

Machine Learning

Introduction

Welcome

Apple - iPhoto - New full-size X

www.apple.com/ilife/iphoto/

Store Mac iPod iPhone iPad iTunes Support

iLife '11

iPhoto iMovie GarageBand Video Showcase Resources Upgrade Now

 iPhoto '11

From your Facebook Wall to your coffee table to your best friend's inbox (or mailbox). Do more with your photos than you ever thought possible. And do it all in one place. iPhoto.

 Watch the iPhoto video ▶



What's New in iPhoto What is iPhoto?



Machine Learning

- Grew out of work in AI
- New capability for computers

Examples:

- Database mining

Large datasets from growth of automation/web.

E.g., Web click data, medical records, biology, engineering

- Applications can't program by hand.

E.g., Autonomous helicopter, handwriting recognition, most of Natural Language Processing (NLP), Computer Vision.

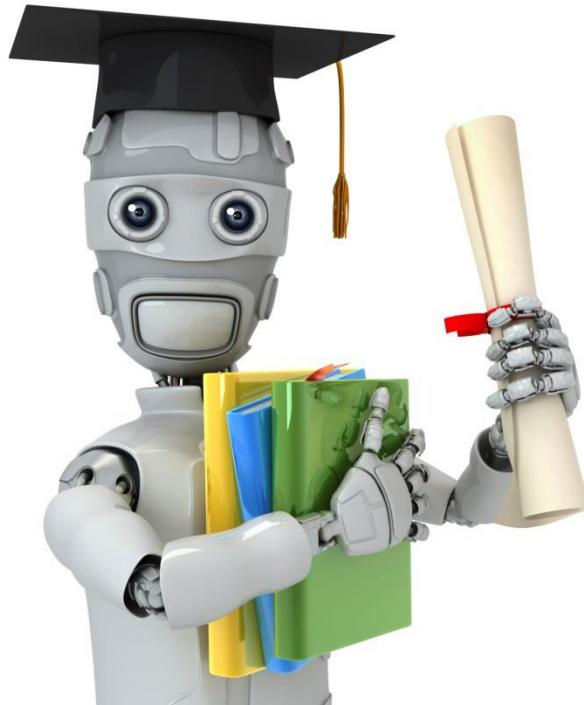


Machine Learning

- Grew out of work in AI
- New capability for computers

Examples:

- Database mining
 - Large datasets from growth of automation/web.
E.g., Web click data, medical records, biology, engineering
- Applications can't program by hand.
 - E.g., Autonomous helicopter, handwriting recognition, most of Natural Language Processing (NLP), Computer Vision.
- Self-customizing programs
 - E.g., Amazon, Netflix product recommendations
- Understanding human learning (brain, real AI).



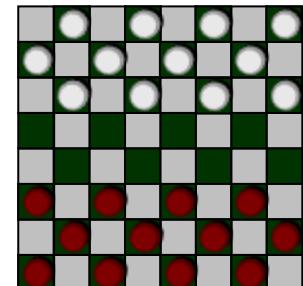
Machine Learning

Introduction

What is machine learning

Machine Learning definition

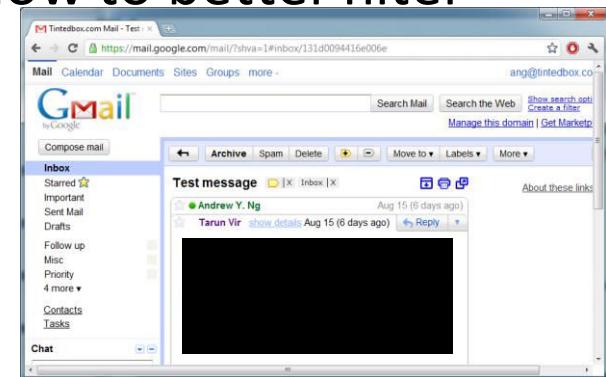
- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell (1998) Well-posed Learning Problem: A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.



“A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam. What is the task T in this setting?

- Classifying emails as spam or not spam.
- Watching you label emails as spam or not spam.
- The number (or fraction) of emails correctly classified as spam/not spam.
- None of the above—this is not a machine learning problem.

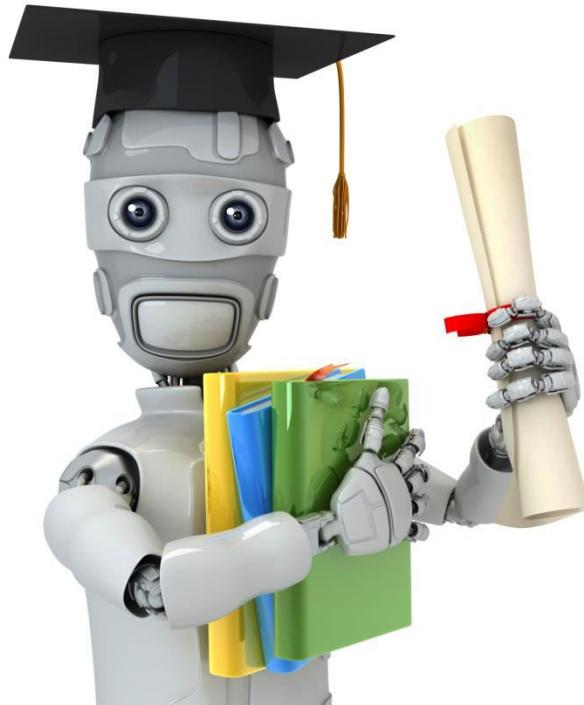


Machine learning algorithms:

- Supervised learning
- Unsupervised learning

Others: Reinforcement learning, recommender systems.

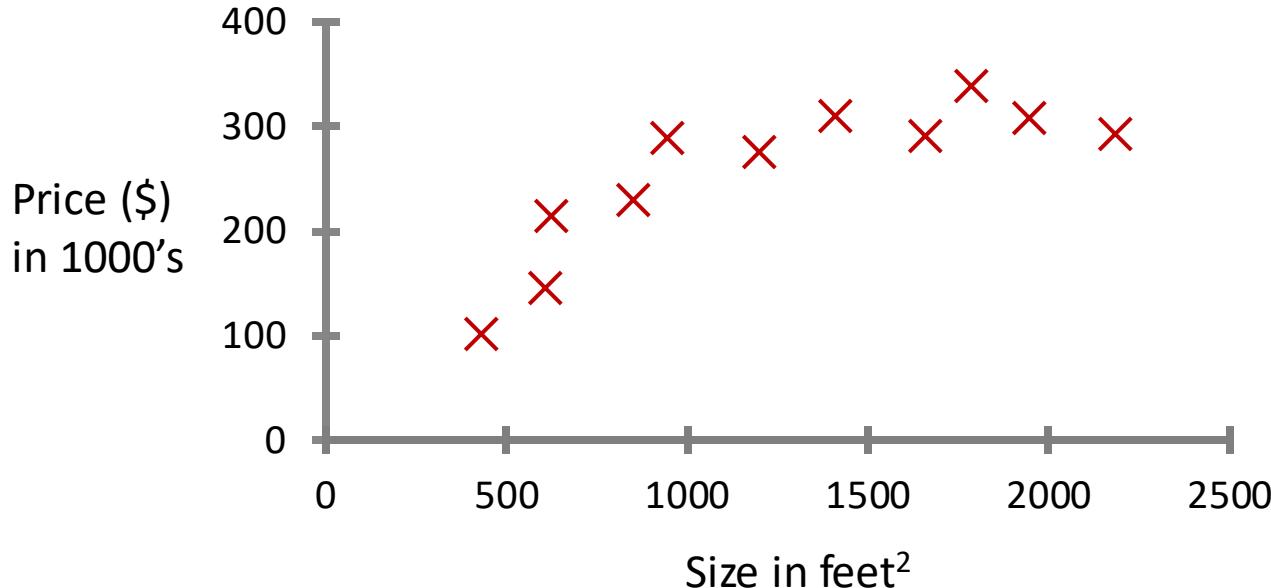
Also talk about: Practical advice for applying learning algorithms.



Machine Learning

Introduction --- Supervised Learning

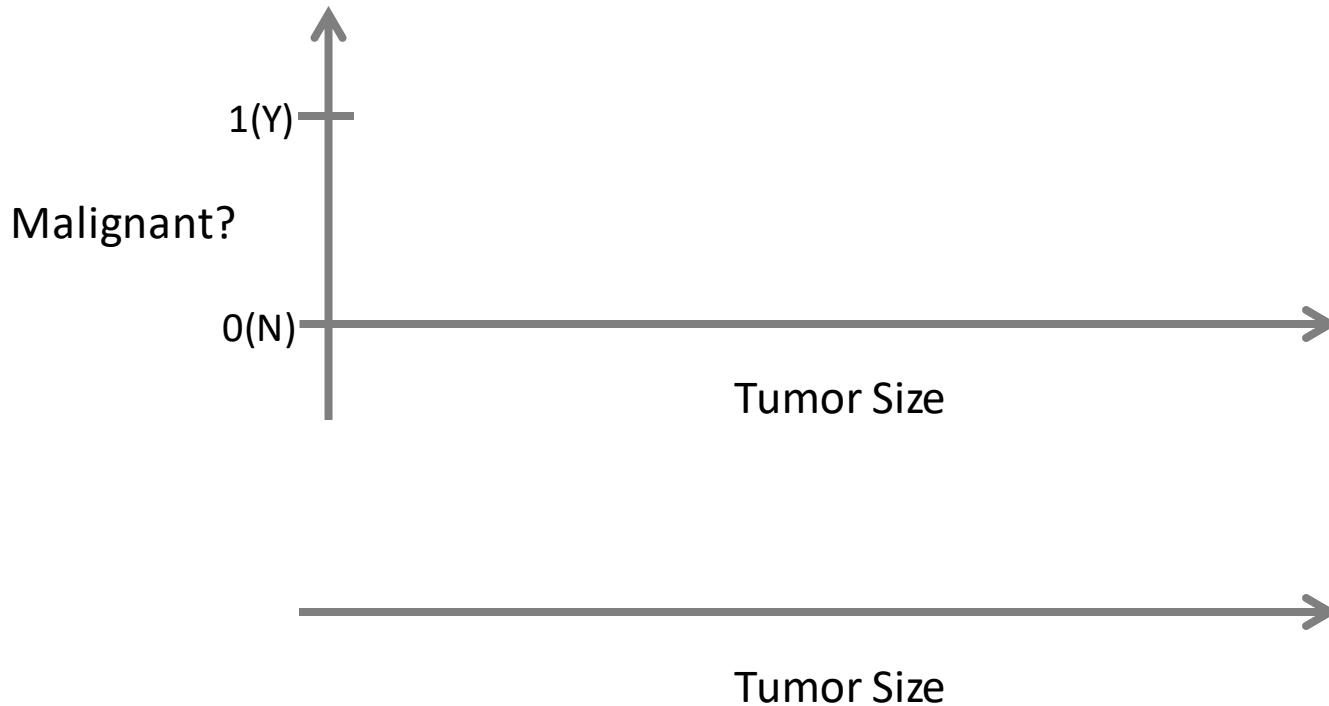
Housing price prediction.



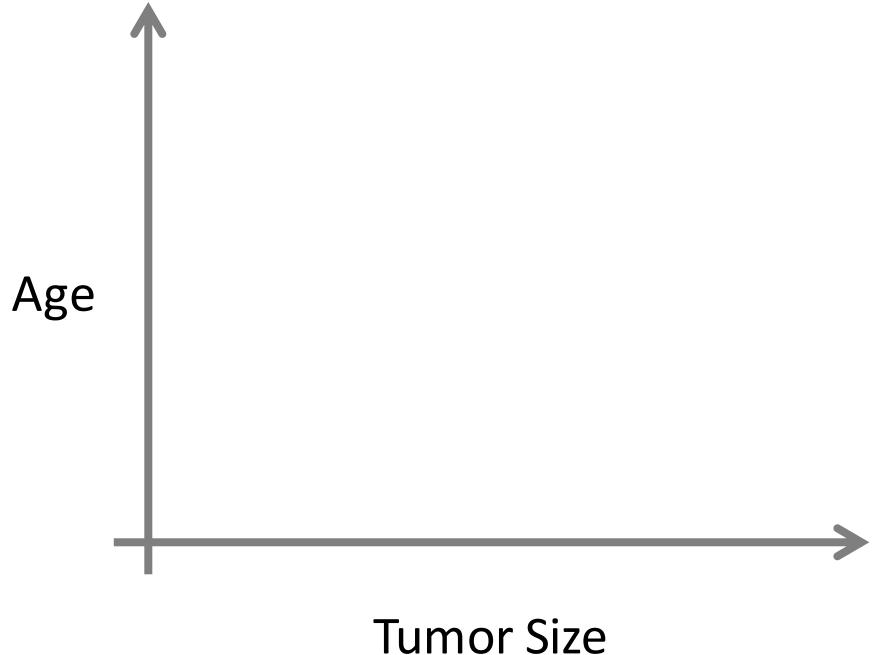
Supervised Learning
“right answers” given

Regression: Predict continuous
valued output (price)

Breast cancer (malignant, benign)



Classification
Discrete valued output (0 or 1)



- Clump Thickness
 - Uniformity of Cell Size
 - Uniformity of Cell Shape
- ...

