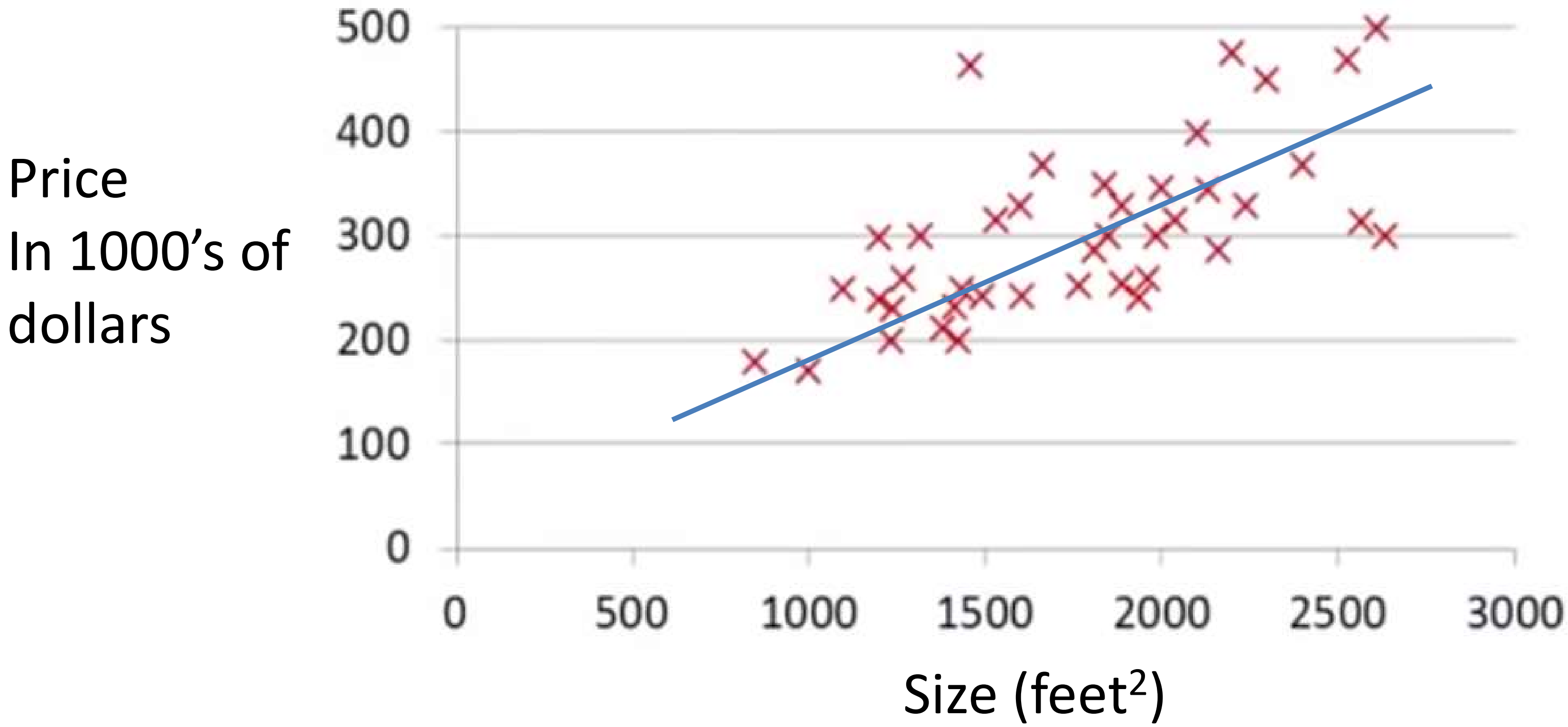
Supervised Learning

Some slides were adapted/taken from various sources, including Prof. Andrew Ng's Coursera Lectures, Stanford University, Prof. Kilian Q. Weinberger's lectures on Machine Learning, Cornell University, Prof. Sudeshna Sarkar's Lecture on Machine Learning, IIT Kharagpur, Prof. Bing Liu's lecture, University of Illinois at Chicago (UIC), CS231n: Convolutional Neural Networks for Visual Recognition lectures, Stanford University and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and **NOT** to distribute it.

Supervised Learning



- Given the right answer for each example of the data
  - Classification: discrete no. of outputs
  - Regression: Predict real valued data

Supervised Learning

Training set of housing prices

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

Notation:

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

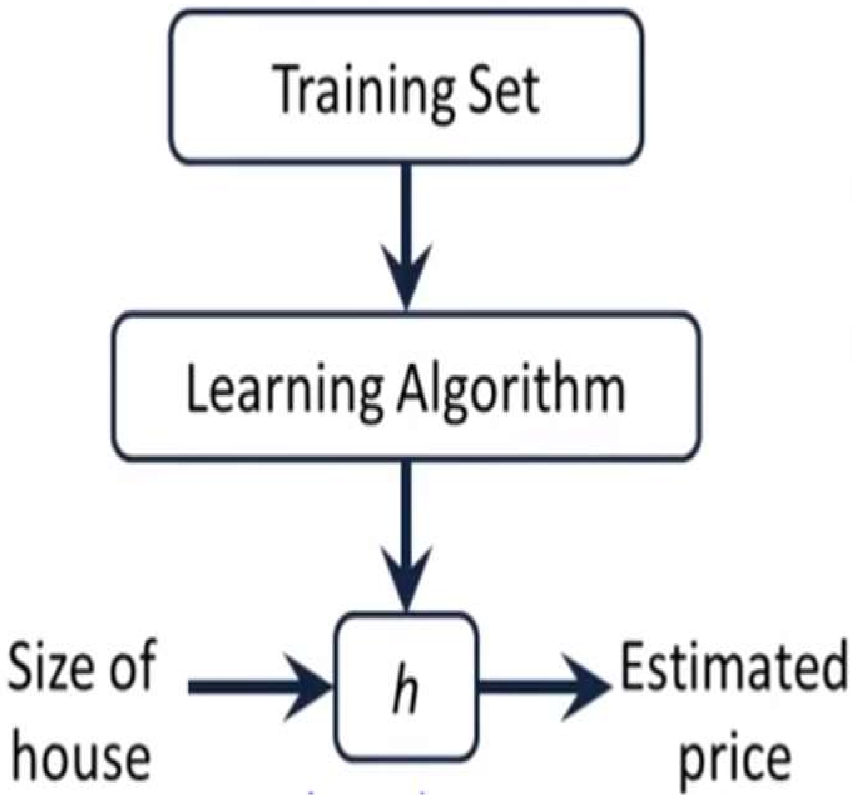
(x,y) → one training example

$(x^{(i)},y^{(i)})$ → i^{th} training example

$x^{(i)} = 2104$

$y^{(i)} = 460$

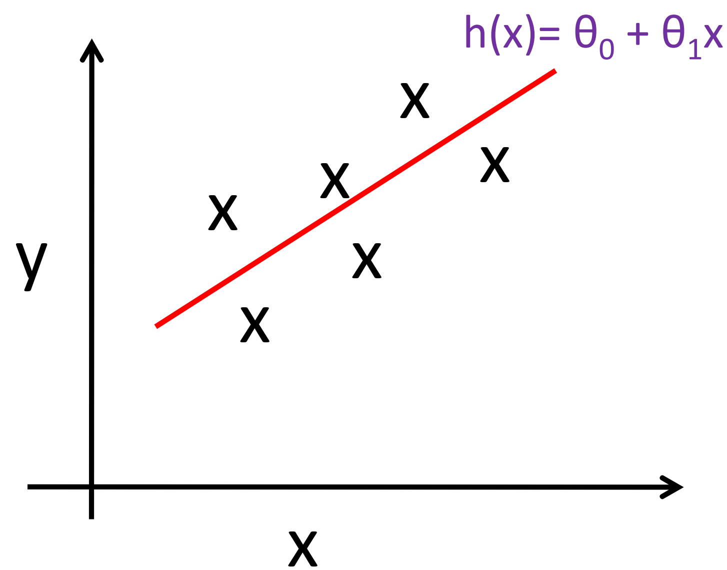
Supervised Learning



x   hypothesis   Estimated value of y

How do we represent h

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



Univariate linear regression: linear regression with one variable

Cost Function

Training Set

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

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

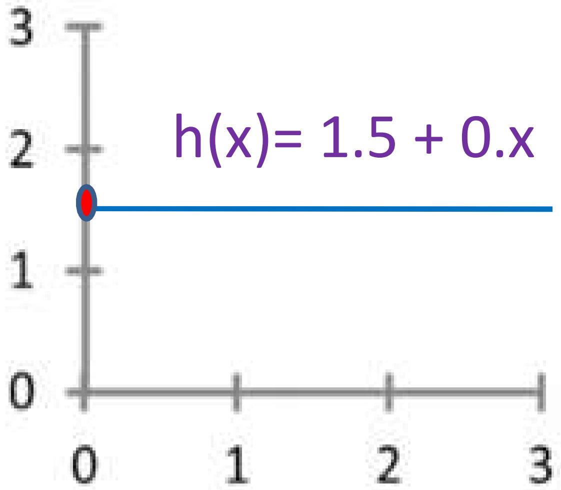
θ_1's → Parameters

How to choose θ_1's

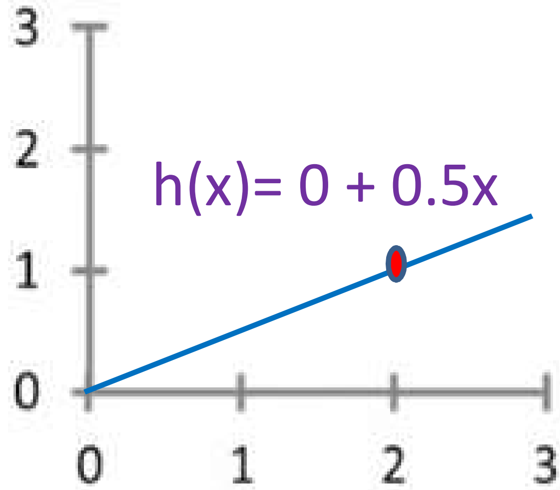
Cost Function

Hypothesis Function:

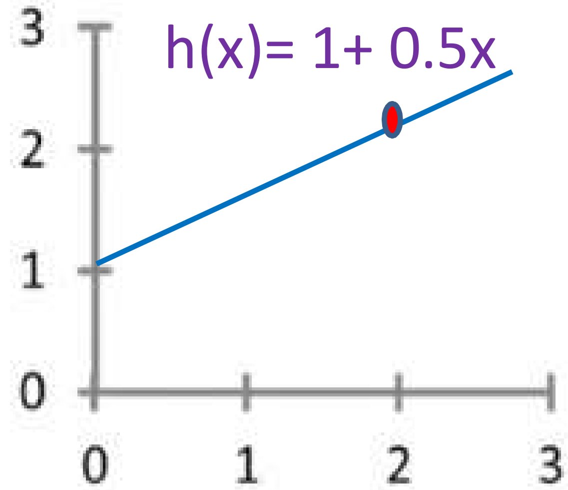
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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

