Do not need to iterate
Feature scaling required	No feature scaling
Works well even when n is large (Typically, if $n > 10^5$ use gradient descent)	Need to compute $(X^TX)^{-1}$ If X is of size (n x n), Computation time approx. $O(n^3)$ → Slow if n is large

Unit 2

Logistic Regression

Machine Learning – Regression and Decision Trees

Classification

- Problem of Classification
  - Email: Spam / Not Spam?
  - Online Transactions: Fraudulent (Yes / No)?
  - Tumor: Malignant / Benign ?

$y \in \{0, 1\}$

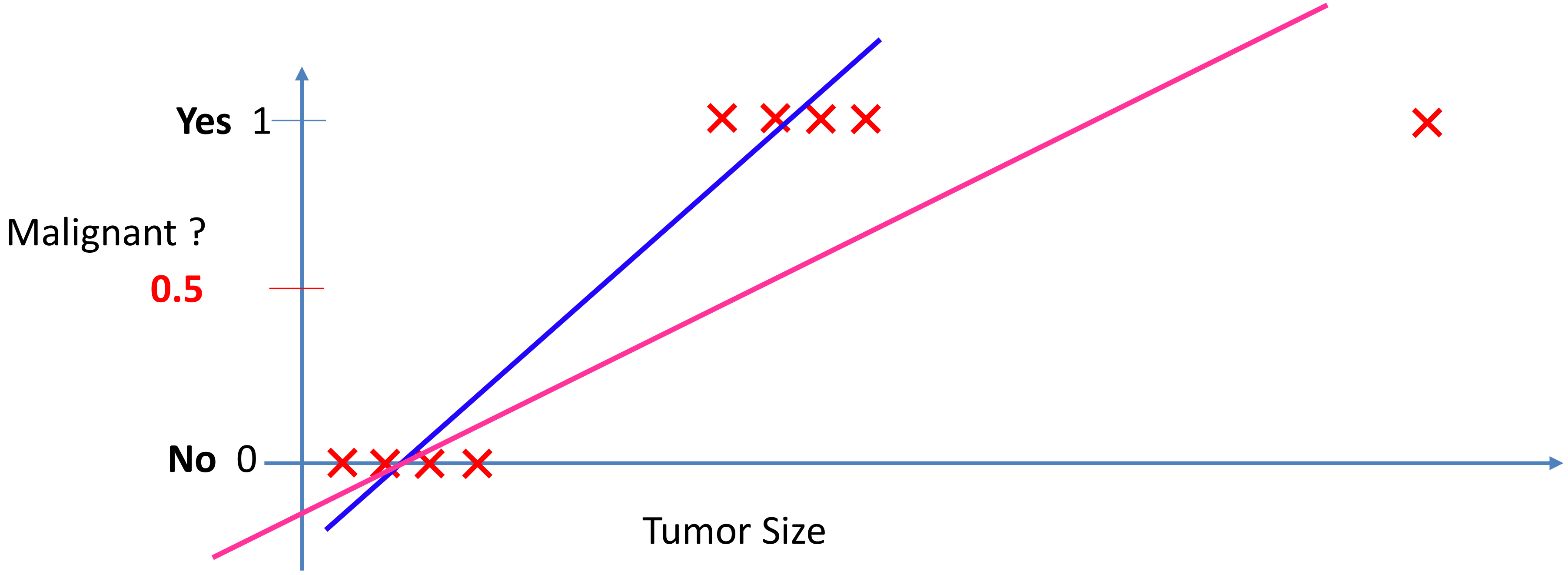
0: Negative Class (Benign)
1: Positive Class (Malignant)

Binary classification

$y \in \{0, 1, 2, 3\}$

Multiclass classification

How to develop a classification algorithm?



$$\boxed{h_\theta(\mathbf{X}) = \boldsymbol{\theta}^T \mathbf{X}}$$

Threshold classifier output $h_\theta(x)$ at 0.5 :
If $h_\theta(x) \geq 0.5$, predict "y = 1"
If $h_\theta(x) < 0.5$, predict "y = 0"

Applying Linear Regression to Classification Problem is not a good idea !

Logistic Regression

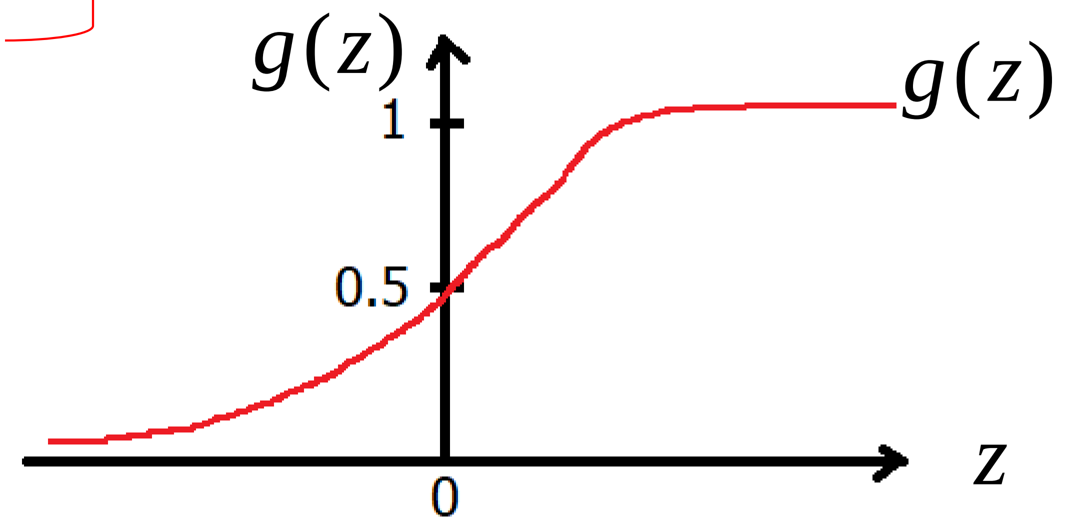
- Find hypothesis such as $0 \le h_\theta(x) \le 1$

$$h_\theta(\mathbf{X}) = g(\theta^T \mathbf{X})$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$$h_\theta(\mathbf{X}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \mathbf{X}}}$$