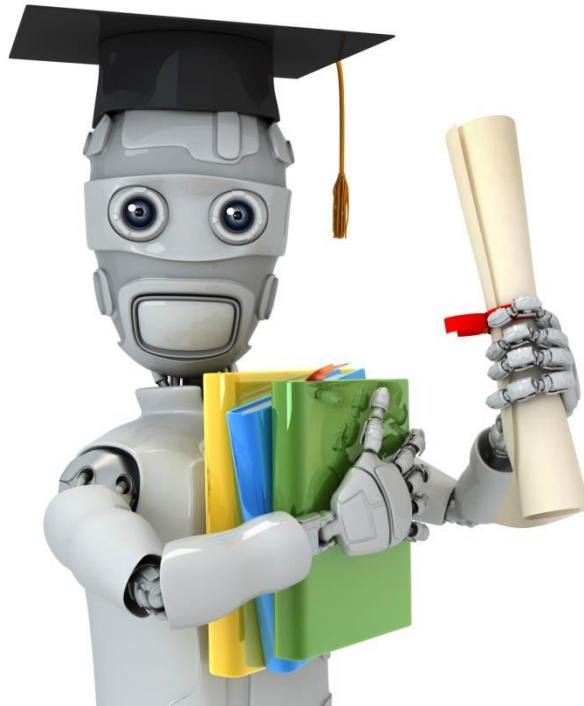
You're running a company, and you want to develop learning algorithms to address each of two problems.

Problem 1: You have a large inventory of identical items. You want to predict how many of these items will sell over the next 3 months.

Problem 2: You'd like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised.

Should you treat these as classification or as regression problems?

- Treat both as classification problems.
- Treat problem 1 as a classification problem, problem 2 as a regression problem.
- Treat problem 1 as a regression problem, problem 2 as a classification problem.
- Treat both as regression problems.

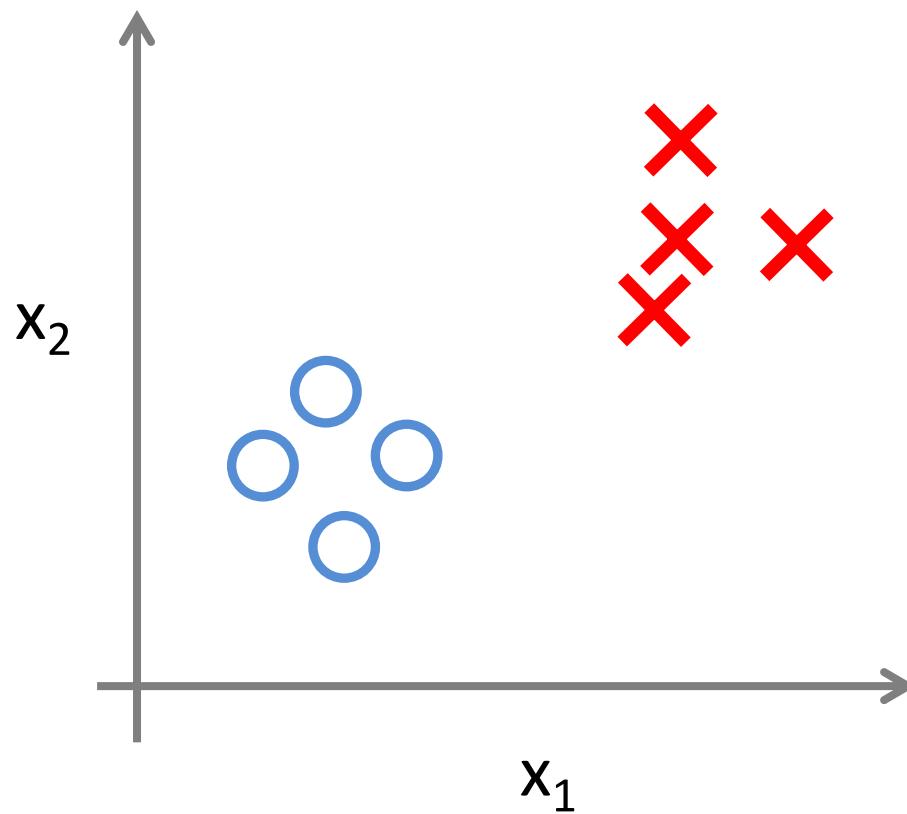


Machine Learning

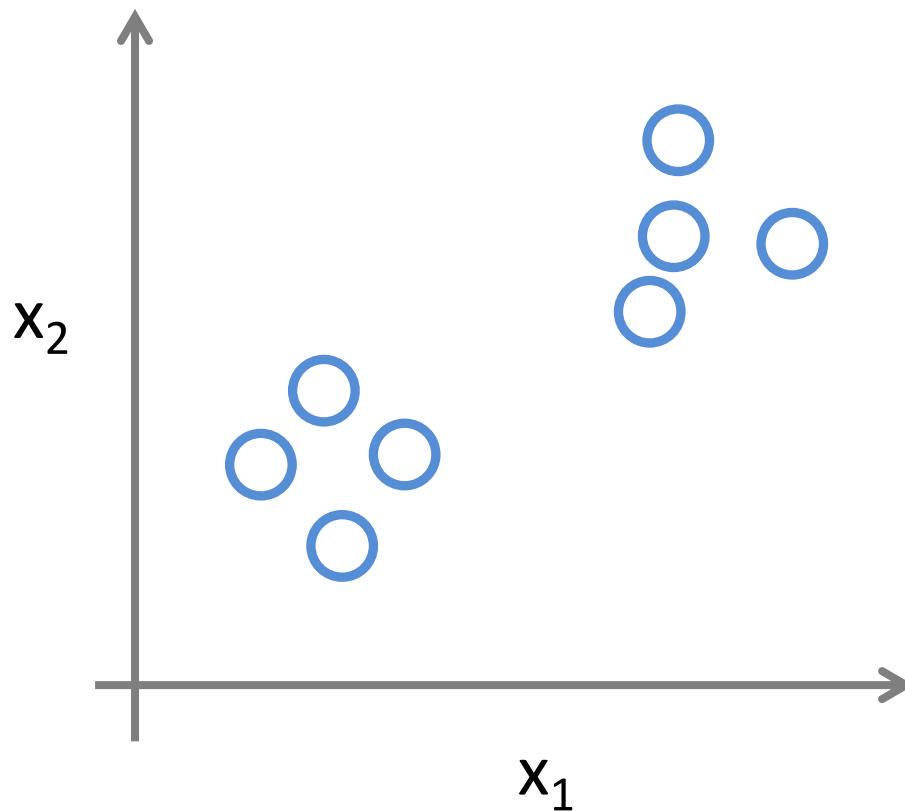
Introduction

Unsupervised Learning

Supervised Learning



Unsupervised Learning



Google News  news.google.com  

Web Images Videos Maps News Shopping Gmail more ▾ andrewyantakng@gmail.com | Web History | Settings ▾ | Sign out

Google news

Search News Search the Web Advanced news search U.S. edition ▾ Add a section ▾

Top Stories

- Deepwater Horizon
- Fed meeting
- Foreign exchange market
- Lindsay Lohan
- IBM
- Tom Brady
- Toronto International Film Festival
- Paris Hilton
- Iran
- Hurricane Igor

Starred                 - San Francisco Bay Area**
- World**
- U.S.**
- Business**
- Sci/Tech**
- More Top Stories**
- Spotlight**
- Health**
- Sports**
- Entertainment**

All news Headlines Images

Top Stories

Christine O'Donnell » **White House official denies Tea Party-focused ad campaign** CNN International - Ed Henry - 1 hour ago Democratic sources say the White House is not considering an ad campaign tying Republicans to the Tea Party. Washington (CNN) -- A top White House official sharply denied a report that claims President Obama's political advisers are weighing a national ...
Tea Party is misplacing the blame, former President Bill Clinton claims New York Daily News
GOP tea party backer defends Christine O'Donnell The Associated Press Atlanta Journal Constitution - Politics Daily - MyFox Washington DC - Salon all 726 news articles »

US Stocks Climb After Recession Called Over, Homebuilders Gain MarketWatch - Kristina Peterson - 16 minutes ago NEW YORK (MarketWatch) -- US stocks climbed Monday, gaining speed after a key nonprofit organization officially called the recession over, giving investors a boost of confidence in the gradual economic recovery. Longest recession since 1930s ended in June 2009, group says Los Angeles Times
Downturn Was Longest in Decades, Panel Confirms New York Times Wall Street Journal - AFP - CNN - USA Today all 276 news articles »

Deepwater Horizon » **BP Oil Well, Site of National Catastrophe, Dies at One** Vanity Fair - Juli Weiner - 22 minutes ago The BP oil well, site of the Deepwater Horizon explosion that led to the worst oil spill in US history, died today at one year old.
Video: Blown-out BP Well Finally Killed in Gulf  The Associated Press Weiss Doubts BP Would End Operations in Gulf of Mexico: Video Bloomberg CNN International - Wall Street Journal (blog) - The Guardian - New York Times all 2,292 news articles »

Recent

Recession officially ended in June 2009 CNNMoney - Chris Isidore - 39 minutes ago Hurricane Igor lashes Bermuda USA Today - Gerry Broome - 5 minutes ago 'Explain what you want from us,' reads front-page editorial msnbc.com - Olivia Torres - 10 minutes ago

Crisis response: Pakistan floods

San Francisco Bay Area - Edit  
Clorox » Bay Biz Buzz: Clorox close to selling STP, Armor All San Jose Mercury News - 48 minutes ago - all 24 articles »
Google's official beekeeper keeps the company buzzing with excitement San Jose Mercury News - Bruce Newman - 1 hour ago

Jon Sylvia » Martinez man still unconscious as police investigate weekend shooting San Jose Mercury News - Robert Salonga - 48 minutes ago - all 6 articles »

Spotlight

Sarkozy rages at EU 'humiliation' Financial Times - Peggy Hollinger - Sep 16, 2010

Google News

news.google.com

Top Stories

Christine O'Donnell [Home](#) [official denies Tea Party-focused ad campaign](#)

U.S. [Recession officially ended in June 2009](#)

U.S. [Hurricane Igor](#)

U.S. [US Stocks Climb After Recession Called Over](#)

U.S. [Homebuilders Gain](#)

U.S. [BP Oil Well, Site of National Catastrophe, Dies at One](#)

U.S. [Allen: Well is dead, but much Gulf Coast work remains](#)

U.S. [Crisis response: Pakistan floods](#)

U.S. [San Francisco Bay Area - Edit](#)

U.S. [RECESSION](#)

U.S. [San Jose Mercury News - 48 minutes ago](#)

U.S. [Google's official homepage keeps the company buzzing with excitement](#)

U.S. [San Jose Mercury News - Bruce Newman - 1 hour ago](#)

U.S. [San Jose Mercury News - Robert Salonga - 48 minutes ago](#)

U.S. [Spotlight](#)

U.S. [Sarkozy rages at EU humiliation](#)

All news Headlines Images

Search News Search the Web Advanced news search

andyrewantakng@gmail.com | Web History | Settings | Sign out

BP Kills Macondo; But Its Legacy Lives On

THE SOURCE

Financial Services Transport Leisure Insurance Oil & Gas Sport Caught on the Web Betting Technology

September 20, 2010, 12:44 PM GMT

Article Comments (2)

Email Print Permalink Like 2 Text +

By James Herron

BP confirmed late Sunday that the Macondo well that leaked almost five million barrels of oil into the Gulf of Mexico has been permanently sealed, but the well will continue to affect BP and the wider oil industry for many years.