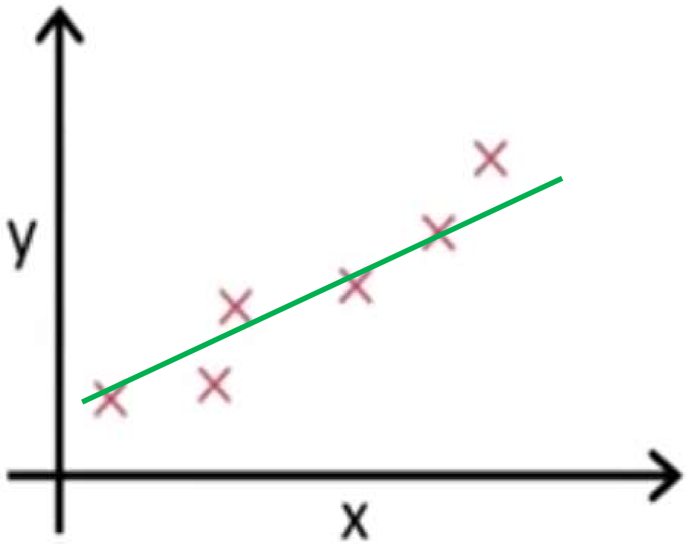


$\theta_0 = 0$
$\theta_1 = 0.5$



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

Cost Function



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

Parameters: θ_0, θ_1

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

Squared error function

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

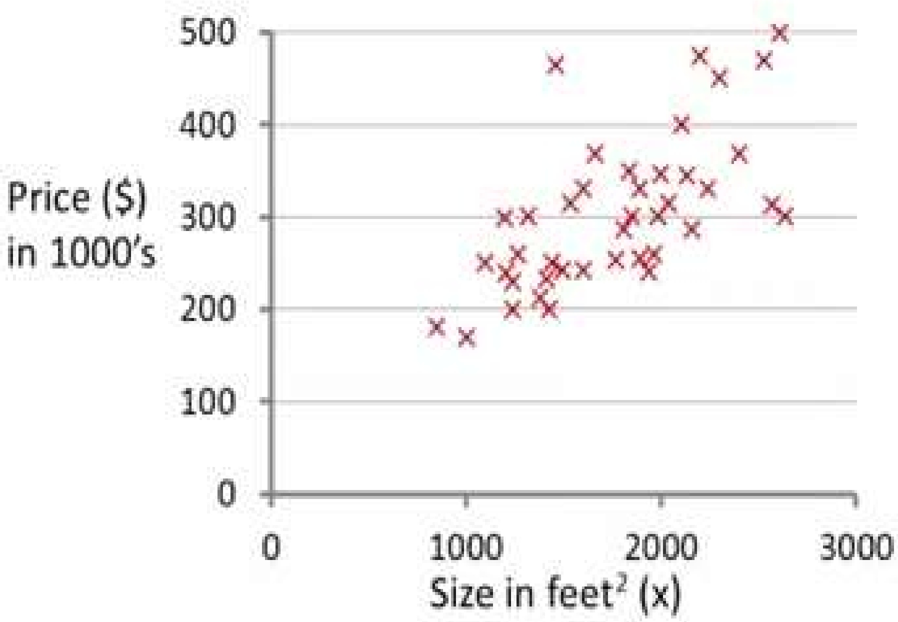
Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

m = No. of training samples

Cost Function

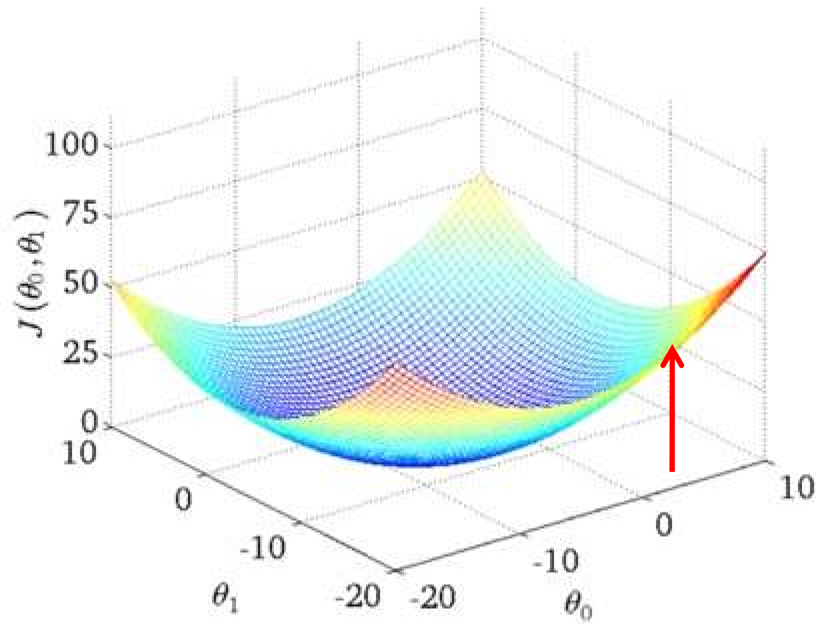
$h_\theta(x)$

(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)

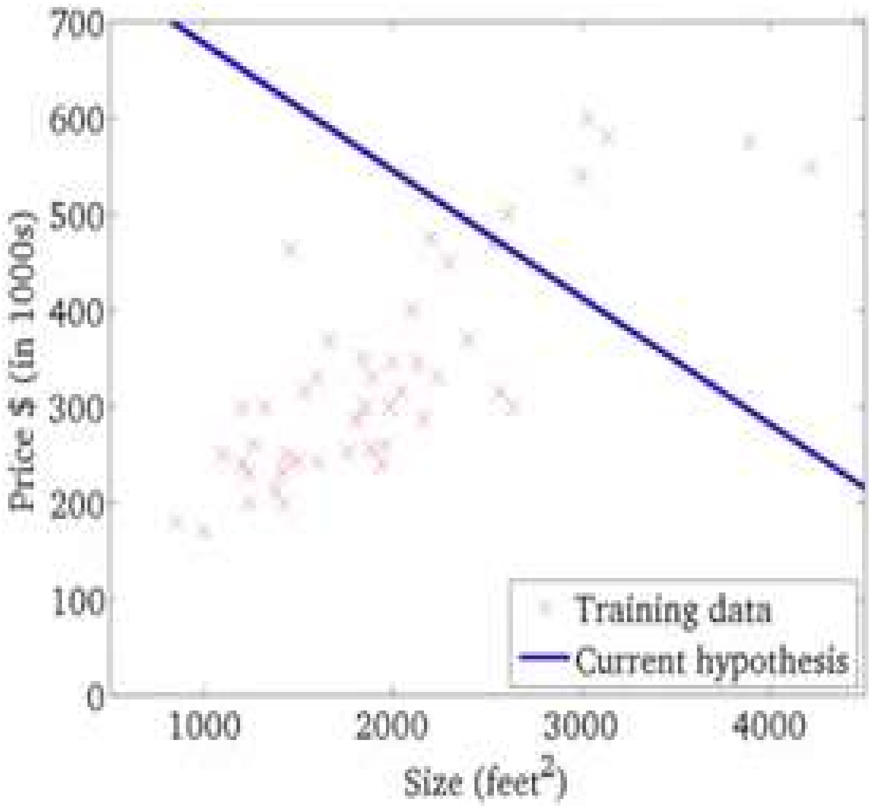


$J(\theta_0, \theta_1)$= value of the height of the surface

Contour Plots / Figures

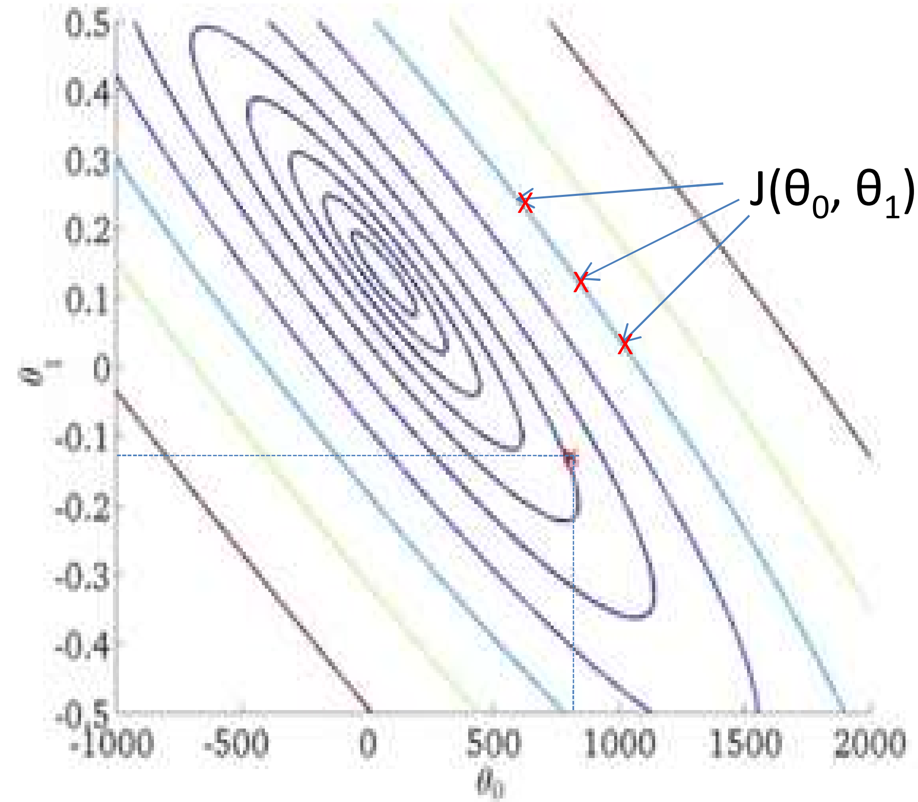
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

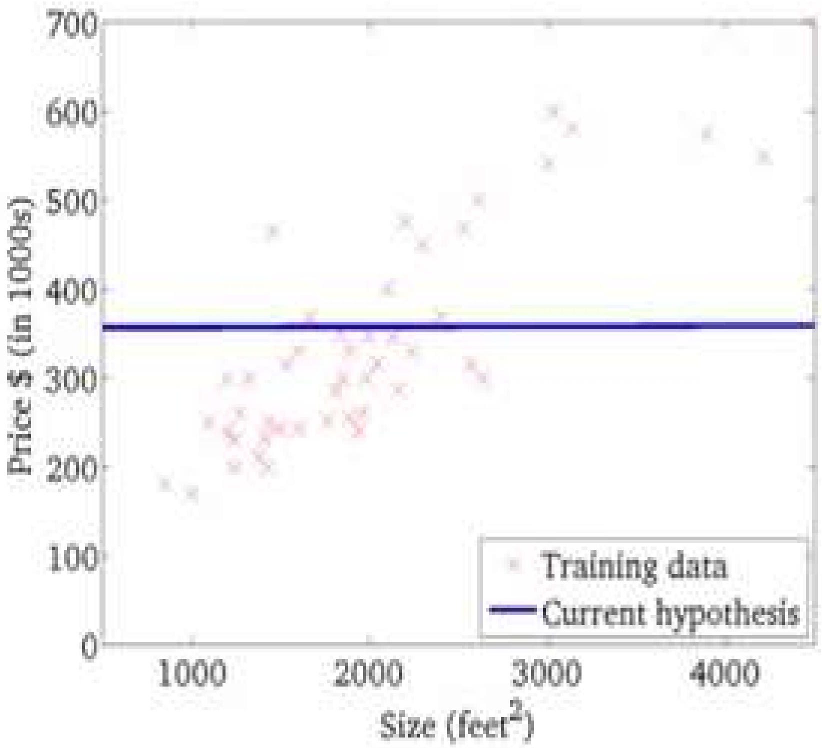
(function of the parameters θ_0, θ_1)