Sigmoid function /
Logistic function



Logistic Regression

- How to interpret hypothesis ?
  - $h_\theta(x)$ = estimated probability that y = 1 on input x
  - Example

$$\mathbf{X} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ tumorSize \end{bmatrix}$$

  - $h_\theta(x)$ = 0.8 ➔ probability that y = 1 is 0.8
    ➔ 80 % chance to have tumor as malignant

$$\boxed{h_\theta(\mathbf{X}) = P(y=1 \mid \mathbf{X};\theta)}$$

"probability that y = 1, given X, parameterized by θ"

$$\boxed{P(y=0 \mid \mathbf{X};\theta) + P(y=1 \mid \mathbf{X};\theta) = 1}$$

$$\boxed{P(y=0 \mid \mathbf{X};\theta) = 1 - P(y=1 \mid \mathbf{X};\theta)}$$

Unit 2

Logistic Regression – Decision Boundary, Cost Function

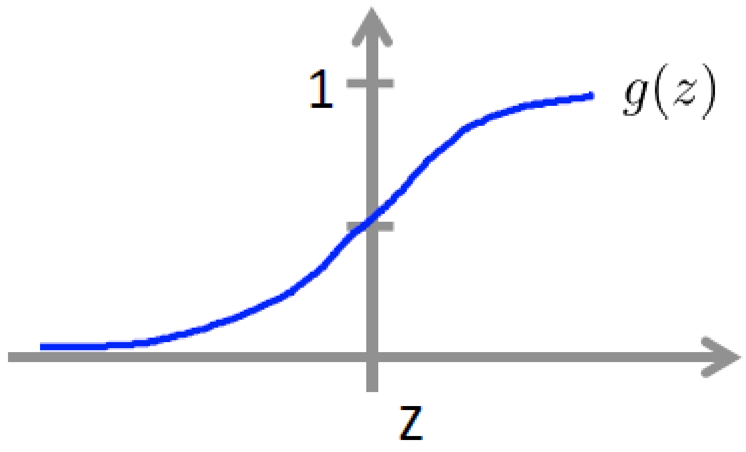
Machine Learning – Regression and Decision Trees

Decision Boundary

- Logistic Regression

$$h_\theta(\mathbf{X}) = g(\theta^T \mathbf{X}) \quad = P(y=1 \mid \mathbf{X};\theta)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



Predict y = "1" if $h_\theta(x) \geq 0.5$

whenever $\theta^T X \geq 0$

If $z \geq 0$, $g(z) \geq 0.5$

➔ $h_\theta(x) = g(\theta^T X) \geq 0.5$

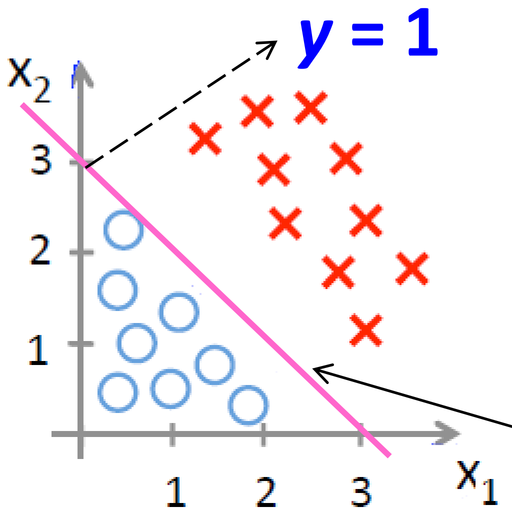
whenever $\theta^T X \geq 0$

Predict y = "0" if $h_\theta(x) < 0.5$

whenever $\theta^T X < 0$

If $z < 0$, $g(z) < 0.5$

Decision Boundary



Decision Boundary

$x_1 + x_2 = 3$

$$h_\theta(\mathbf{X}) = g(\underbrace{\theta_0}_{=-3} + \underbrace{\theta_1}_{=1} x_1 + \underbrace{\theta_2}_{=1} x_2)$$

$$\boldsymbol{\theta} = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$h_\theta(\mathbf{X}) = g(-3 + x_1 + x_2)$$