The most immediate worry for BP and its shareholders is how the authorities will apportion blame for the spill. BP's own investigation so far has blamed the accident on a faulty blowout preventer.

Fire boat response crews battled the burning remnants of the offshore oil rig Deepwater Horizon on April 21, 2010.

The Source is WSJ.com's Europe home for rapid-fire analysis of the day's big business and finance stories. It is edited by Lauren Mills, based in London.

About The Source

Who Needs Plaza II Anyway

Will Banks Be Forced to Split Retail And Banking Arms?

Timing of Ratings Award Intriguing

BP Kills Macondo, But Its Legacy Lives On

We Already Need a Sanval to Basal III!

THE SOURCE HOME PAGE

edition.cnn.com/2010/09/20/gulf/oil.disaster/

INTERNATIONAL U.S. MEXICO ARABIC

Set edition preference

CNN

Home Video World U.S. Africa Asia Europe Latin America Middle East Business W

Allen: Well is dead, but much Gulf Coast work remains

By the CNN Wire Staff September 20, 2010 – Updated 1317 GMT (2117 HKT)

Click to play

What next for Gulf oil spill?

(CNN) -- The ruptured Macondo well, a mile under the Gulf of Mexico off the Louisiana coast, has been pronounced dead.

STORY HIGHLIGHTS

BP oil spill cost hits nearly \$10bn

guardian.co.uk

News | Sport | Comment | Culture | Business | Money | Life & style |

Business BP

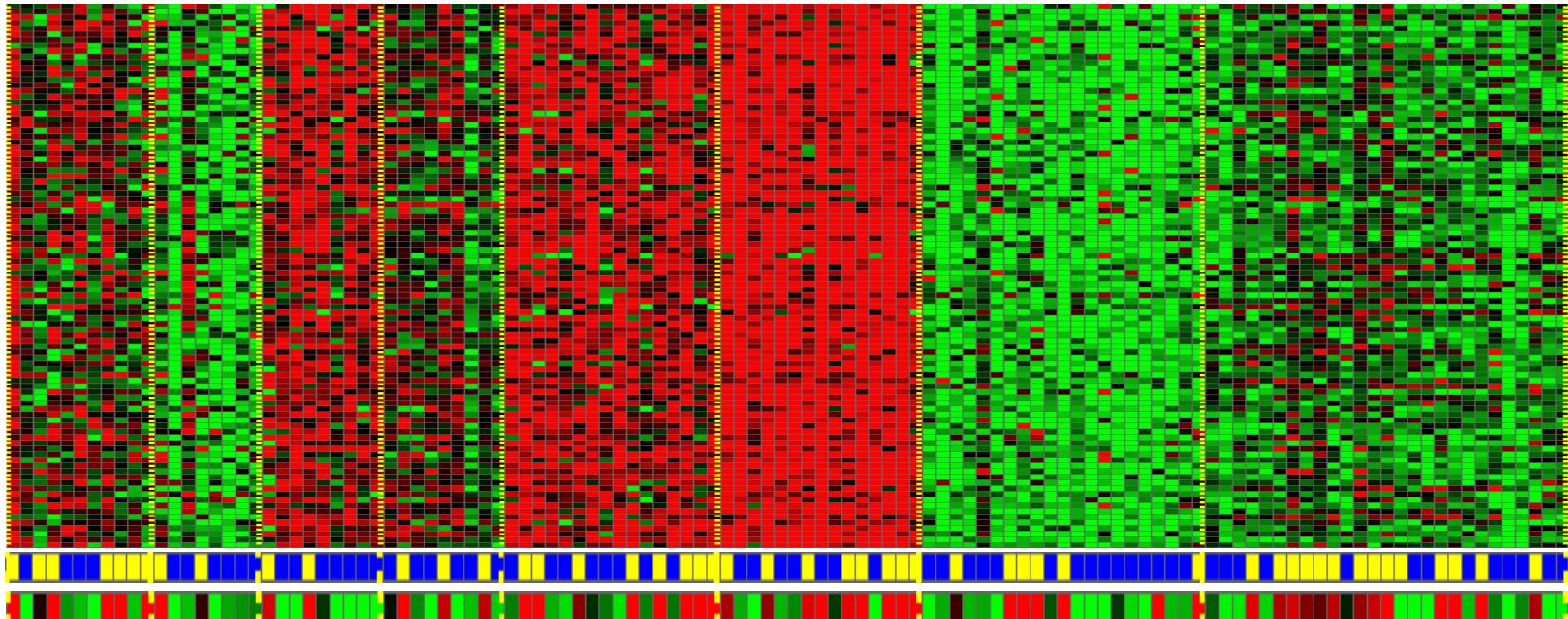
BP oil spill cost hits nearly \$10bn

BP has set up a \$20bn compensation fund after the Deepwater Horizon disaster, which has so far paid out 19,000 claims totalling more than \$240m

Julia Kollom
guardian.co.uk, Monday 20 September 2010 08.33 BST Article history

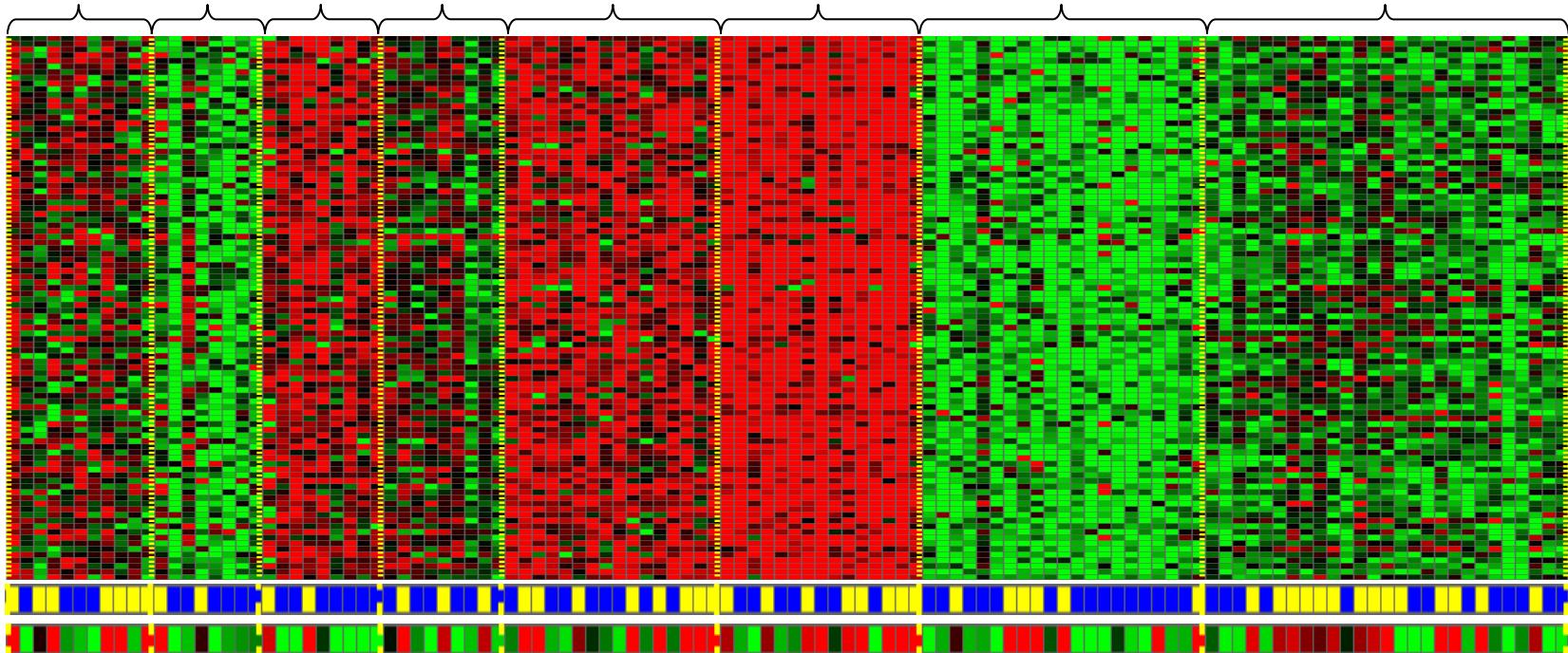
BP's costs for the Deepwater Horizon disaster have hit \$10bn. Photograph: Ho/Reuters

Genes



Individuals

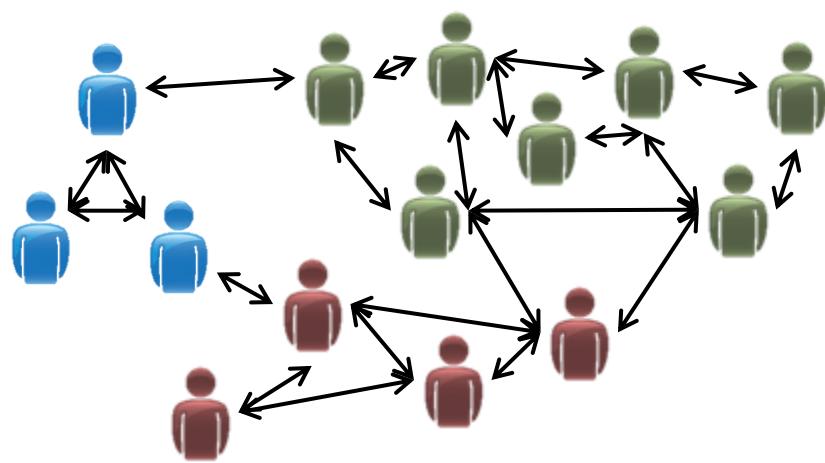
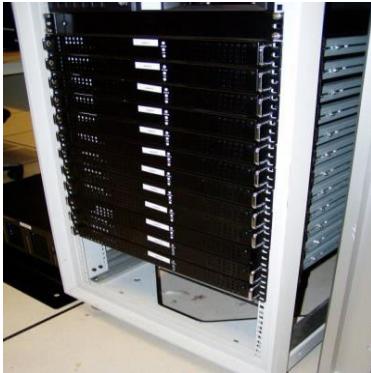
Genes



Individuals



Organize computing clusters



Social network analysis



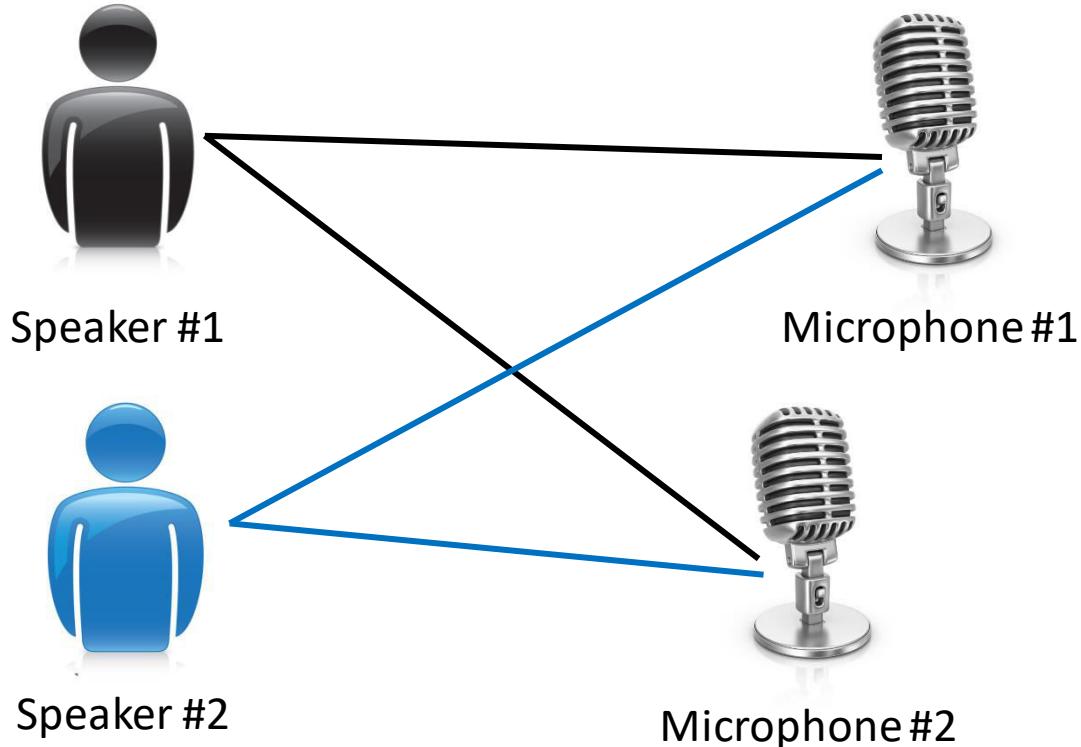
Market segmentation



Image credit: NASA/JPL-Caltech/E. Churchwell (Univ. of Wisconsin, Madison)