$(\theta_0, \theta_1) = (800, -0.125)$

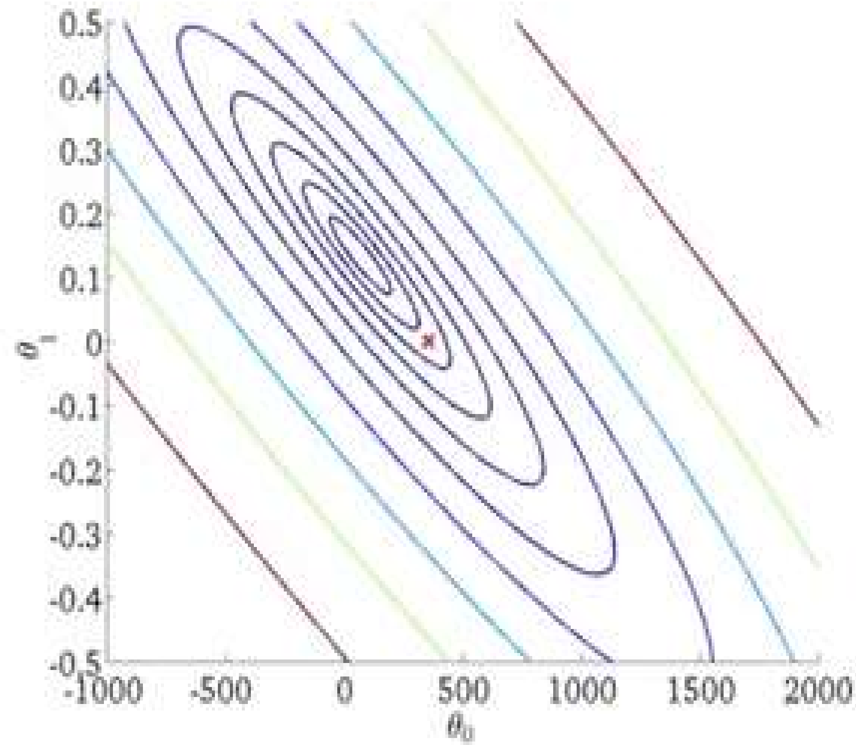
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

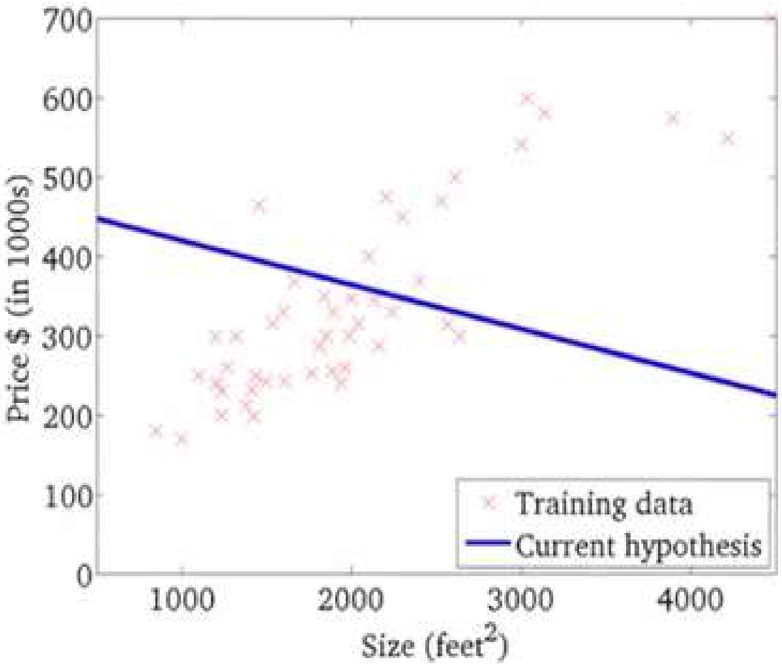
(function of the parameters θ_0, θ_1)



$(\theta_0, \theta_1) = (360, 0)$

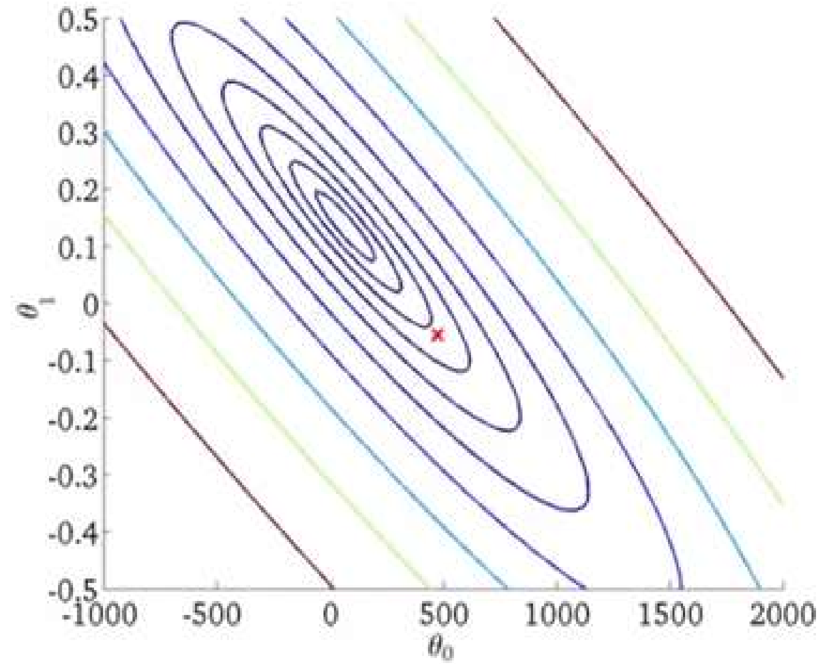
$h_\theta(x)$

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



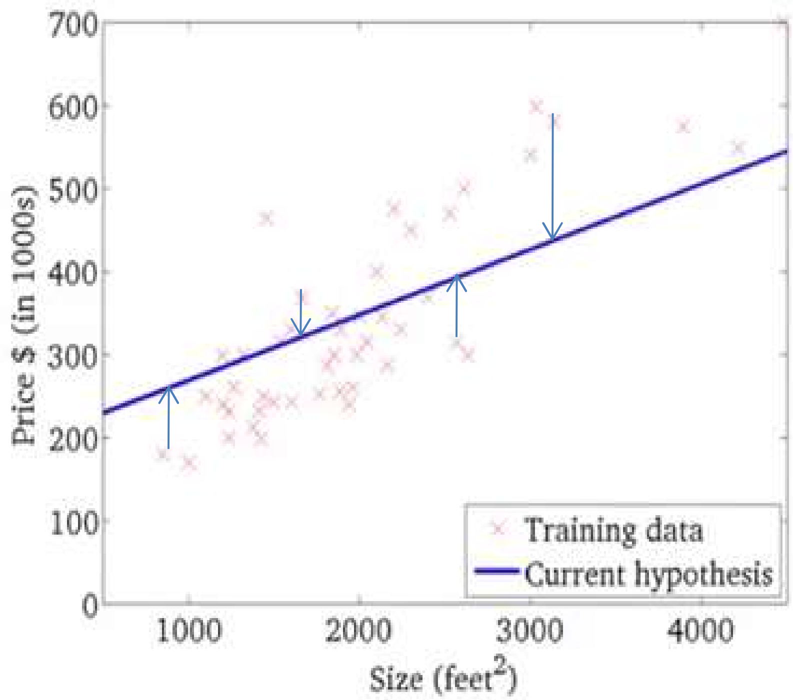
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



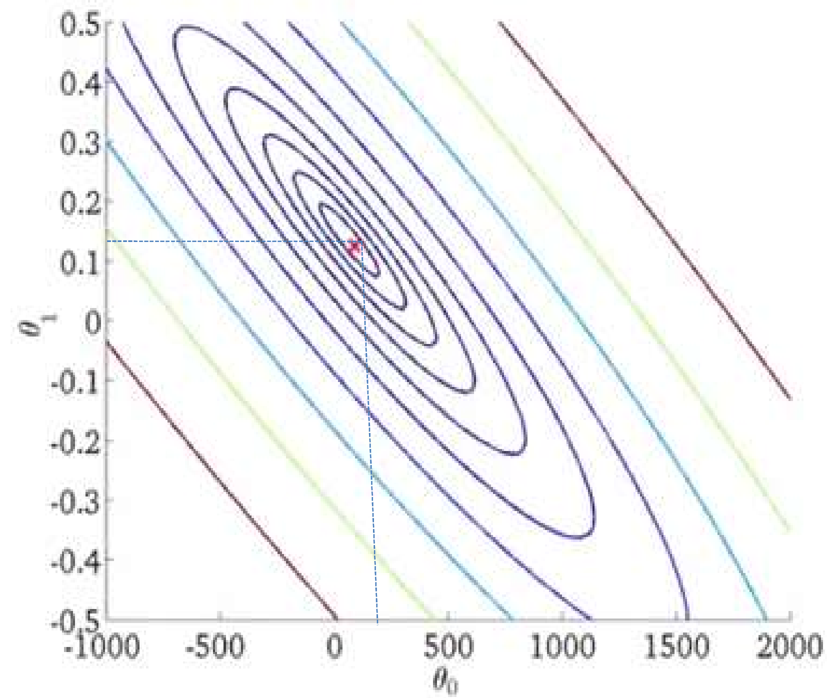
$h_\theta(x)$

(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)