Predict y = "1" if $h_\theta(x) \geq 0.5$

whenever $\theta^T X \geq 0$

Predict y = "1" if $h_\Theta(x) \geq 0.5$ ➔ if -3 + x_1 + $x_2 \geq 0$

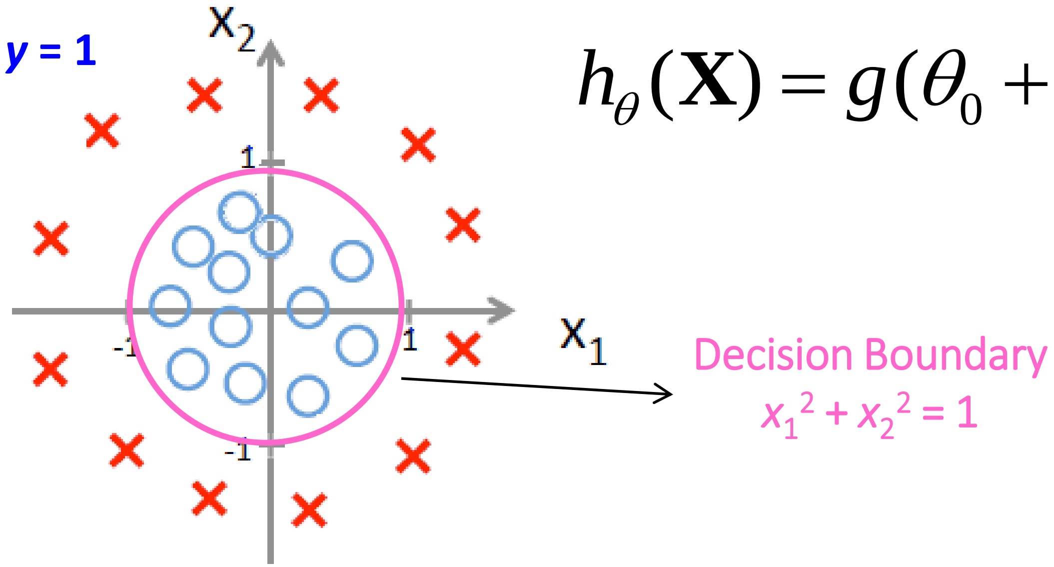
➔ $x_1 + x_2 \geq 3$

Predict y = "0" if $h_\Theta(x) < 0.5$ ➔ if -3 + x_1 + $x_2 < 0$

➔ $x_1 + x_2 < 3$

Slide from Andrew Ng's Coursera course on Machine Learning

Non-linear decision boundaries

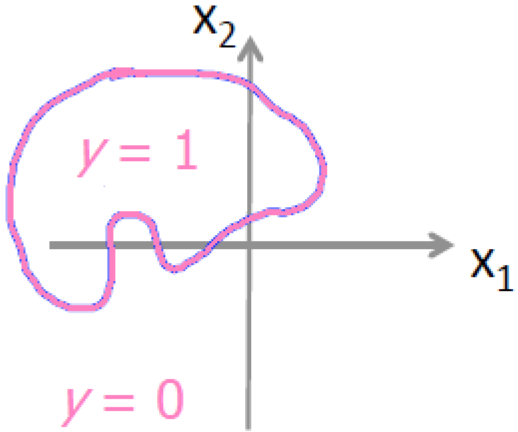


$$h_\theta(\mathbf{X}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\boldsymbol{\theta} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict y = "1" if $h_\Theta(x) \geq 0.5$ ➔ if $-1 + x_1^2 + x_2^2 \geq 0$

➔ $x_1^2 + x_2^2 \geq 1$



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \ldots)$$

Slide from Andrew Ng's Coursera course on Machine Learning

Logistic Regression – Cost Function

Machine Learning – Regression and Decision Trees

Logistic Regression – Cost Function

- Training Set $\{(X^{(1)}, y^{(1)}), (X^{(2)}, y^{(2)}), \ldots. (X^{(m)}, y^{(m)})\}$ ---- m examples, n features each

$$\mathbf{X} = \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ x_n \end{bmatrix}$$

$x_0 = 1, y \in \{0, 1\}$

How to choose parameters θ ?

- Hypothesis

$$\boxed{h_\theta(\mathbf{X}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{X}}}}$$

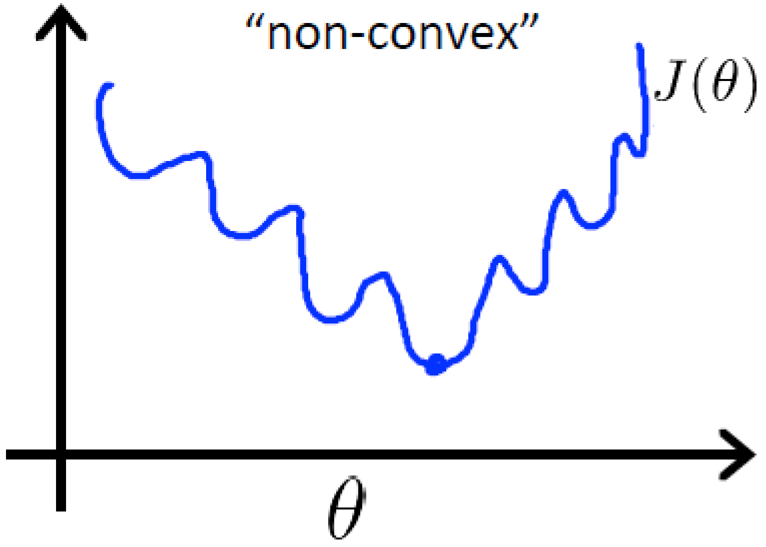
Cost Function

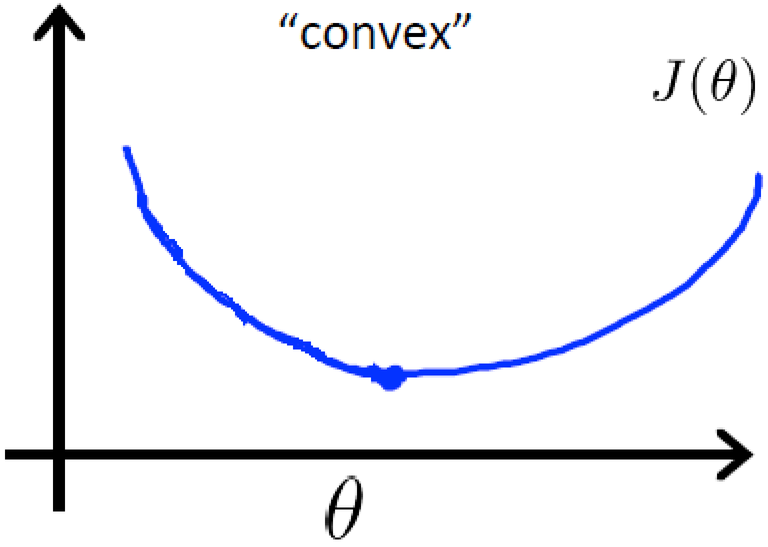
- Linear Regression

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

$$Cost(h_\theta(\mathbf{X}^{(i)}), y^{(i)})$$

$$h_\theta(\mathbf{X}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{X}}}$$