Astronomical data analysis

Cocktail party problem



Microphone #1: 

Output #1: 

Microphone #2: 

Output #2: 

Microphone #1: 

Output #1: 

Microphone #2: 

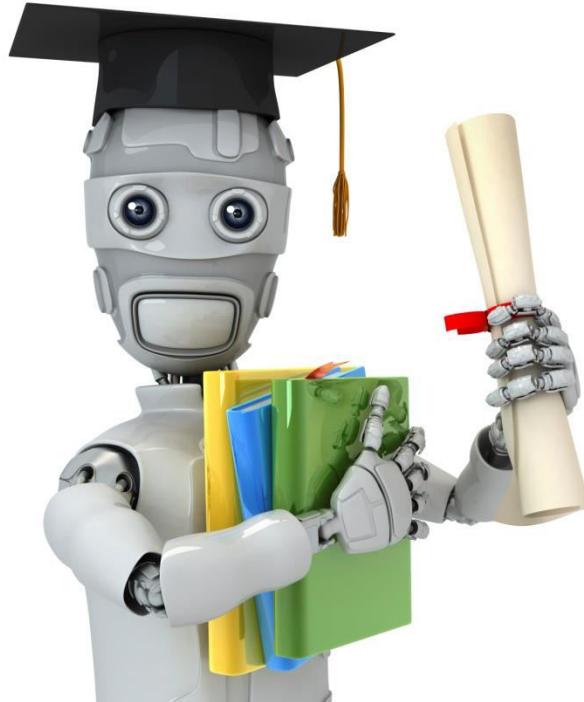
Output #2: 

Cocktail party problem algorithm

```
[W,s,v] = svd((repmat(sum(x.*x,1),size(x,1),1).*x)*x');
```

Of the following examples, which would you address using an unsupervised learning algorithm? (Check all that apply.)

- Given email labeled as spam/not spam, learn a spam filter.
- Given a set of news articles found on the web, group them into set of articles about the same story.
- Given a database of customer data, automatically discover market segments and group customers into different market segments.
- Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.



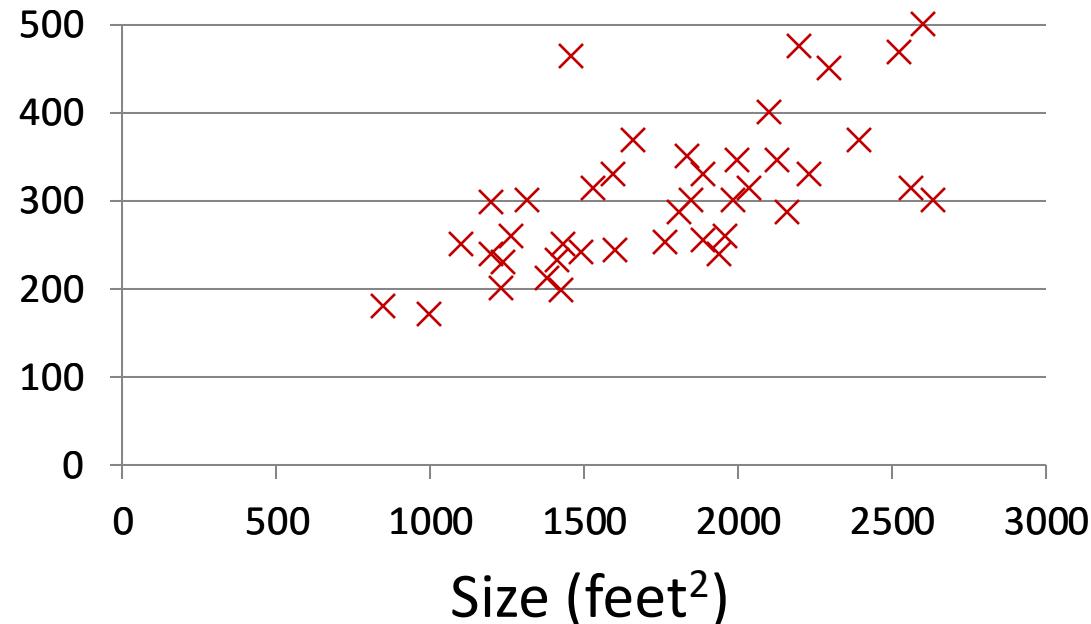
Machine Learning

Linear regression with one variable

Model representation

Housing Prices (Portland, OR)

Price
(in 1000s
of dollars)



Supervised Learning

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

Regression Problem

Predict real-valued output

Training set of housing prices (Portland, OR)

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

Training Set



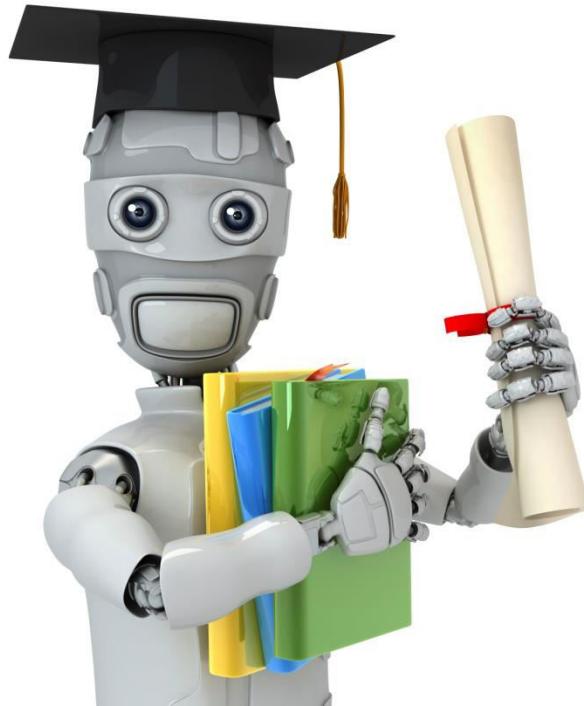
Learning Algorithm

Size of
house



How do we represent h ?

Linear regression with one variable.
Univariate linear regression.



Machine Learning

Linear regression with one variable

Cost function

Training Set

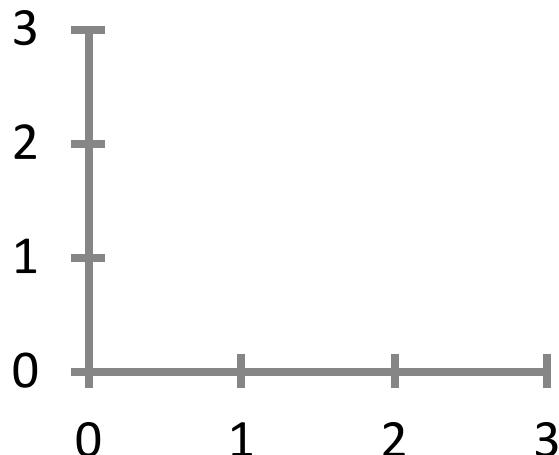
Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

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

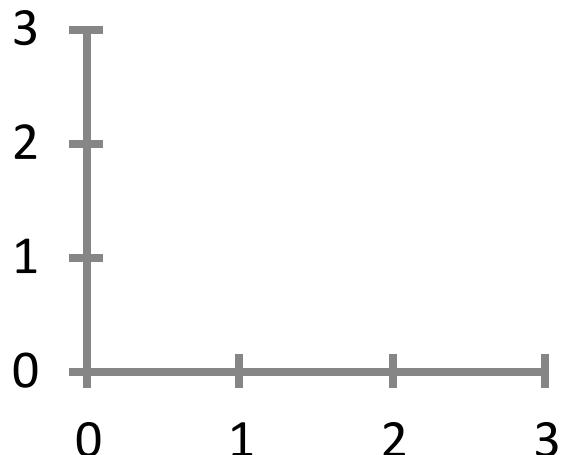
θ_i 's: Parameters

How to choose θ_i 's ?

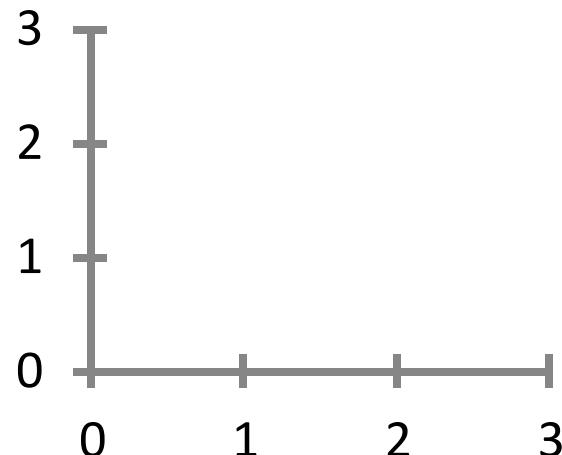
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



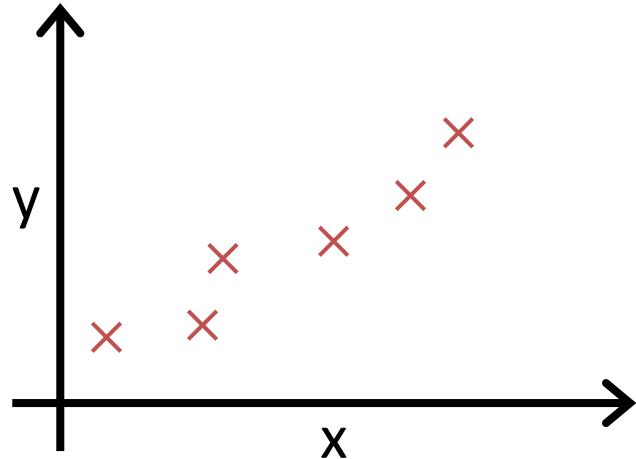
$$\begin{aligned}\theta_0 &= 1.5 \\ \theta_1 &= 0\end{aligned}$$



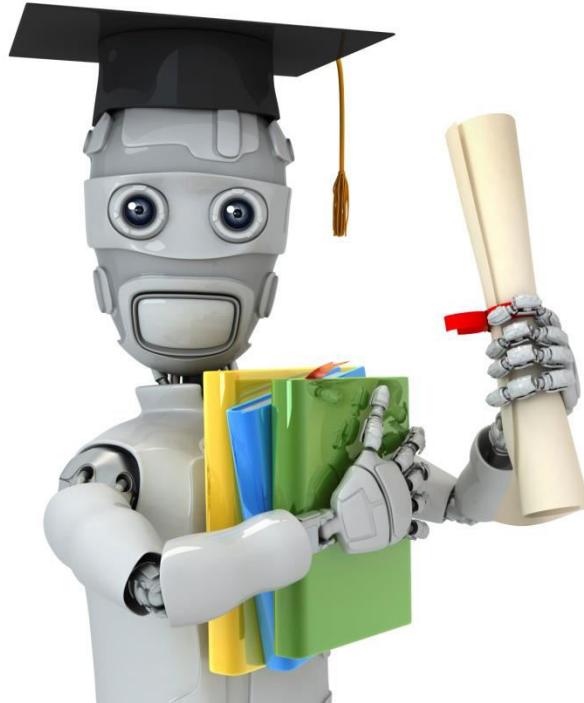
$$\begin{aligned}\theta_0 &= 0 \\ \theta_1 &= 0.5\end{aligned}$$