Gradient Descent

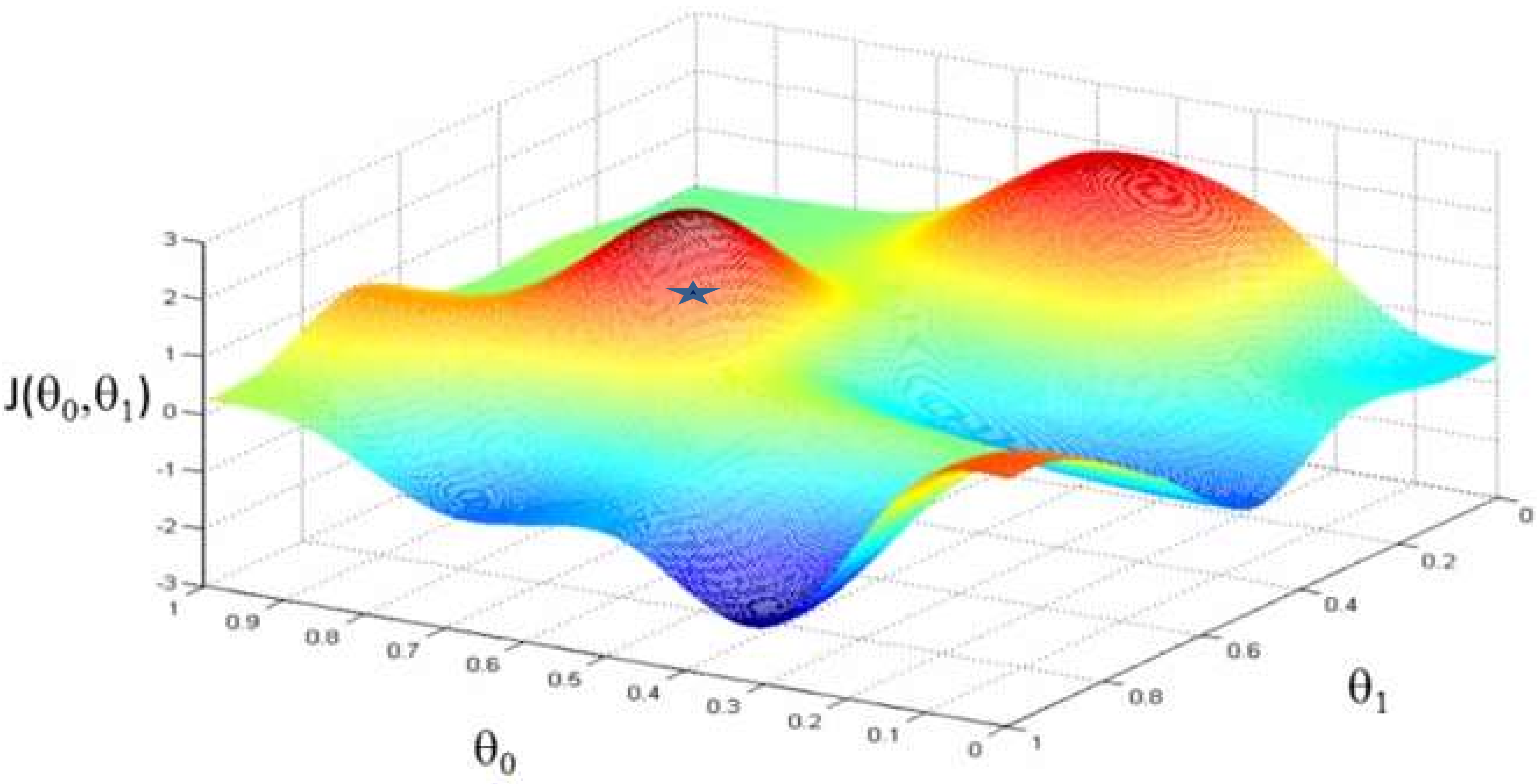
- Let some function $J(\theta_0, \theta_1)$

- We have to find $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

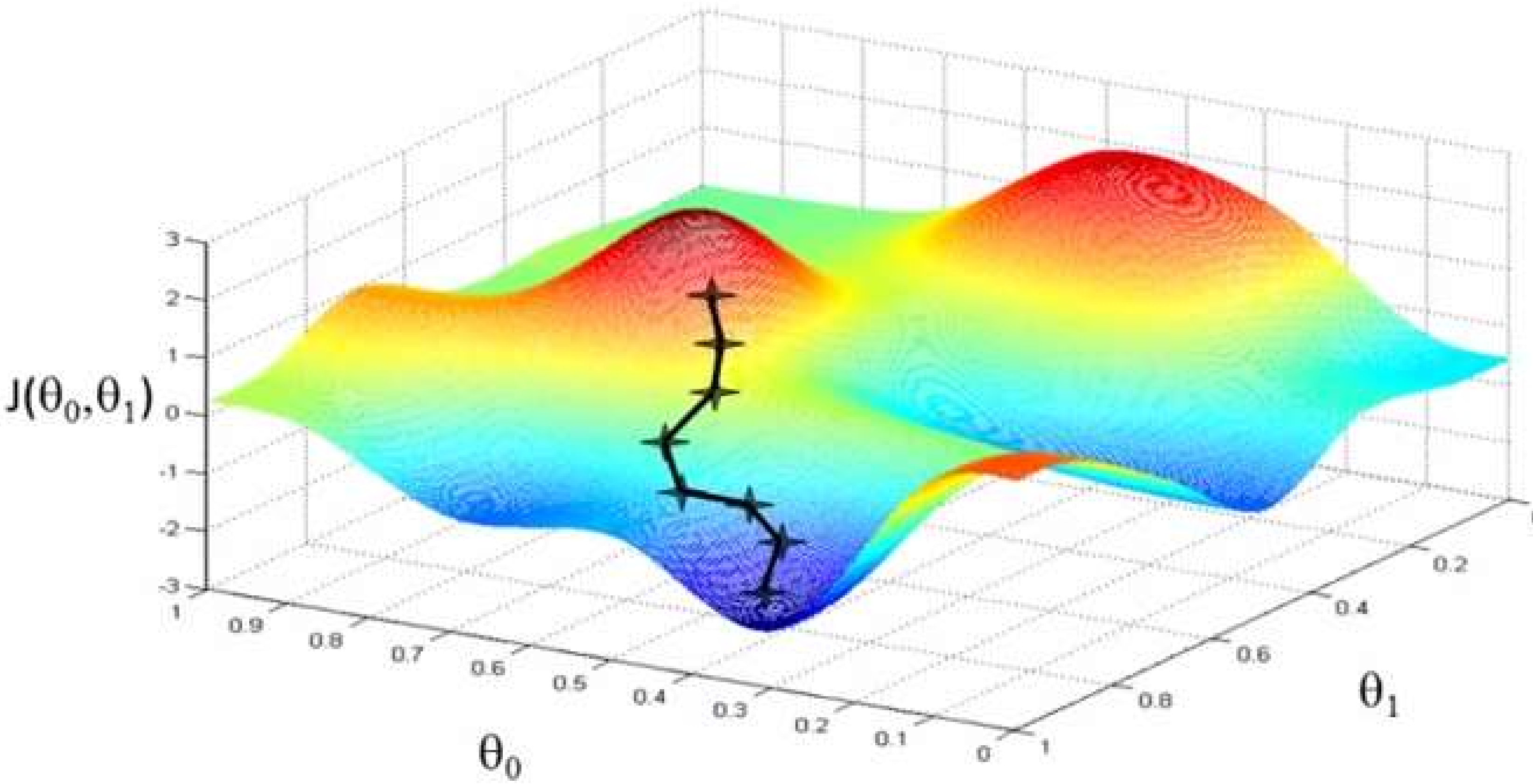
- Start with some (θ_0, θ_1) (let say $\theta_0=0$, $\theta_1=0$)