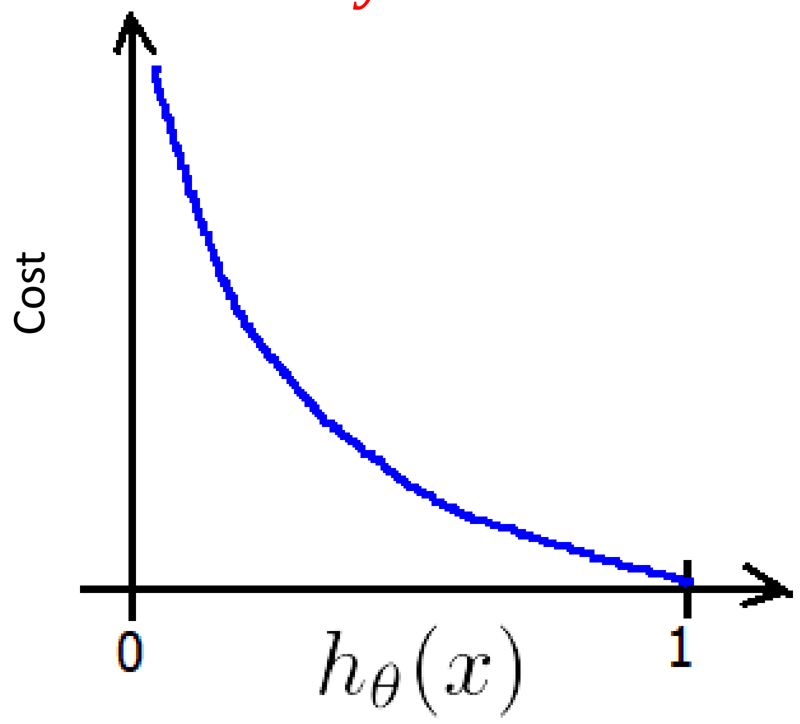




Logistic Regression – Cost Function

$$Cost(h_\theta(\mathbf{X}), y) = \begin{cases} -\log(h_\theta(\mathbf{X})) & \text{if } \; y = 1 \\ -\log(1 - h_\theta(\mathbf{X})) & \text{if } \; y = 0 \end{cases}$$

If $y = 1$



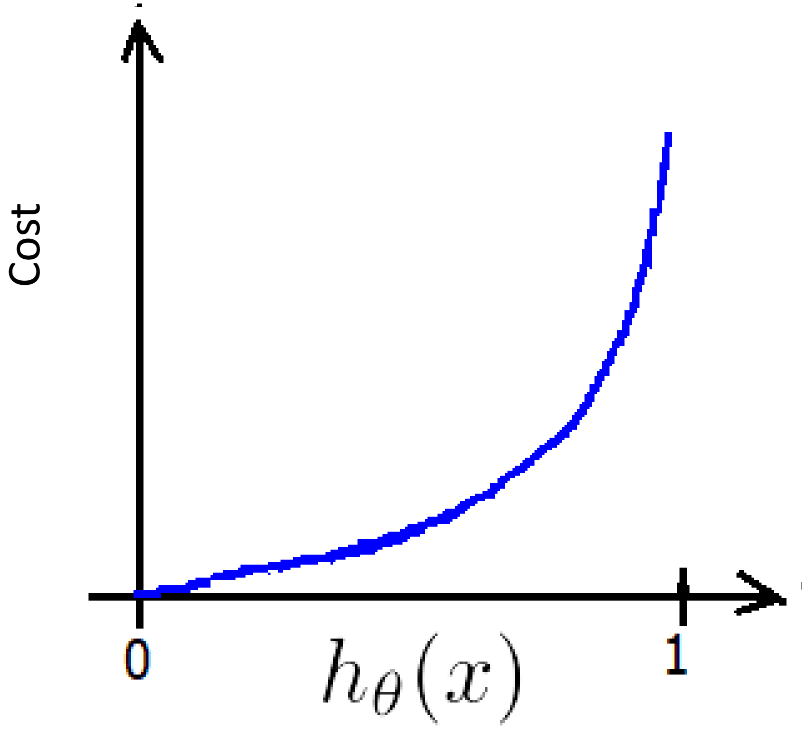
Cost = 0 if y = 1 and $h_\Theta(x)$ = 1

But as $h_\Theta(x) \rightarrow 0$, Cost $\rightarrow \infty$

Logistic Regression – Cost Function

$$Cost(h_\theta(\mathbf{X}) - y) = \begin{cases} -\log(h_\theta(\mathbf{X})) & \text{if } y = 1 \\ -\log(1 - h_\theta(\mathbf{X})) & \text{if } y = 0 \end{cases}$$

If $y = 0$



Cost = 0 if y = 0 and $h_\Theta(x)$ = 0

But as $h_\Theta(x)$ → 1, Cost → ∞

Logistic Regression – Simplified Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(\mathbf{X}^{(i)}), y^{(i)})$$

$$Cost(h_\theta(\mathbf{X}), y) = \begin{cases} -\log(h_\theta(\mathbf{X})) & \text{if } y = 1 \\ -\log(1 - h_\theta(\mathbf{X})) & \text{if } y = 0 \end{cases}$$