$$\begin{aligned}\theta_0 &= 1 \\ \theta_1 &= 0.5\end{aligned}$$



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



Machine Learning

Linear regression with one variable

Cost function intuition I

Simplified

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

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

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

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

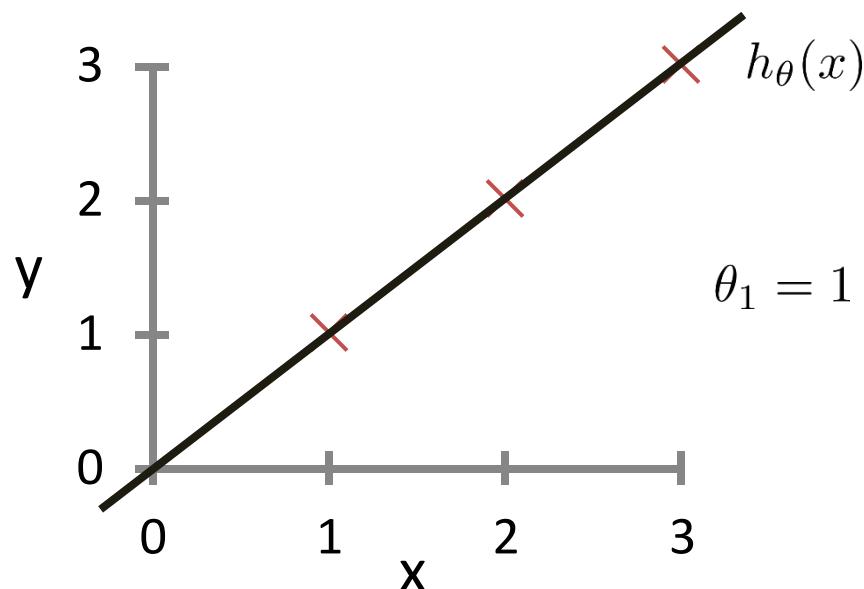
$$\theta_1$$

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

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

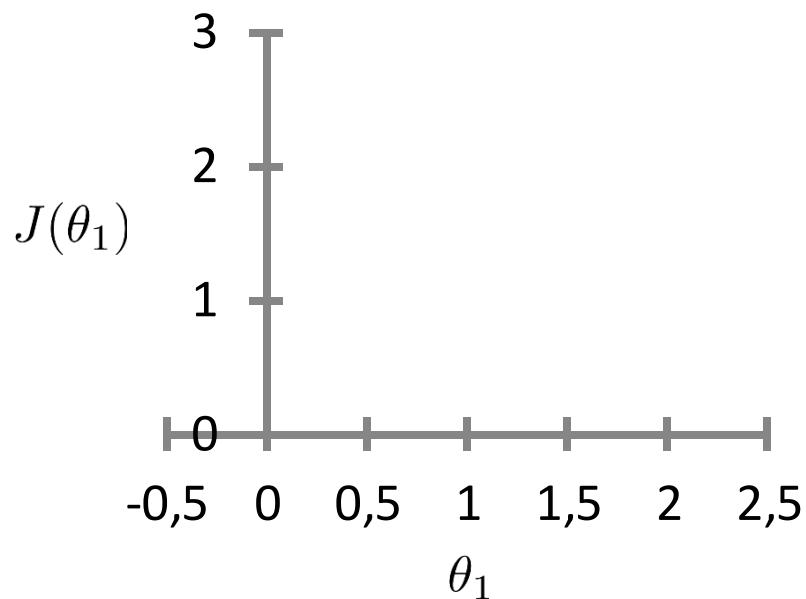
$$h_{\theta}(x)$$

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

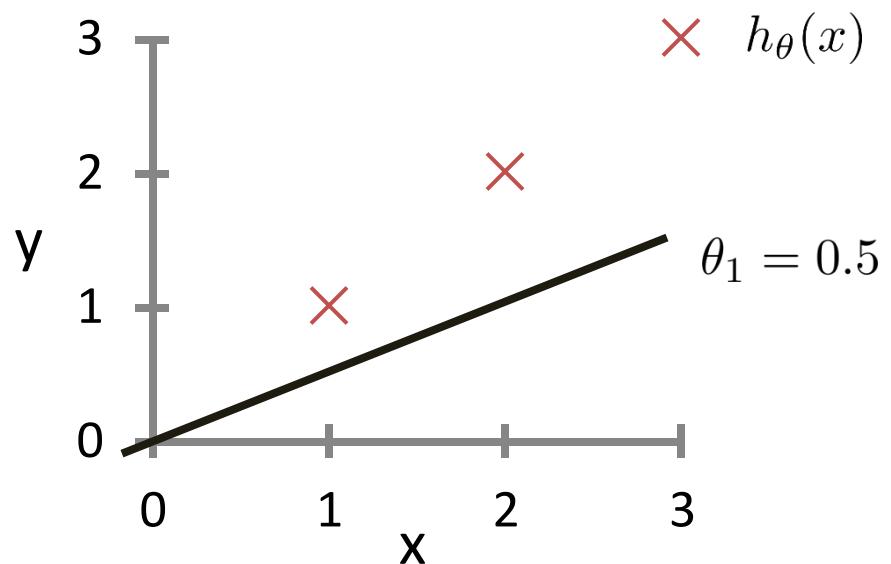


$$J(\theta_1)$$

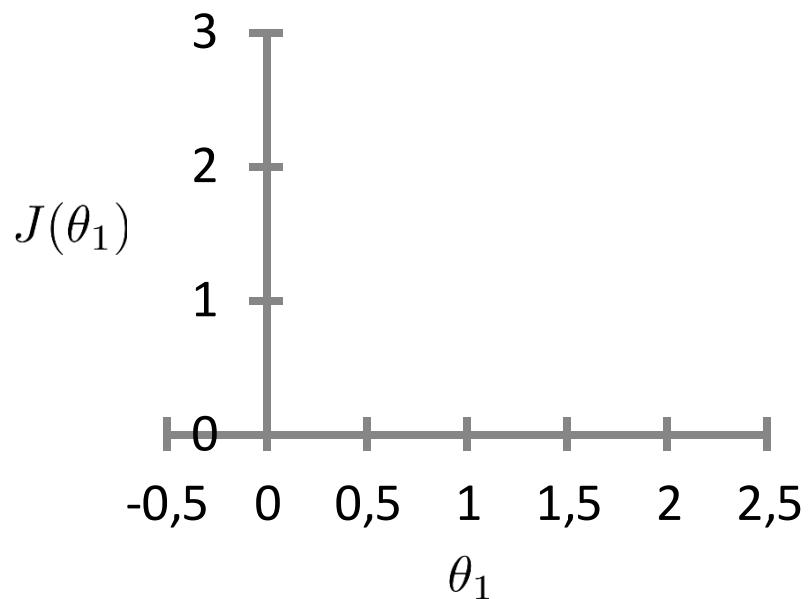
(function of the parameter θ_1)



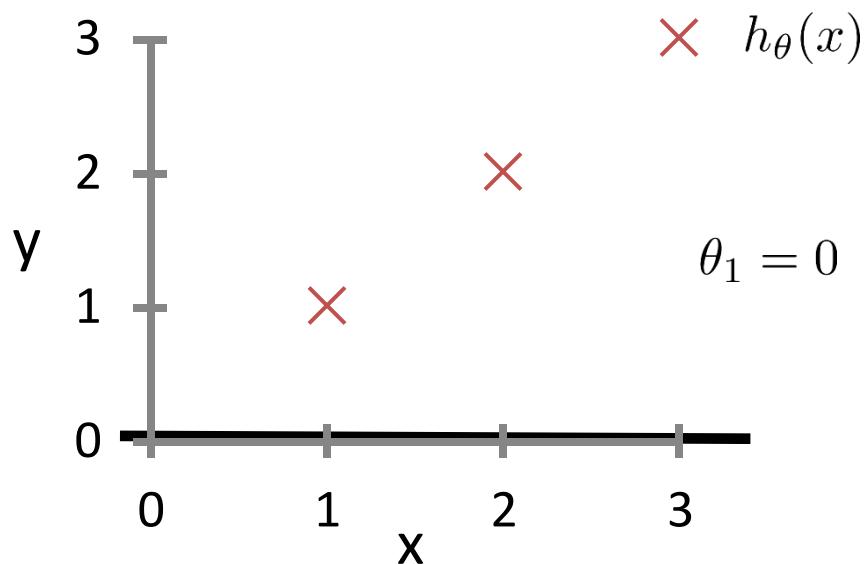
$h_{\theta}(x)$
(for fixed θ_1 , this is a function of x)



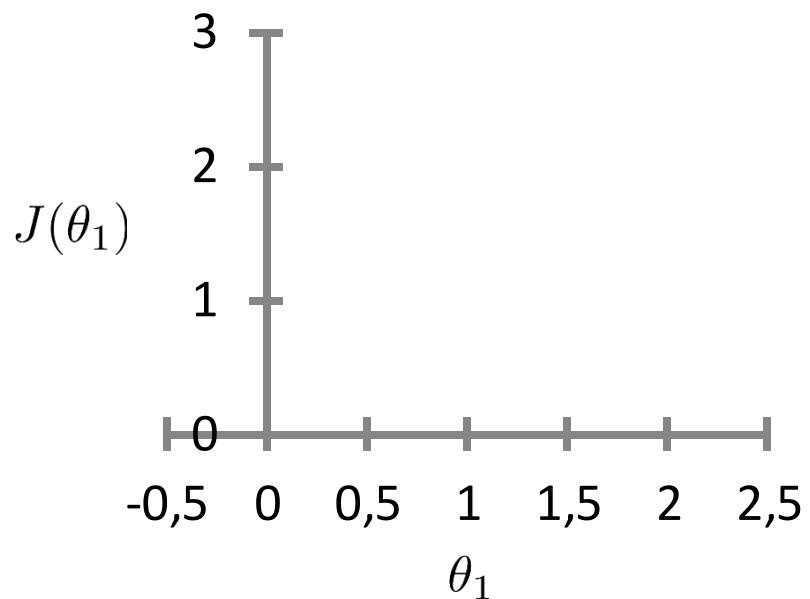
$J(\theta_1)$
(function of the parameter θ_1)

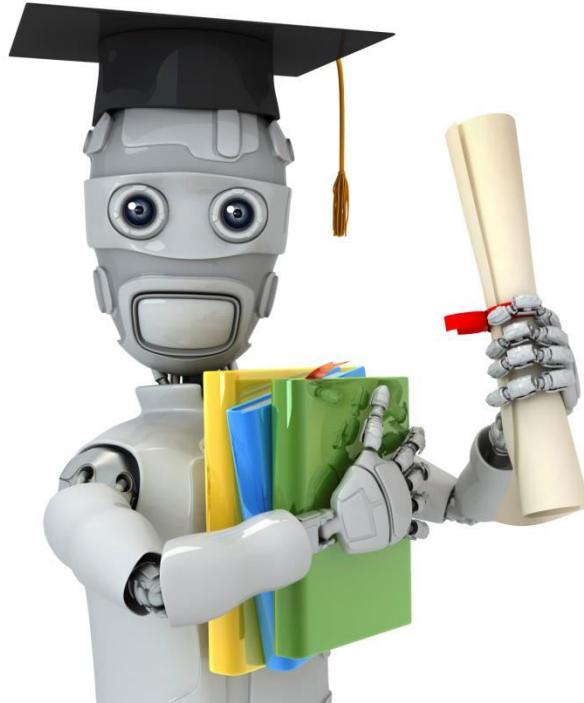


$h_{\theta}(x)$
(for fixed θ_1 , this is a function of x)



$J(\theta_1)$
(function of the parameter θ_1)





Machine Learning

Linear regression with one variable

Cost function intuition II

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

Parameters: θ_0, θ_1

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

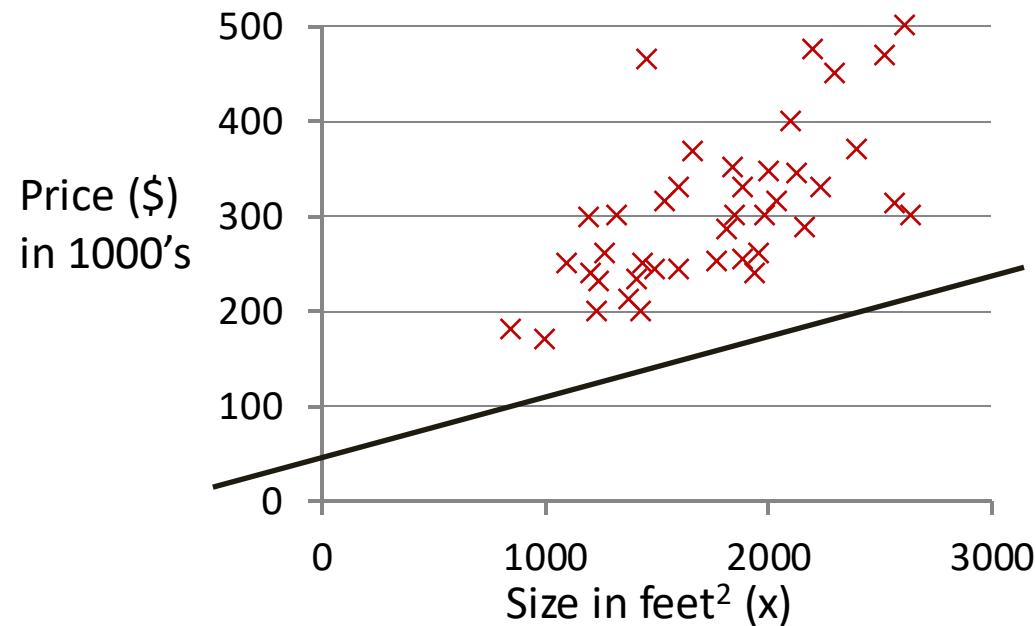
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$