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

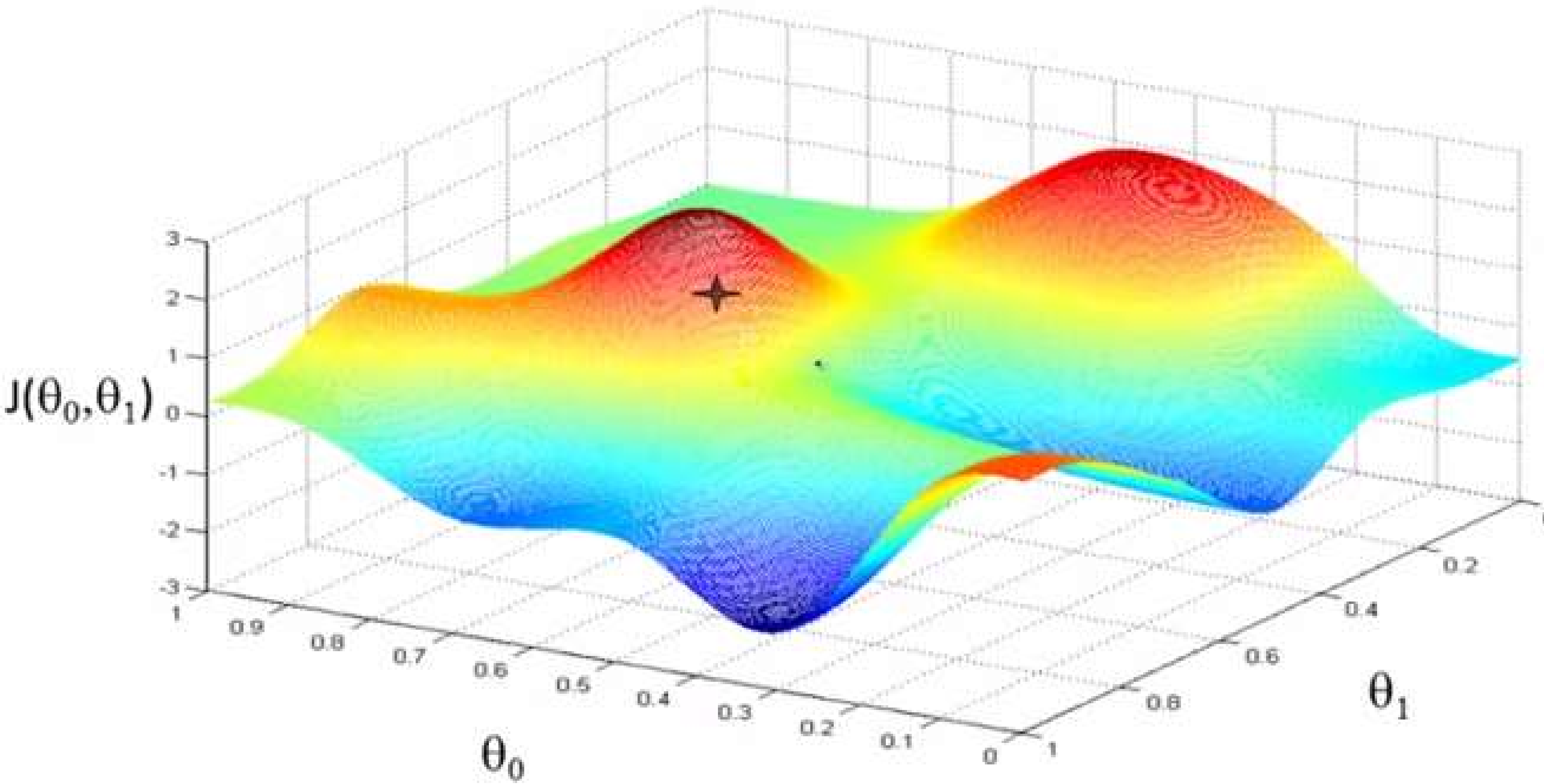
Gradient Descent



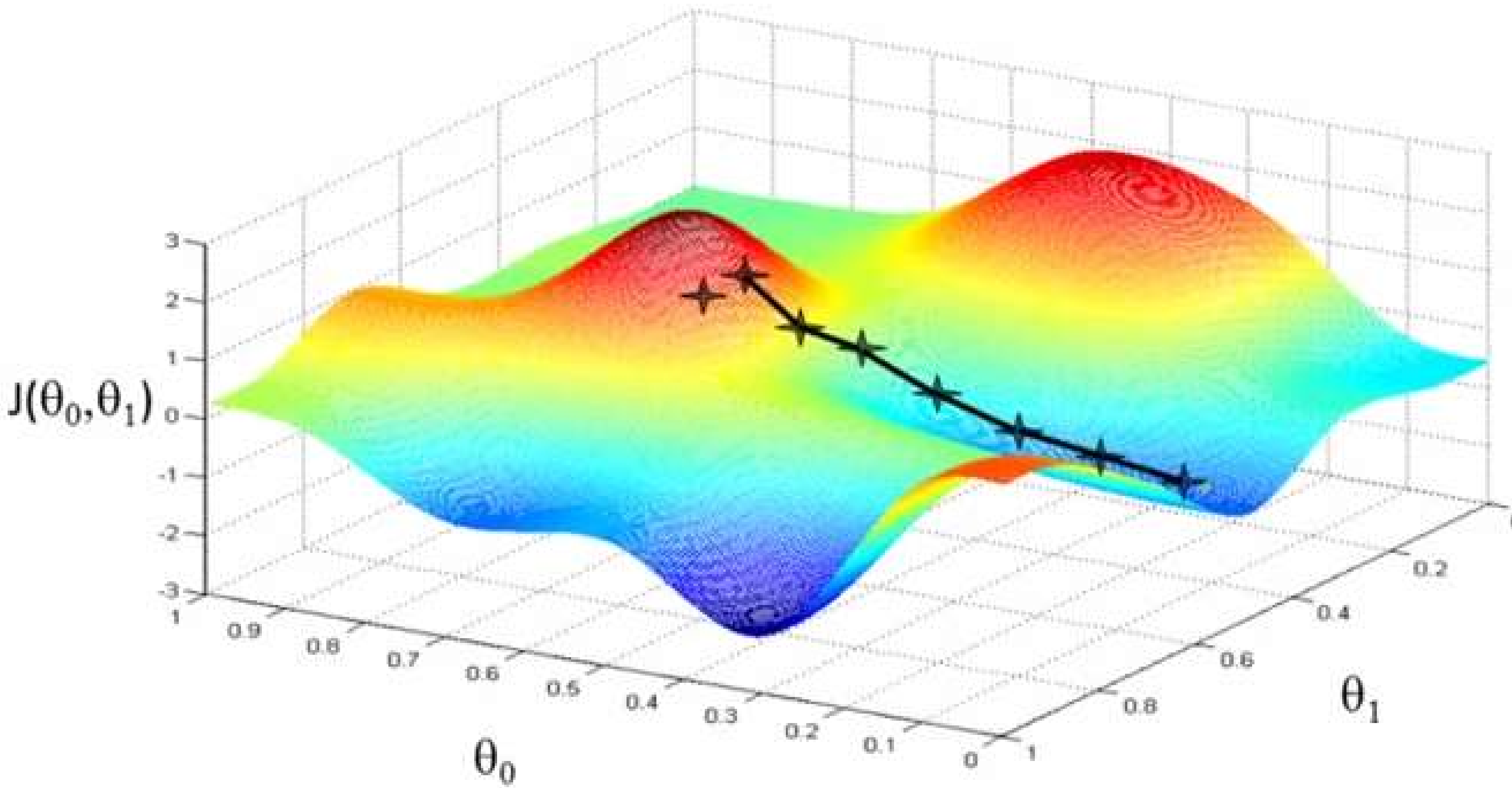
Gradient Descent



Gradient Descent



Gradient Descent



Gradient Descent Algorithm

repeat until convergence {

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

}

α = learning rate

Implication of α = it controls how bigger steps we are taking over gradient descent

Correct: Simultaneous update

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

Incorrect:

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

Gradient Descent Algorithm

- Let take a single variable

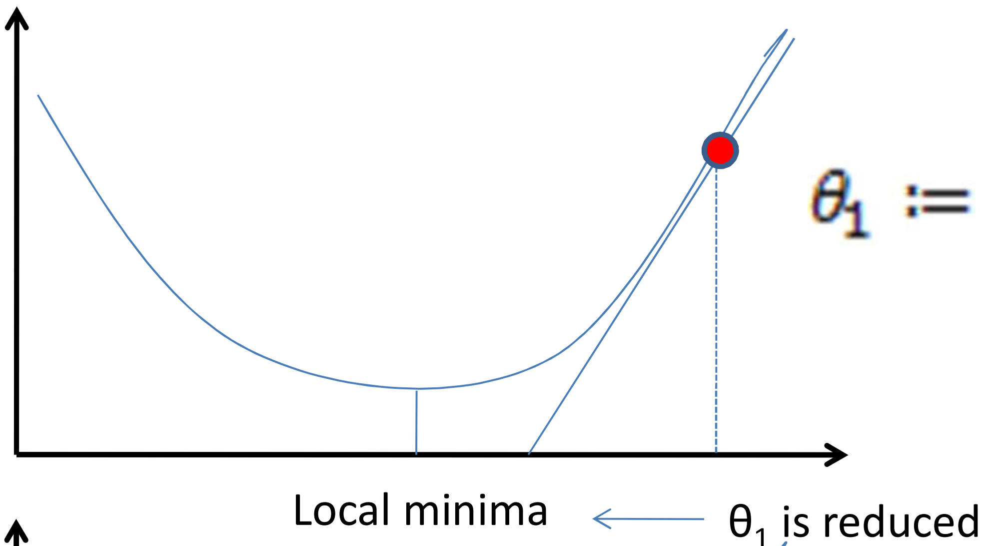
- we have to minimize

$$\min_{\theta_1} J(\theta_1) \quad \text{where } \theta_1 \in R$$

- So the GD algorithm becomes

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

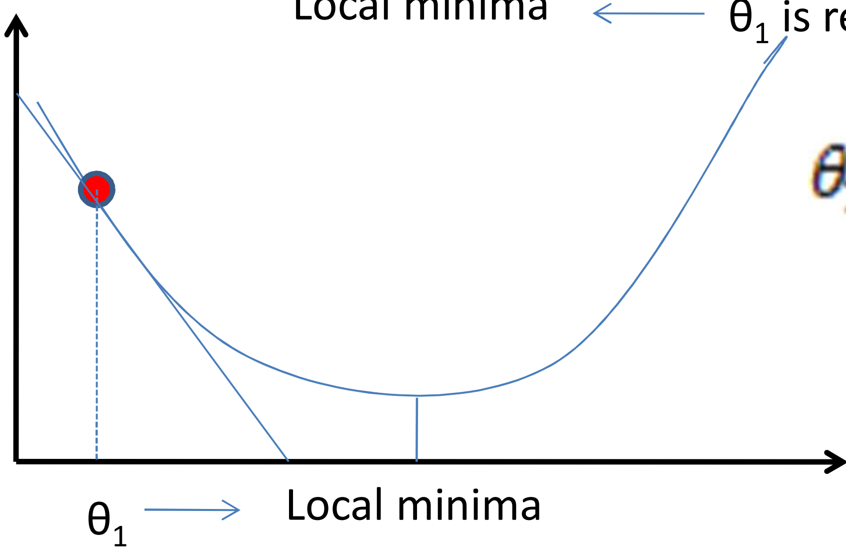
Gradient Descent Algorithm



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

≥ 0

(slope is positive)



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

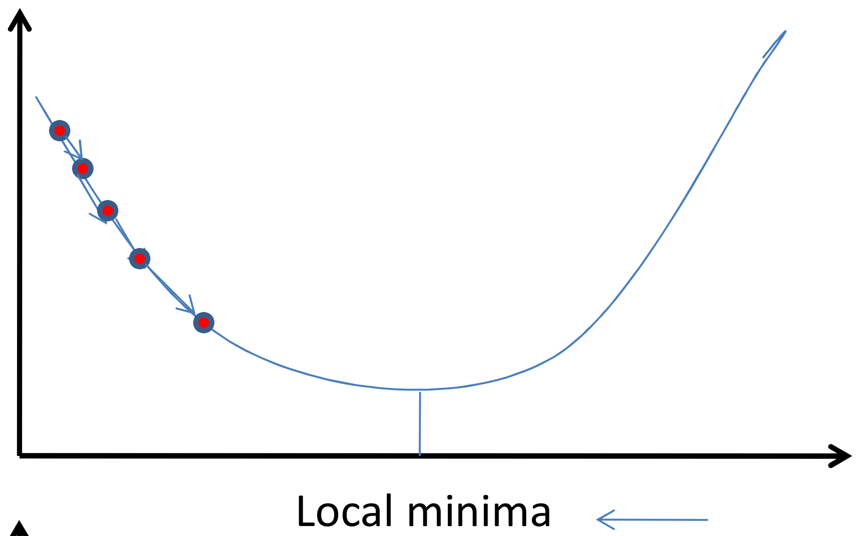
≤ 0

(slope is negative)

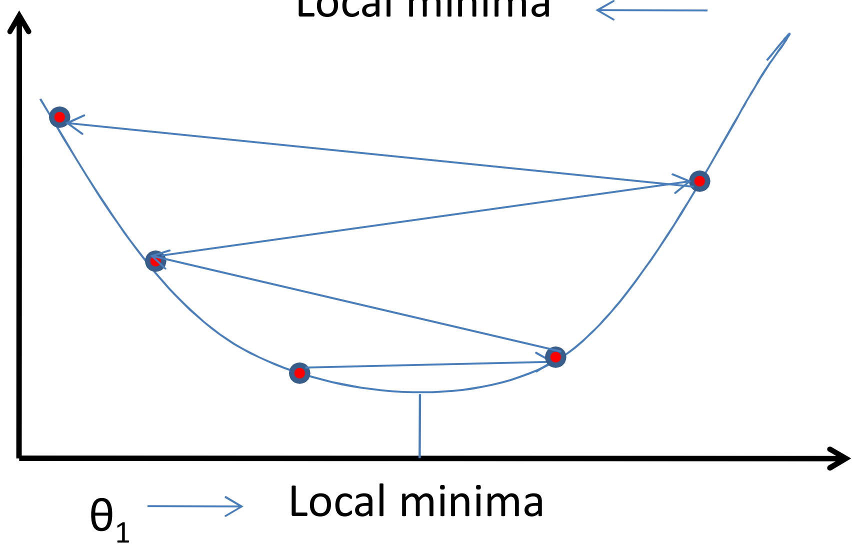
Gradient Descent Algorithm

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

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.