Simplified Cost Function

$$Cost(h_\theta(\mathbf{X}), y) = -y\log(h_\theta(\mathbf{X})) - (1-y)\log(1-h_\theta(\mathbf{X}))$$

Logistic Regression – Simplified Cost Function

$$Cost(h_\theta(\mathbf{X}), y) = -y\log(h_\theta(\mathbf{X})) - (1-y)\log(1-h_\theta(\mathbf{X}))$$

This is for single example / single sample

For m examples / m samples

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(\mathbf{X}^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log(h_\theta(\mathbf{X}^{(i)})) + (1-y^{(i)})\log(1-h_\theta(\mathbf{X}^{(i)}))\right]$$

Find parameters θ such that,

$$\min_\theta J(\theta)$$

Predict $h_\theta(x)$ for new x

$$\boxed{h_\theta(\mathbf{X}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{X}}}}$$

$$P(y=1|\mathbf{x};\theta)$$

Logistic Regression and Gradient Descent

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(\mathbf{X}^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log(h_\theta(\mathbf{X}^{(i)})) + (1-y^{(i)})\log(1-h_\theta(\mathbf{X}^{(i)}))\right]$$

Find parameters θ such that, $\min_{\theta} J(\theta)$

Gradient Descent

$$\boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(\mathbf{X}^{(i)}) - y^{(i)})x_j^{(i)}}$$

Repeat {

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

}

Simultaneous update all θ_j

Logistic Regression and Gradient Descent

Observation:

For Linear Regression,

$$\boxed{h_\theta(\mathbf{X}) = \boldsymbol{\theta}^T \mathbf{X}}$$

For Logistic Regression,

$$\boxed{h_\theta(\mathbf{X}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T \mathbf{X}}}}$$

$$- y^{(i)})\log(1-h_\theta(\mathbf{X}^{(i)}))\Big]$$

$$\underset{\theta}{\text{in}}\ J(\theta)$$

Gradient Descent

Whether it is identical to linear regression ?

Repeat {

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

}

Simultaneous update all θ_j

Logistic Regression for Multi-class Classification

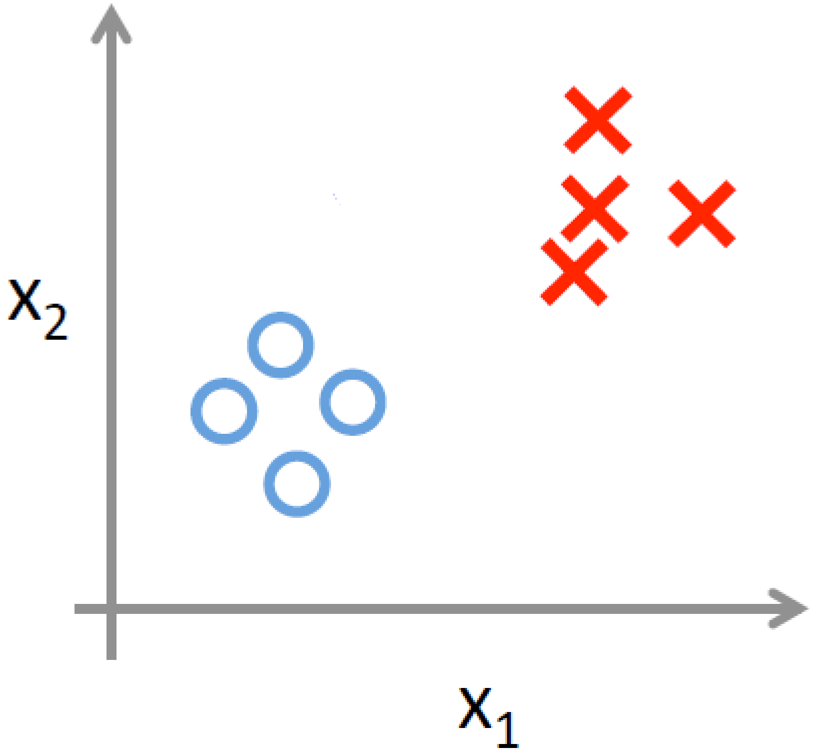
- Categorize emails as:

  Work, Friends, Family

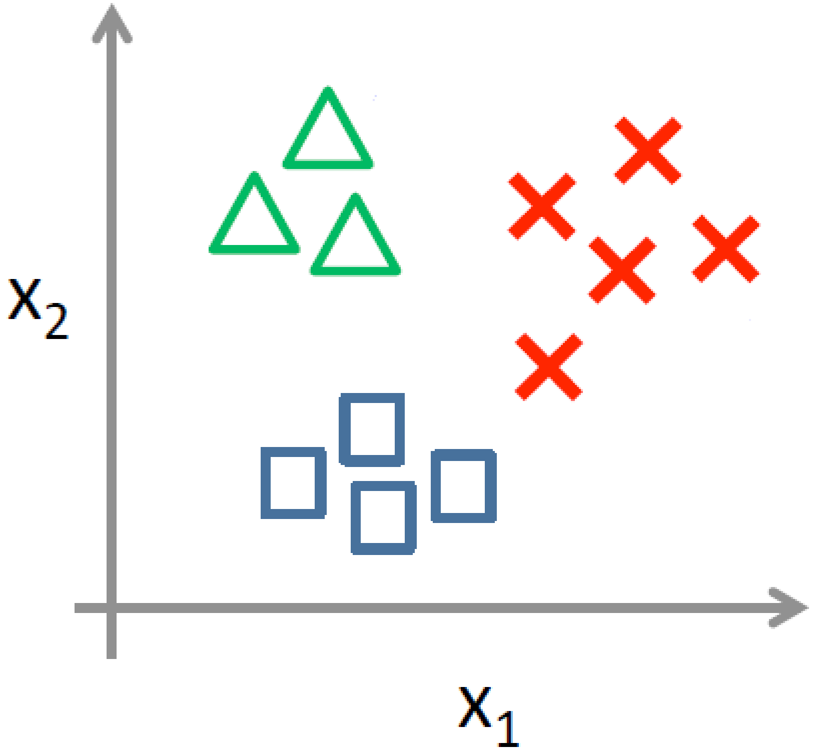
- Categorize Weather as:

  Sunny, Cloudy, Rainy, Snow

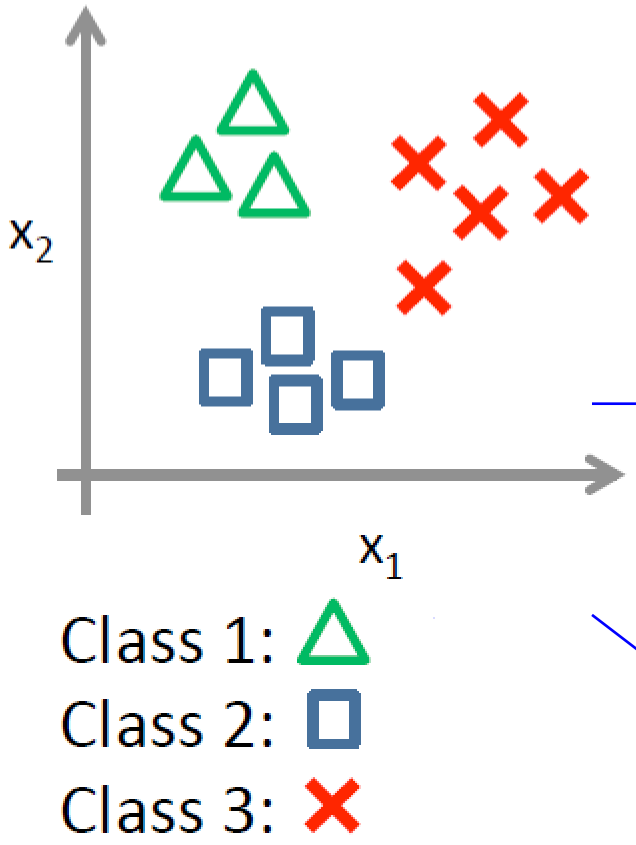
Binary Classification



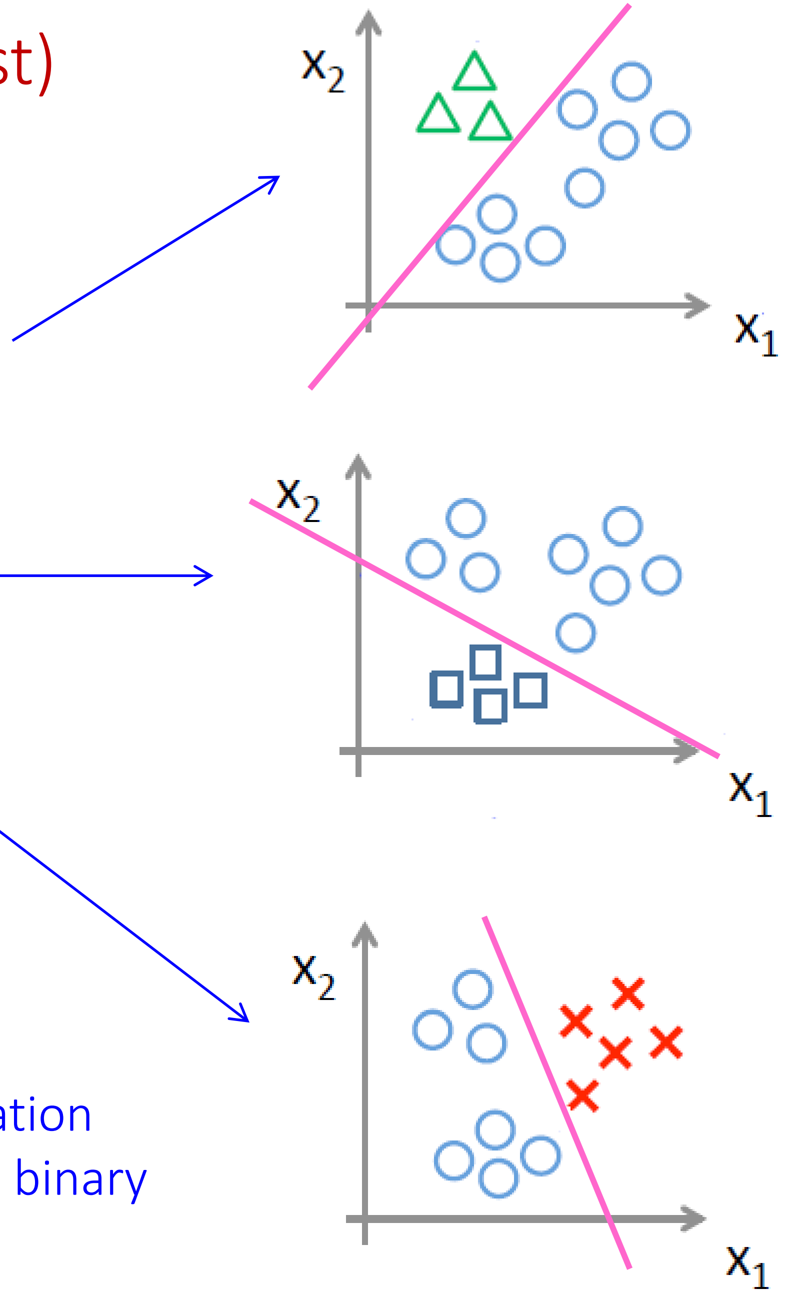
Multi-class Classification



One-vs-all (one-vs-rest)