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

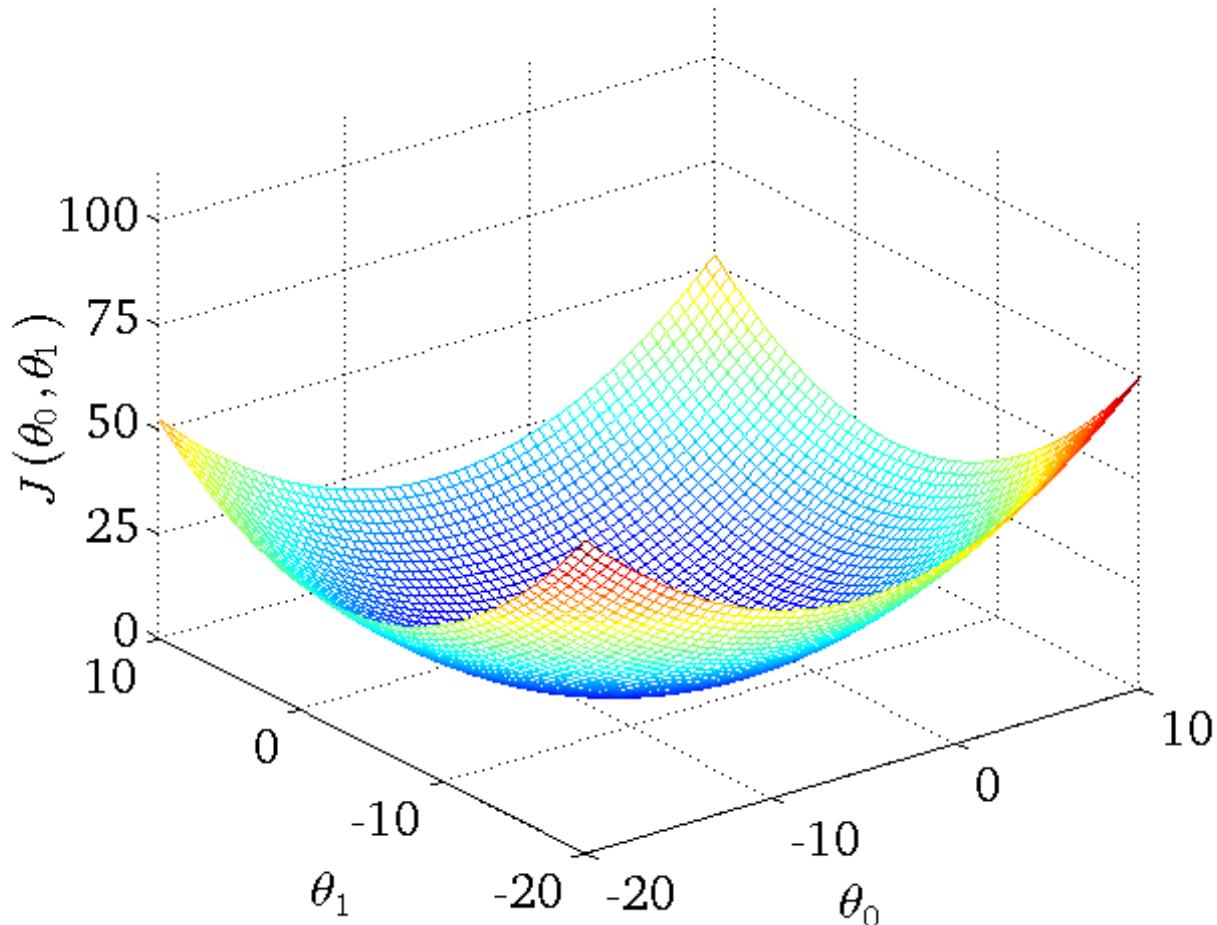
(for fixed θ_0, θ_1 , this is a function of x)

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

(function of the parameters θ_0, θ_1)

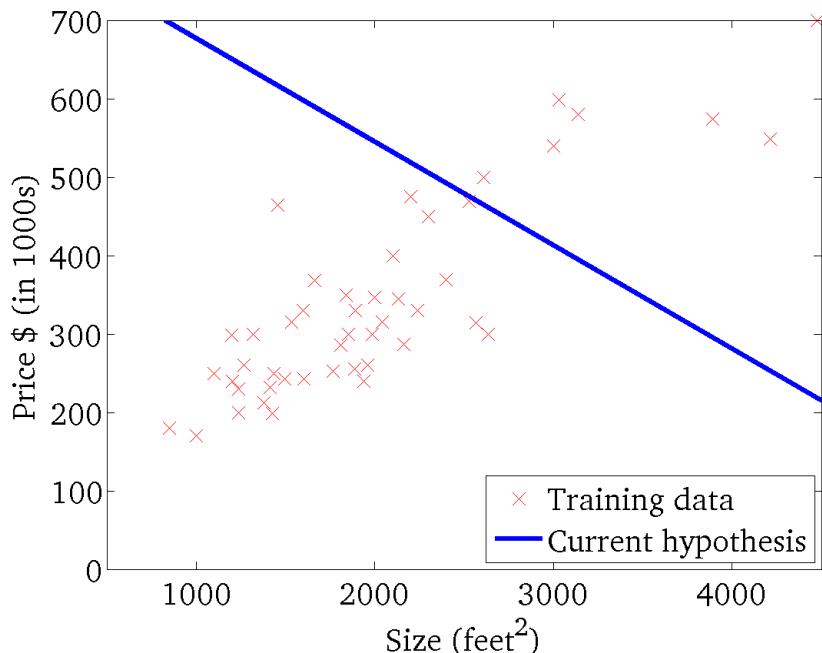


$$h_{\theta}(x) = 50 + 0.06x$$



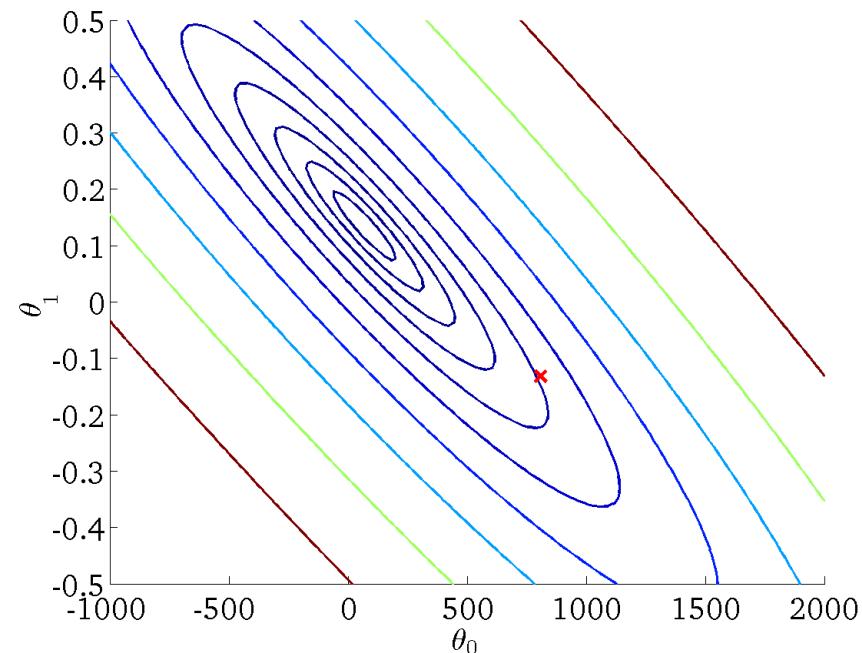
$$h_{\theta}(x)$$

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



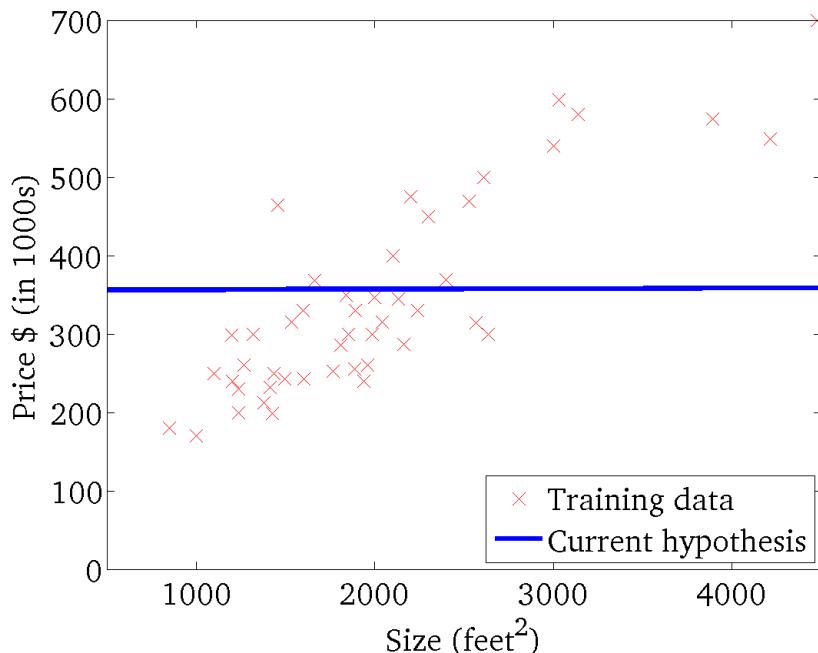
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



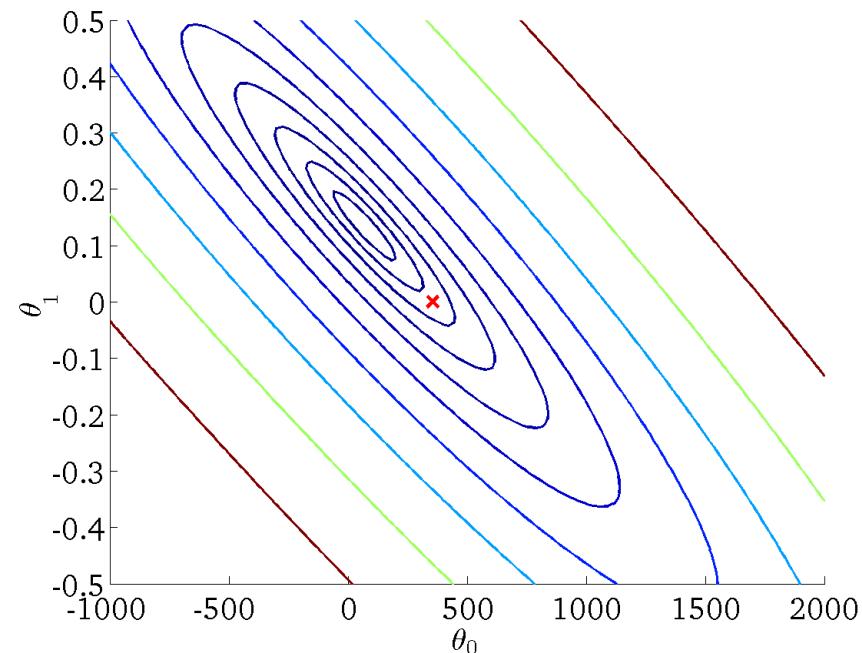
$$h_{\theta}(x)$$

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



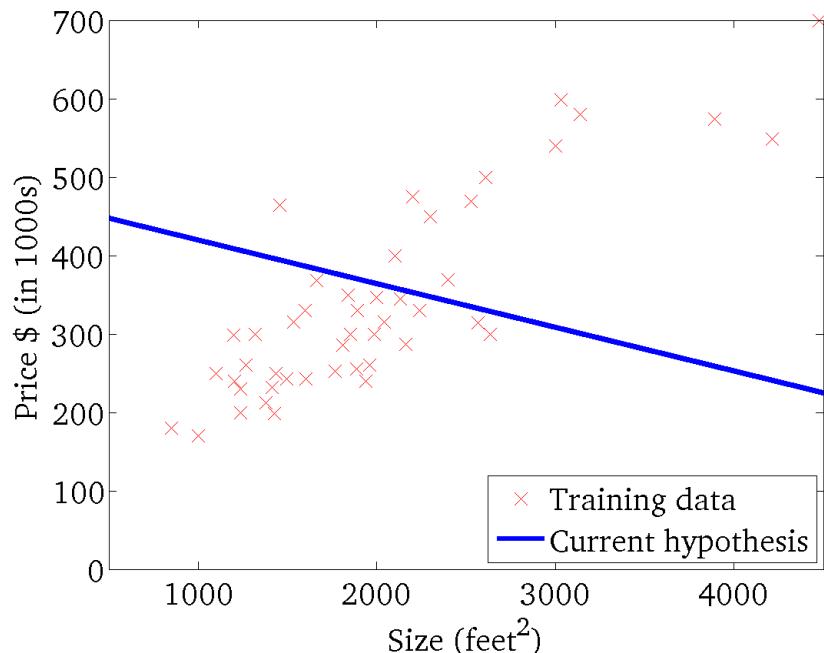
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