Multivariate Linear Regression

Univariate Hypothesis function:

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

Multivariate Hypothesis function:

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

$$\mathrm{X} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \quad \Theta = \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \\ \vdots \\ \Theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\Theta^T x = [\Theta_0 \quad \Theta_1 \quad \dots \quad \Theta_n] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boxed{h_\Theta(\mathrm{x}) = \Theta^T x}$$

where $x_0 = 1$

Multivariate Gradient Descent

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: $\theta_0, \theta_1, \ldots, \theta_n$ → **ϴ :** n+1 dimensional vector

Cost function:

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

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \underbrace{\boxed{J(\theta_0, \ldots, \theta_n)}}_{\mathbf{J(\Theta)}}$$

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

Multivariate Gradient Descent

Gradient Descent

Previously (n=1):

Repeat {

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

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

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

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

$\frac{\partial}{\partial \Theta_j} J(\Theta)$

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

}

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

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

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

Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

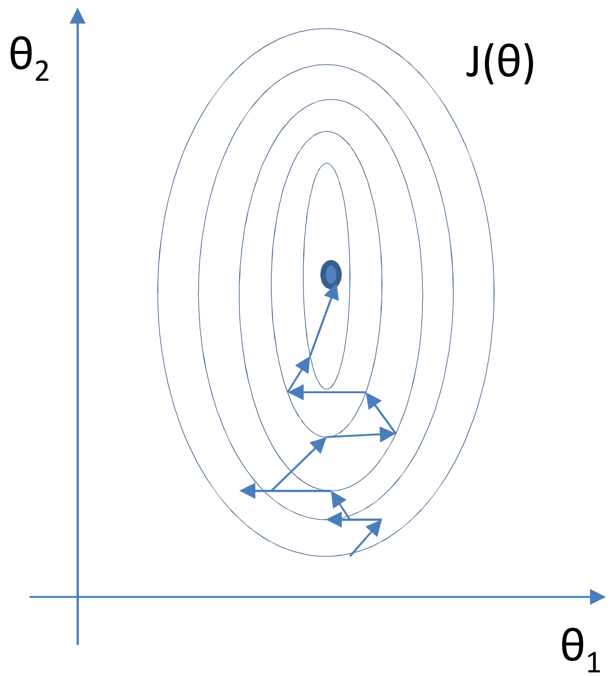
E.g. x_1 = size (0-2000 feet^2)

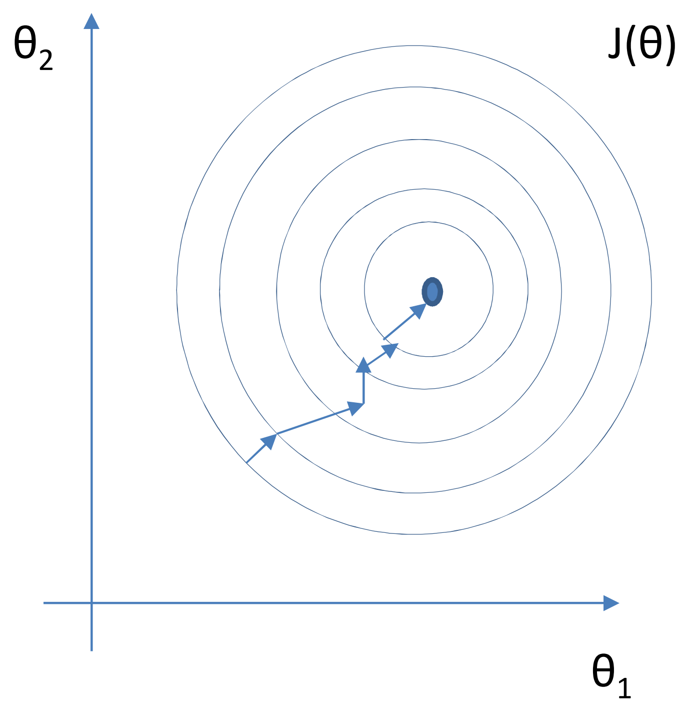
x_2 = number of bedrooms (1-5)

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

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

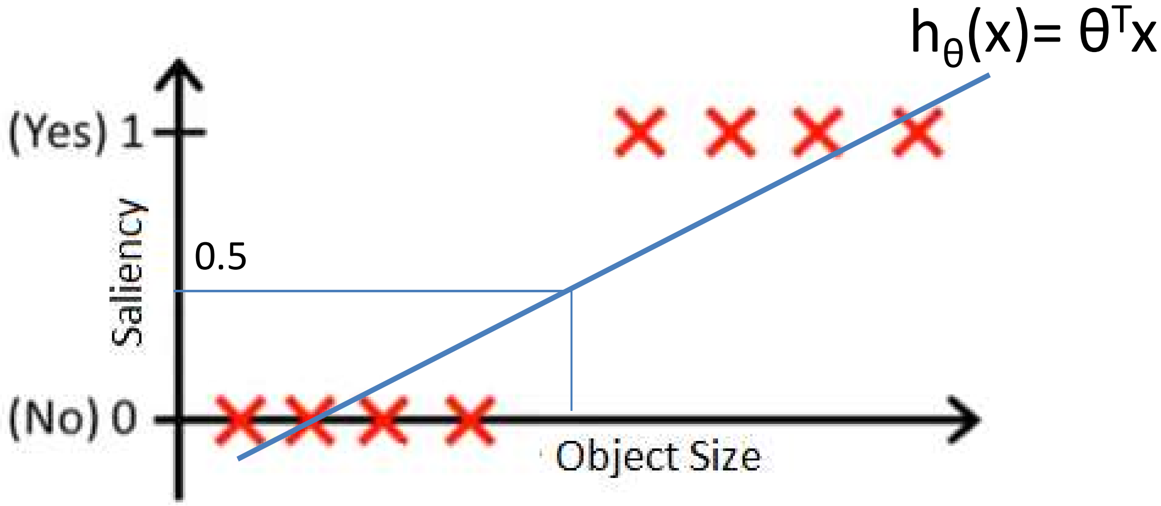
$0 \leq x_1 \leq 1 \qquad 0 \leq x_2 \leq 1$





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

Logistic Regression: Classification

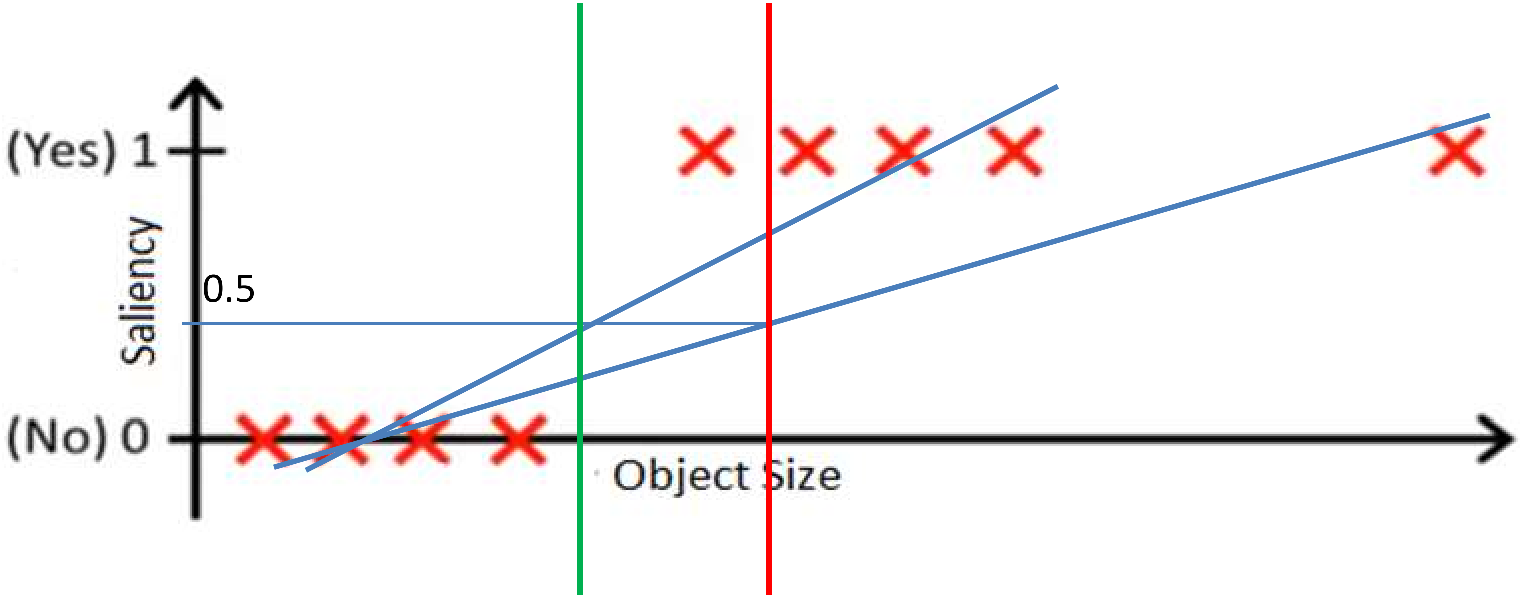


$h_\theta(x)= \theta^T x$

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"

Logistic Regression



Linear regression for classification problem is not always good

Classification: y = 0 or 1

$h_\theta(x)$ can be > 1 or < 0

Logistic Regression: $0 \leq h_\theta(x) \leq 1$

Logistic Regression Model

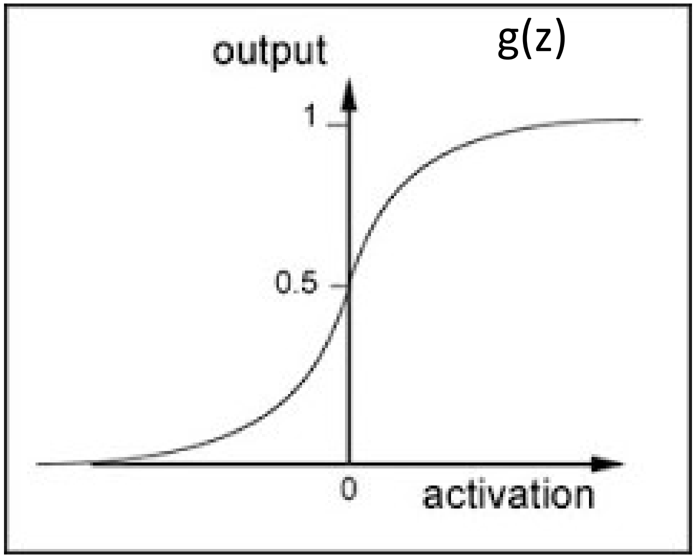
Logistic Regression: $0 \le h_\theta(x) \le 1$

Linear Regression: $h_\theta(x) = \theta^T x$

Logistic Regression:

$$h_\Theta(x) = g(\Theta^T x) \qquad h_\Theta(x) = \frac{1}{1 + e^{-\Theta^T x}}$$

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



Sigmoid Function or Logistic function

Hypothesis Representation

$h_\Theta(x) =$ estimated probability that y=1 on input x

Example: if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ Object\ Size \end{bmatrix}$

$h_\Theta(x) = 0.7$

There is 70% chance that the object is salient

$h_\Theta(x) =$ p(y=1|x, ϴ)

i.e. “probability that y=1, given x, parameterized by ϴ”

p(y=0|x; ϴ) + p(y=1|x; ϴ) = 1

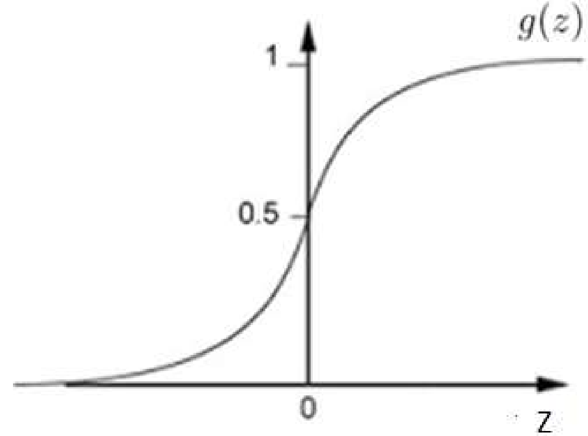
p(y=0|x; ϴ) = 1 - p(y=1|x; ϴ)