Convert multi-class classification problem into three separate binary class classification problems



$h_\theta^{(1)}(\mathbf{X})$

$h_\theta^{(2)}(\mathbf{X})$

$h_\theta^{(3)}(\mathbf{X})$

$$h_\theta^{(i)}(\mathbf{X}) = P(y = i \mid \mathbf{X}; \theta) \qquad i = 1,2,3$$

One-vs-all (one-vs-rest)

- Train a logistic regression classifier $h_\theta^{(i)}(\mathbf{X})$ for each class *i* to predict the probability that *y* = *i*

- On a new input x, to make a prediction, pick the class *i* that maximizes

$$\max_i h_\theta^{(i)}(\mathbf{X})$$

Slide from Andrew Ng's Coursera course on Machine Learning

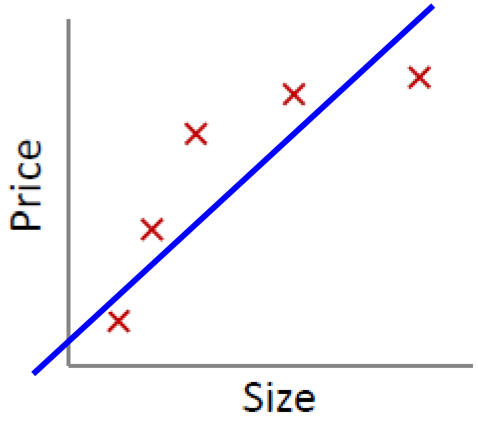
Unit 2

The Problem of Overfitting

Machine Learning – Regression and Decision Trees

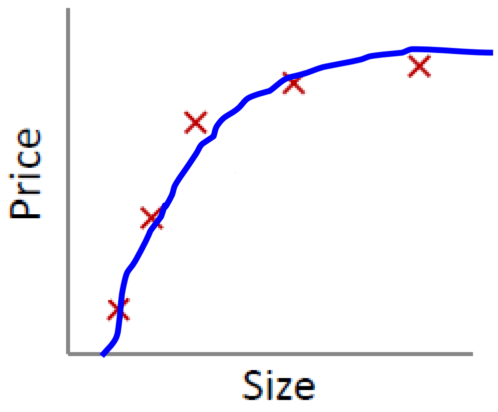
The Problem of Overfitting

Example: Linear Regression (Housing Price)



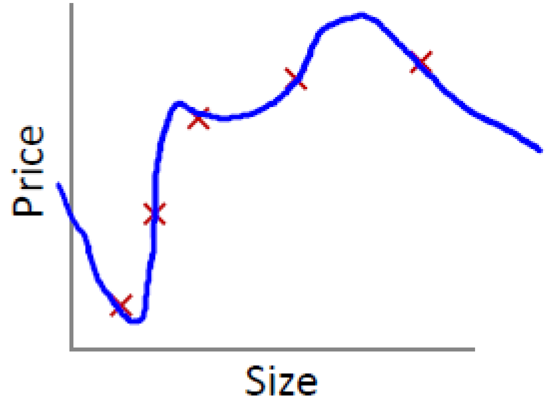
$$h_\theta(x) = \theta_0 + \theta_1 x$$

Under fitting



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

Right fitting



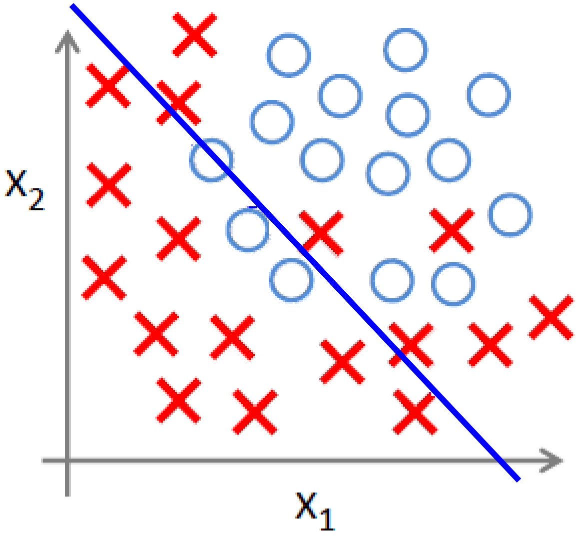
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Overfitting

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well. i.e. $J(\boldsymbol{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(\mathbf{X}^{(i)}) - y^{(i)})^2 \approx 0$, but fail to generalize to new examples