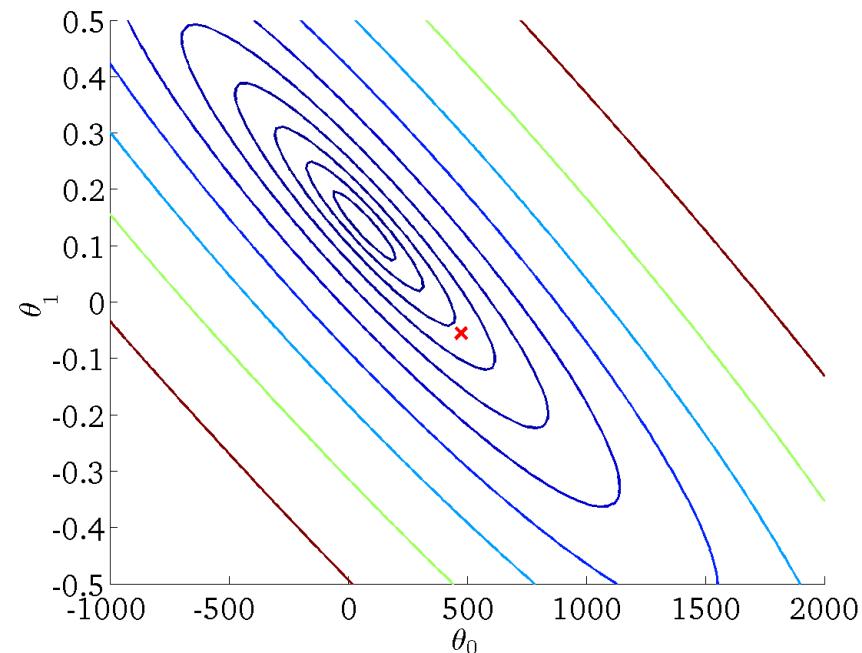
$$h_{\theta}(x)$$

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



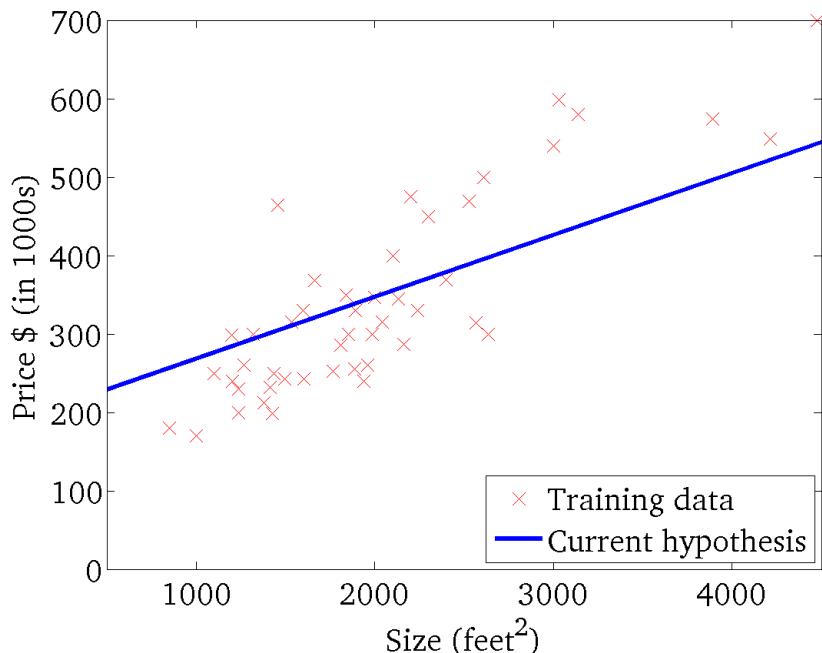
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



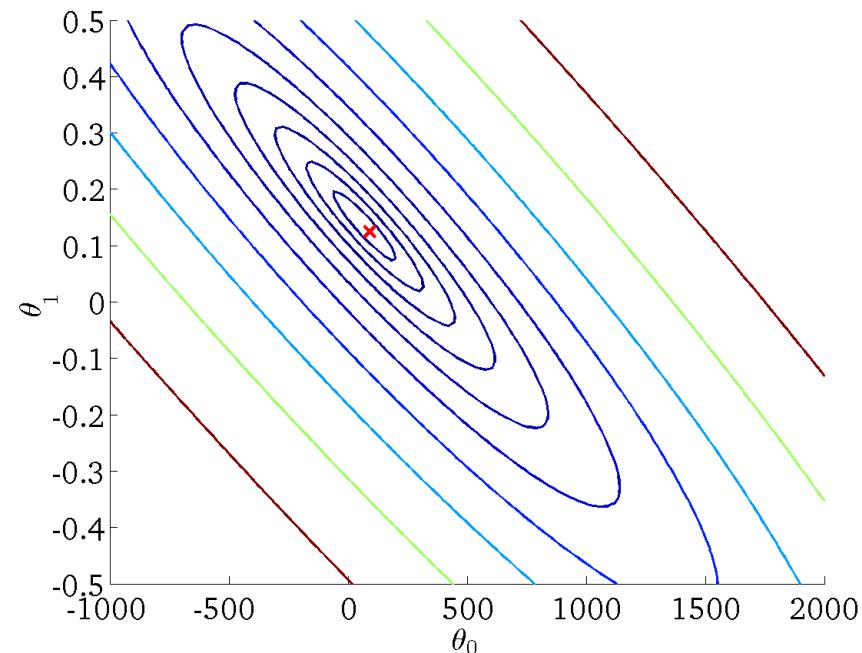
$$h_{\theta}(x)$$

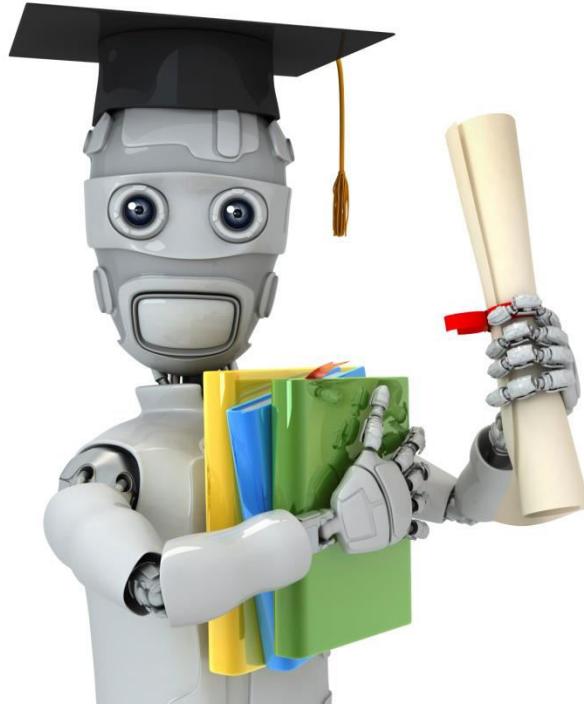
(for fixed θ_0, θ_1 , this is a function of x)



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

(function of the parameters θ_0, θ_1)