Decision Boundary

Logistic regression

$$h_\theta(x) = g(\theta^T x)$$

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



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

$g(z) \geq 0.5 \quad when \quad z \geq 0$

i.e. $\Theta^T x \geq 0$

$h_\Theta(x) = g(\Theta^T x) \geq 0.5$

whenever $\Theta^T x \geq 0$

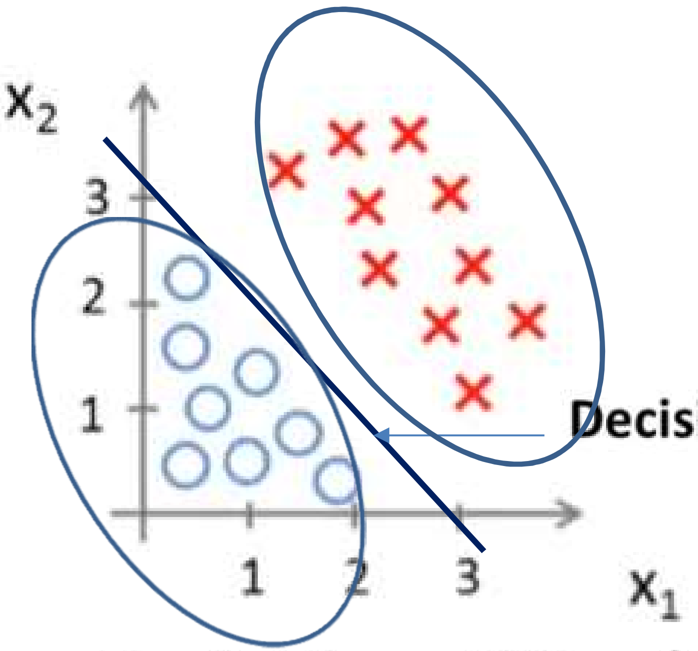
$\boxed{z}$

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

$h_\Theta(x) = g(\Theta^T x)$

i.e. $\Theta^T x < 0$

Decision Boundary



Decision Boundary

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

$$\Theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$x_1 + x_2 = 3$$

$$h_\Theta(x) = g(\Theta^T x) = 0.5$$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

i.e. $\Theta^T x \geq 0$

$$x_1 + x_2 \geq 3$$

Predict "y = 0" if

$$x_1 + x_2 < 3$$

Decision boundary is a property of hypothesis function NOT of a data set

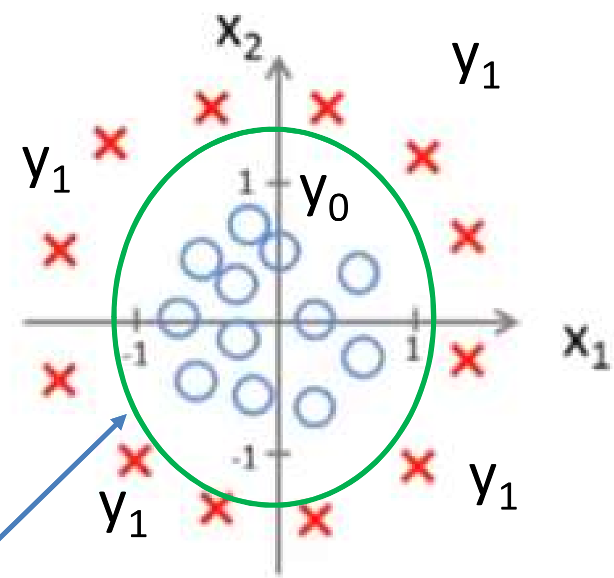
Non-Linear Decision Boundary

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

Let $\Theta^T = [-1 \quad 0 \quad 0 \quad 1 \quad 1]$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

$$x_1^2 + x_2^2 \geq 1$$



$$x_1^2 + x_2^2 = 1$$

Decision Boundary

Again, decision boundary is a property of hypothesis function NOT of a data set

Cost Function

- Optimization objective of the cost function

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $\quad x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

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

How to choose parameters θ ?

Cost Function

Cost function

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

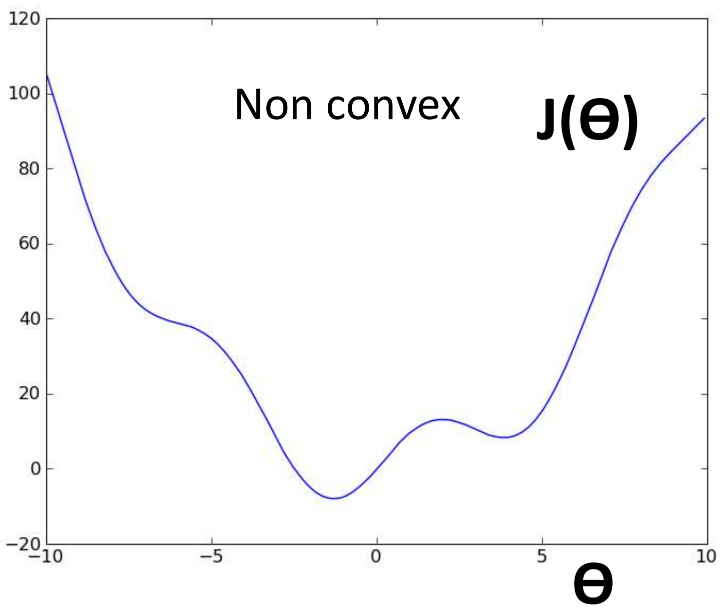
Let, $\text{Cost}\left(h_\Theta\left(x^{(i)}\right), y^{(i)}\right) = \frac{1}{2} \left(h_\Theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

So, $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$

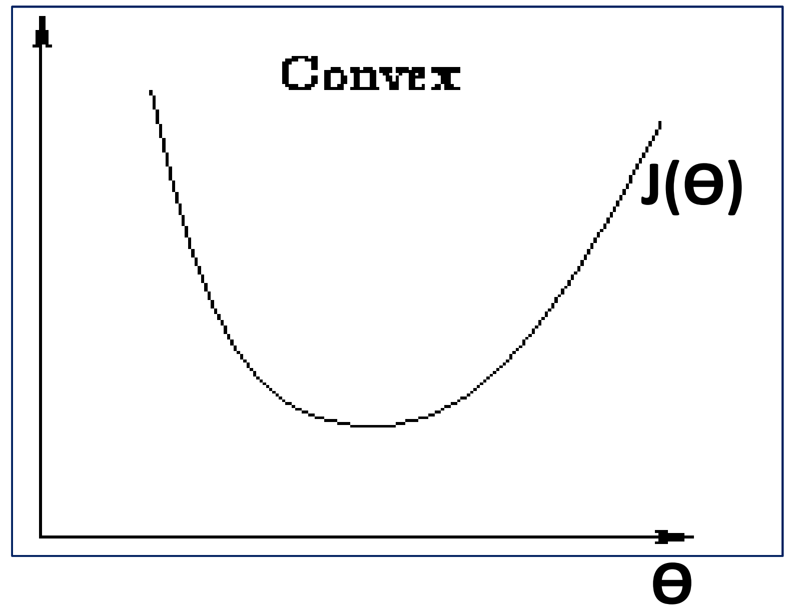
where, for logistic regression

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

Cost Function



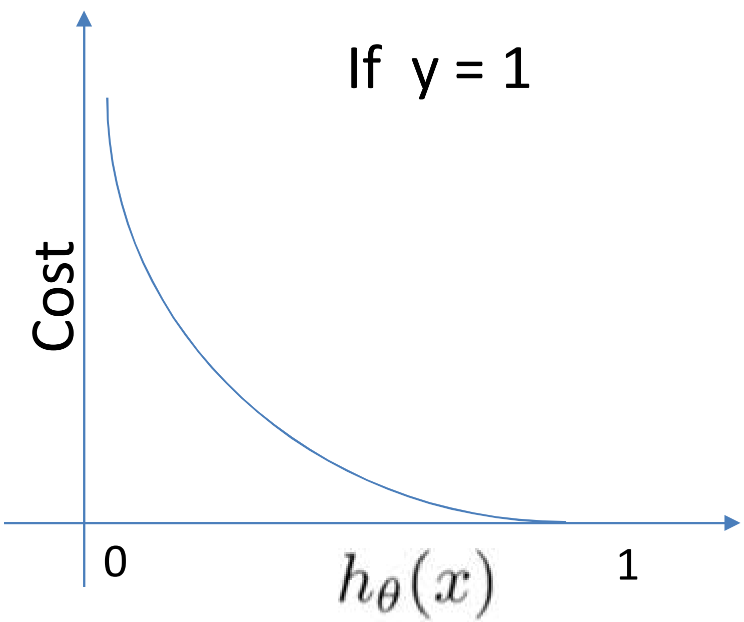
Logistic Regression



Linear Regression

Cost Function: Logistic Regression

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



Cost $= 0$ if $y = 1, h_\theta(x) = 1$

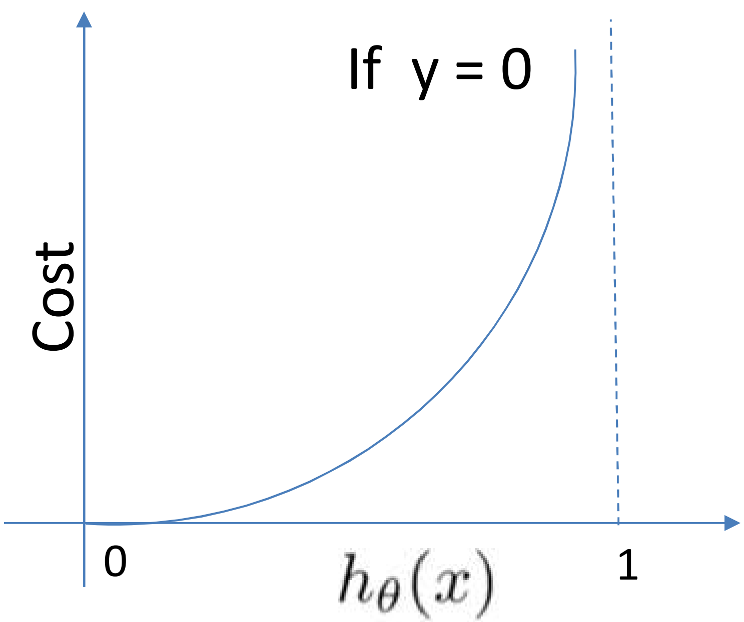
But as $h_\theta(x) \to 0$

$Cost \to \infty$

Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Cost Function: Logistic Regression

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



Cost =0 if y=0, $h_\Theta(x) = 0$
But as $h_\Theta(x) \rightarrow 1$
Cost $\rightarrow \infty$
Captures intuition that if $h_\Theta(x) = 1$, (predict $P(y=0|x; \Theta) = 1$), but y = 0, We will penalize learning algorithm by a very large cost.

It can be shown that the overall cost function is convex function and local optimum free. But details of such convexity analysis is beyond of the scope of this course.

Cost Function: Logistic Regression

Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

Note: $y = 0$ or 1 always

$$\mathrm{Cost}\,(h_\Theta(x), y) = -y\log\big(h_\Theta(x)\big) - (1-y)\log(1 - h_\Theta(x)\big)$$

$$\text{If } \mathrm{y} = 1: \ \mathrm{Cost}\,(h_\Theta(x), y) = -\log\big(h_\Theta(x)\big)\big)$$

$$\text{If } \mathrm{y} = 0: \ \mathrm{Cost}\,(h_\Theta(x), y) = -\log(1 - h_\Theta(x)\big)$$

Cost Function: Logistic Regression

Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right]$$

Principle of Maximum Likelihood Estimation

To fit parameters θ:

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

and get ϴ

To make a prediction given new x:

Output: $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

For p(y=1|x; ϴ)

How to minimize J(ϴ) ?