The problem of Overfitting

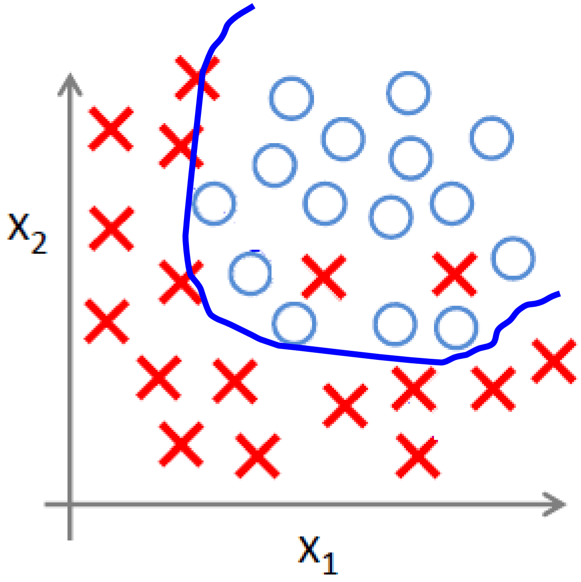
Example: Logistic Regression



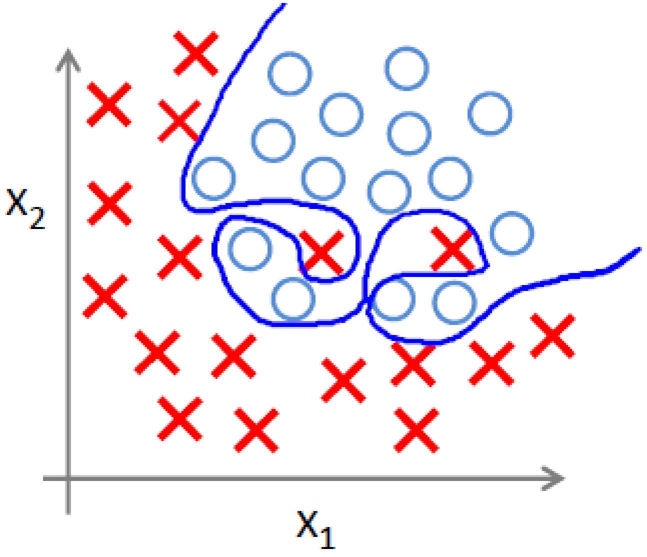
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Sigmoid function

Under fitting



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \ldots)$$

Overfitting

How to address overfitting ?

x_1 = size of house

x_2 = no. of bedrooms

x_3 = no. of floors

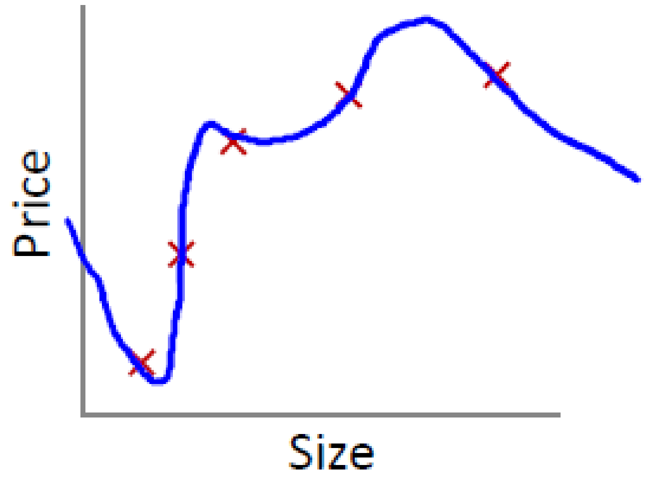
x_4 = age of house

x_5 = average income in neighborhood

x_6 = kitchen size

$\vdots$

x_{100}

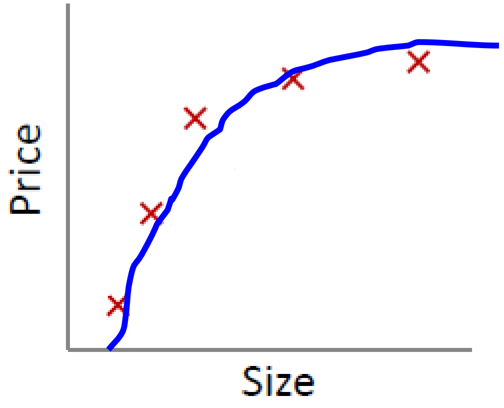


If we have a lot of features, and, very little training data, then, overfitting can become a problem

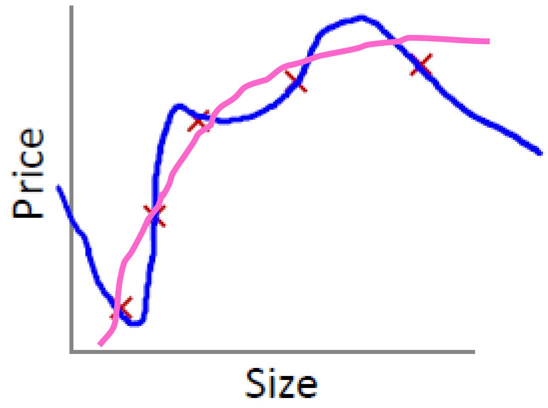
How to address overfitting ?

1. Reduce number of features
   - Manual selection of features
   - Feature selection algorithm

2. Regularization
   - Regularisation is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting.
   - Keep all the features but reduce magnitude / values of parameter θ_j
   - Works well when we have a lot of features, each of which contributes a bit to predicting



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



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Make Θ_3, Θ_4 very small

Make $\Theta_3 \approx 0$, $\Theta_4 \approx 0$

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{X}_i) - y_i)^2 \qquad \min_\theta J(\theta)$$

Regularization

- Small values for parameters $\theta_1, \theta_2, \ldots, \theta_m$
  - Simpler hypothesis
  - Less prone to overfitting

- Example : (x_0, x_1, x_2, x_3, …, x_{100}) and

$$(\theta_1, \theta_2, \ldots, \theta_{100})$$

Regularization term

$$J(\boldsymbol{\theta}) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(\mathbf{X}^{(i)}) - y^{(i)})^2 + \boxed{\lambda \sum_{j=1}^{n} \theta_j^{\ 2}}]$$

$$\min_{\theta} J(\theta)$$

Regularization parameter

Regularized Linear Regression

$$J(\boldsymbol{\theta}) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(\mathbf{X}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^{\ 2}] \qquad \min_\theta J(\theta)$$

- Gradient Descent

Repeat {

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

$$\theta_j := \theta_j - \alpha[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(\mathbf{X}^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j]$$

}

j = 1,2,3,…n

$$\theta_j := \theta_j \underbrace{\boxed{(1 - \alpha \frac{\lambda}{m})}}_{<1} - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(\mathbf{X}^{(i)}) - y^{(i)})x_j^{(i)}$$