Cost Function and Gradient Descent

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

$$\frac{\partial}{\partial \Theta_j} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Cost Function and Gradient Descent

Gradient Descent

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

Want $\min_\theta J(\theta)$:

Repeat {

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

(simultaneously update all θ_j)

}

For Linear Regression:

$$h_\Theta(\mathrm{x}) = \Theta^T x$$

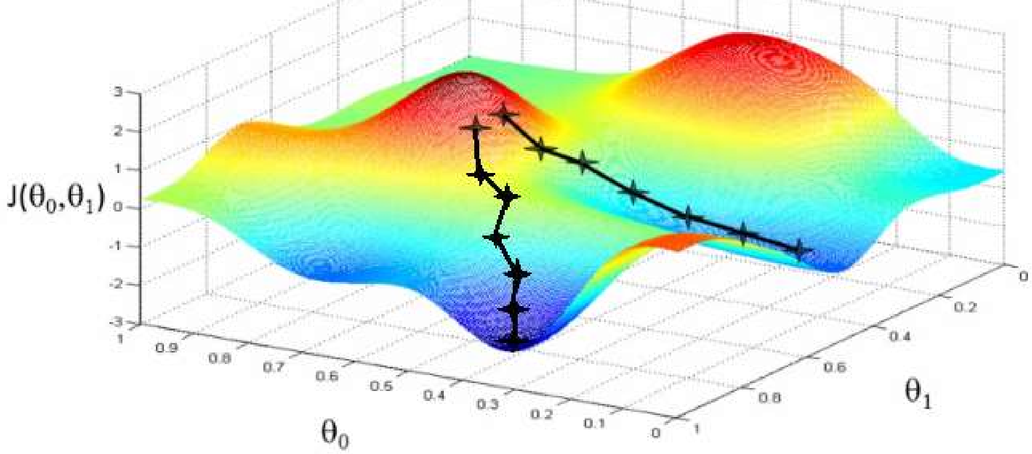
For Logistic Regression:

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

Algorithm looks identical to linear regression!

Gradient descent optimization

- Problems:
  - Choosing step size
    - too small $\rightarrow$ convergence is slow and inefficient
    - too large $\rightarrow$ may not converge
  - Can get stuck on “flat” areas of function
  - Easily trapped in local minima



Stochastic gradient descent

Stochastic (definition):

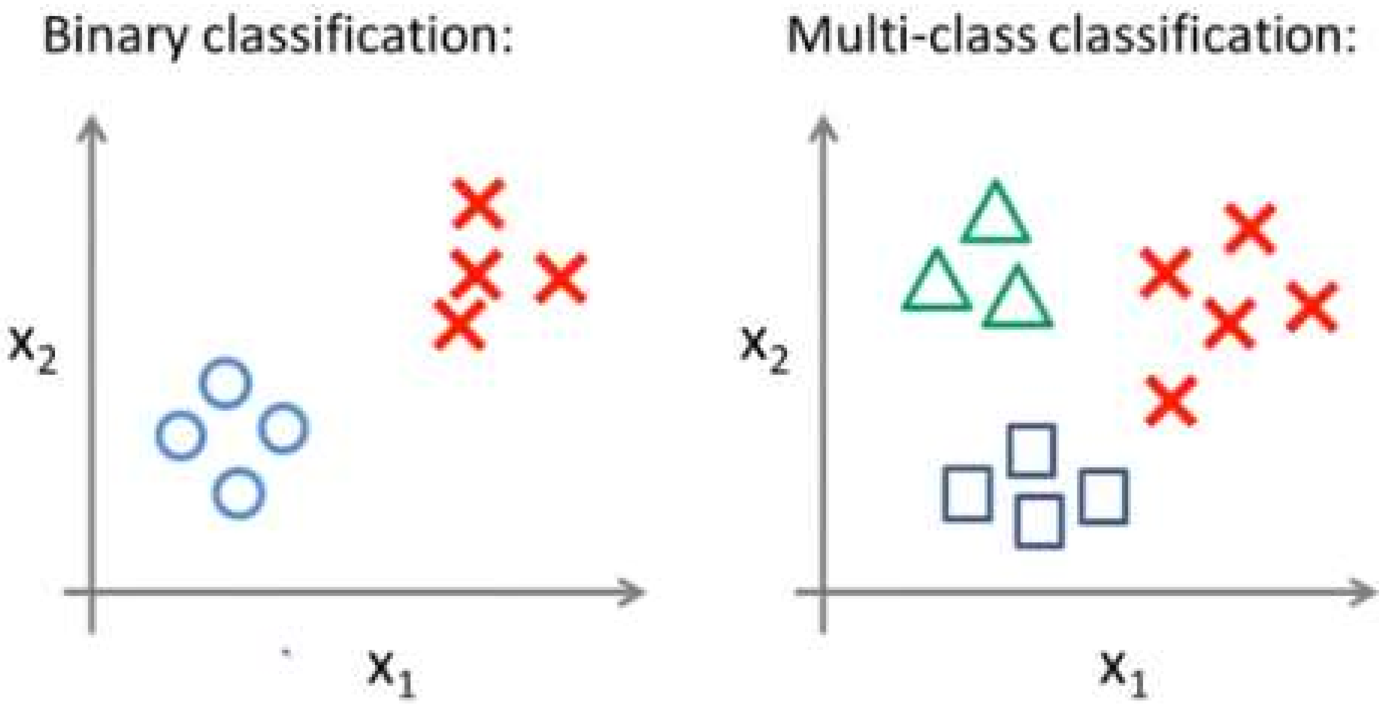
1. involving a random variable
2. involving chance or probability; probabilistic

Stochastic gradient descent

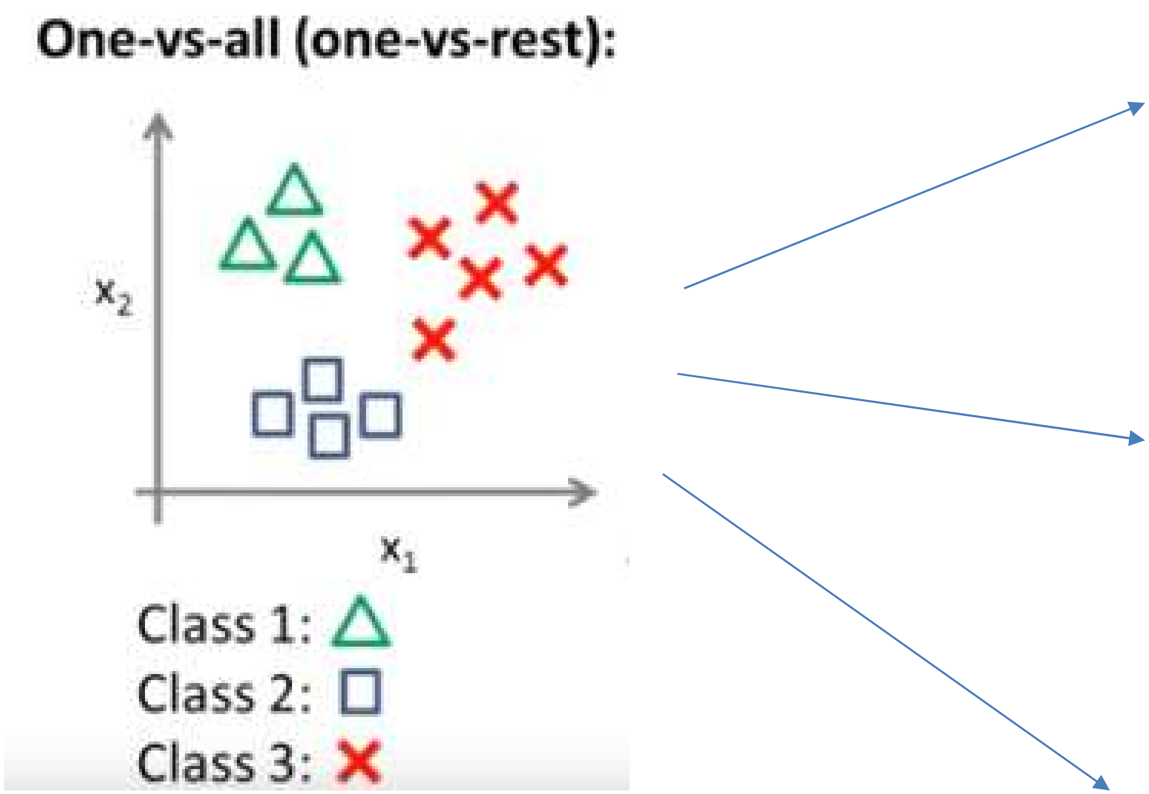
- Application to training a machine learning model:
  1. Choose one sample from training set
  2. Calculate loss function for that single sample
  3. Calculate gradient from loss function
  4. Update model parameters a single step based on gradient and learning rate
  5. Repeat from 1) until stopping criterion is satisfied

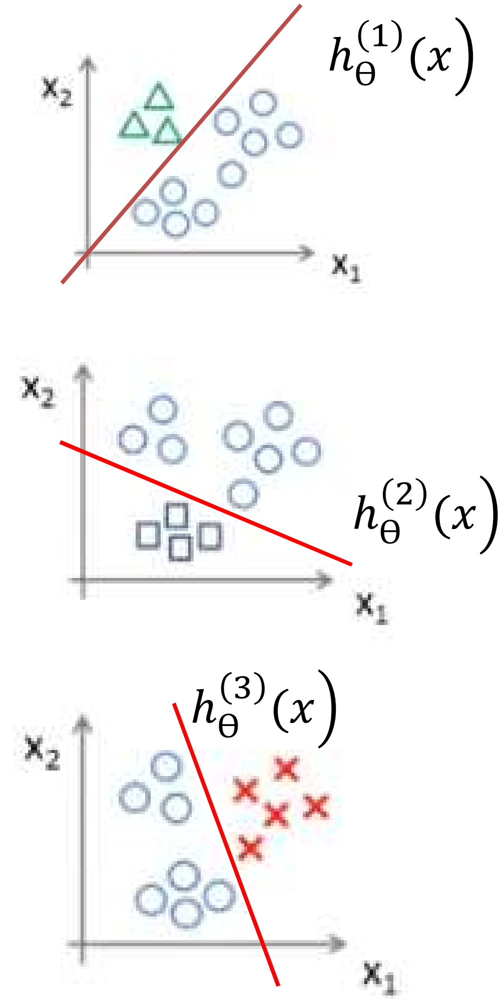
- Typically entire training set is processed multiple times before stopping.
- Order in which samples are processed can be fixed or random.

Multi Class Classification



One vs. All (One vs. Rest)





$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 1, 2, 3)$$

One vs. All (One vs. Rest)

Train a logistic regression classifier $h_\theta^{(i)}(x)$ 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

$$\max_i h_\theta^{(i)}(x)$$

To continue…