Machine Learning

Linear regression
with one variable

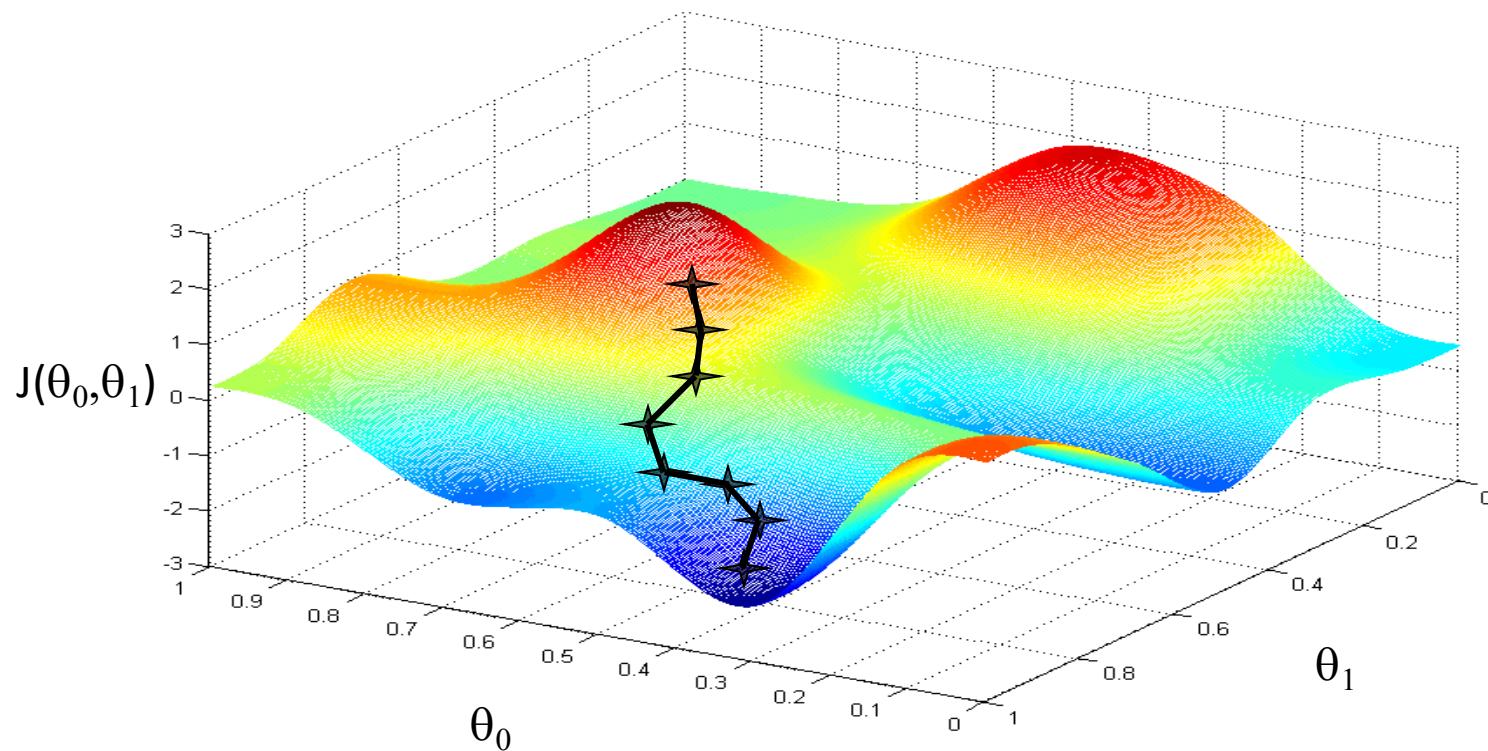
Gradient
descent

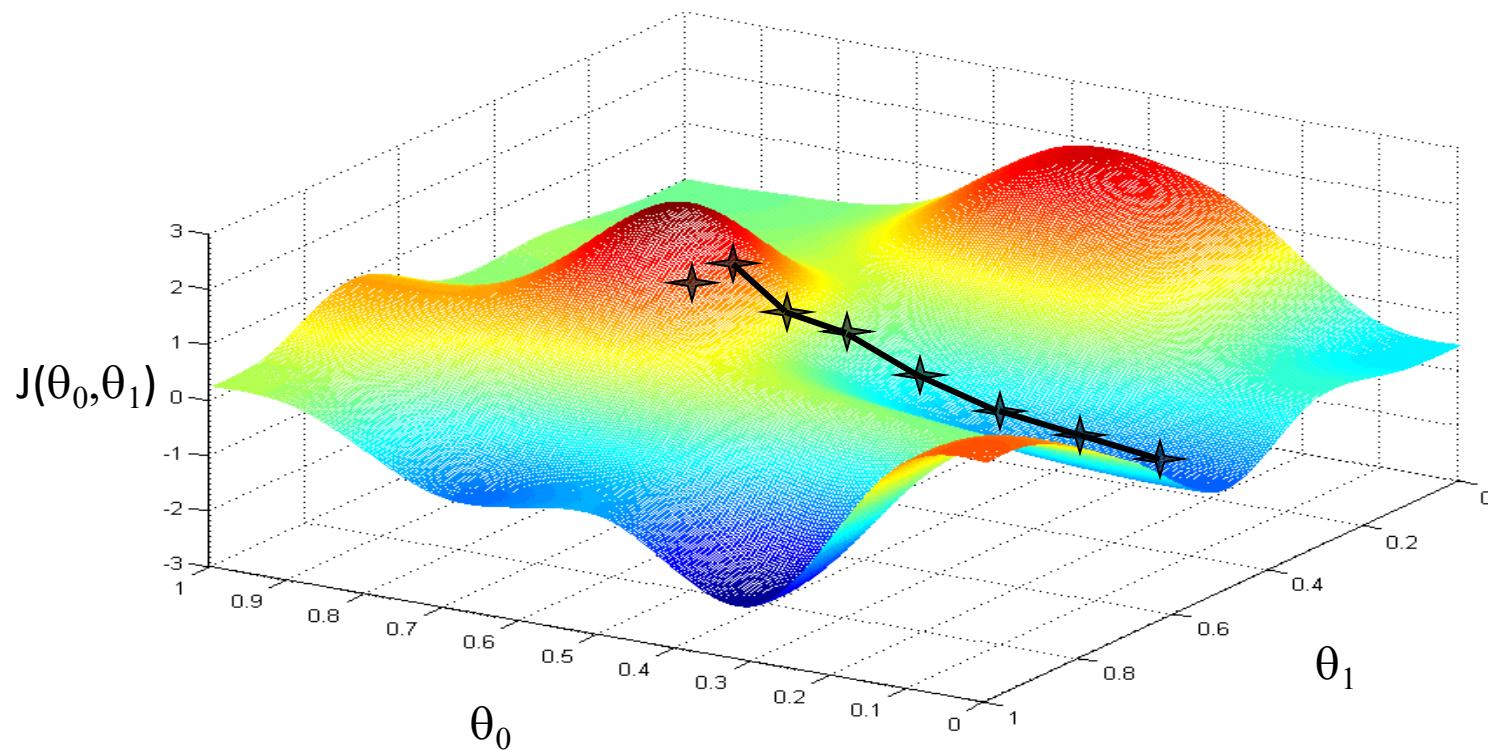
Have some function $J(\theta_0, \theta_1)$

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 algorithm

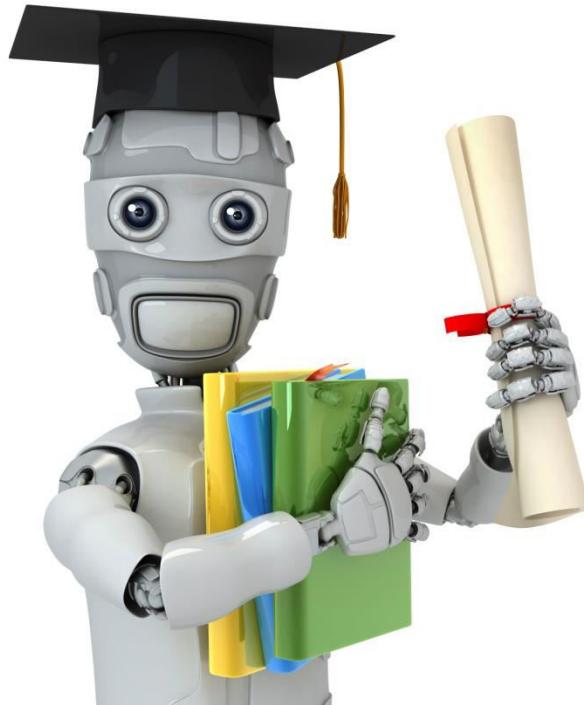
```
repeat until convergence {  
     $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$     (for  $j = 0$  and  $j = 1$ )  
}
```

Correct: Simultaneous update

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

Incorrect:

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



Machine Learning

Linear regression
with one variable

Gradient descent
intuition

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \begin{matrix} \text{(simultaneously update} \\ j = 0 \text{ and } j = 1 \end{matrix}$$

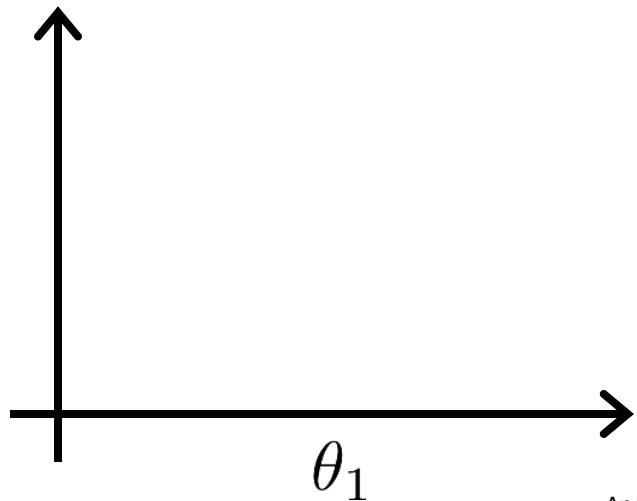
}

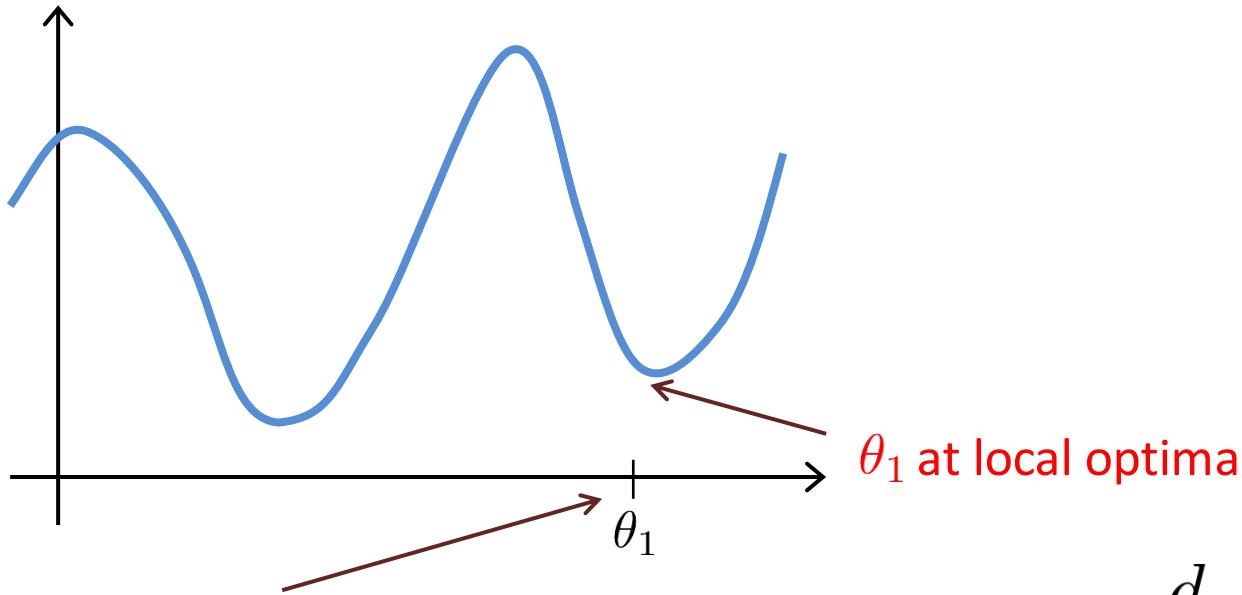


$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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.





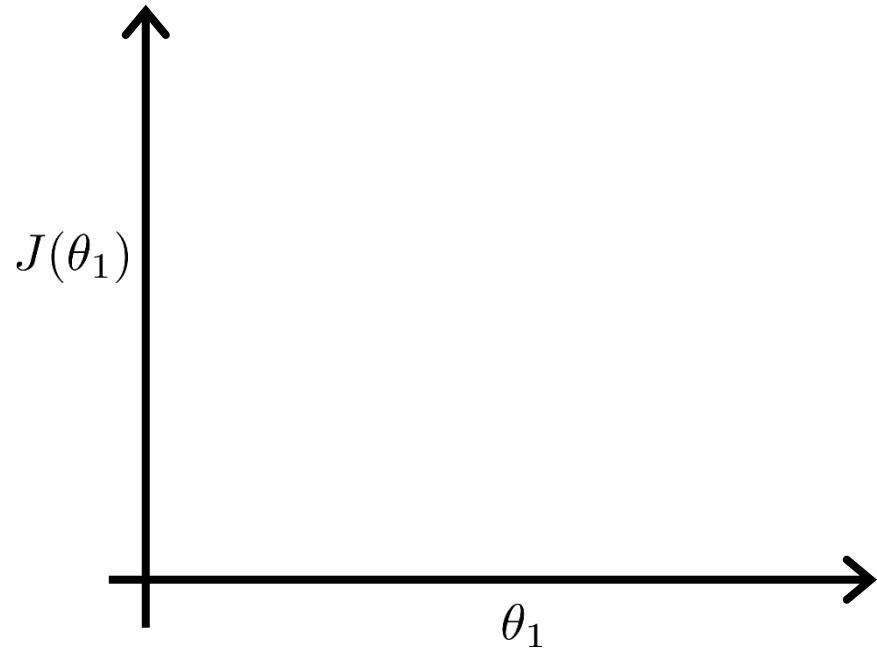
Current value of θ_1

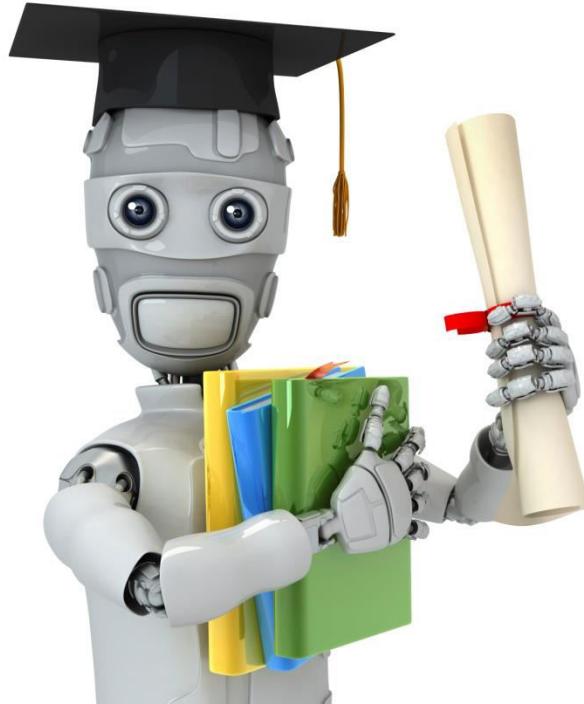
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

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.





Machine Learning

Linear regression with one variable

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  $j = 1$  and  $j = 0$ )  
}
```

Linear Regression Model

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

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

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

$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) =$$

$$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) =$$

Gradient descent algorithm

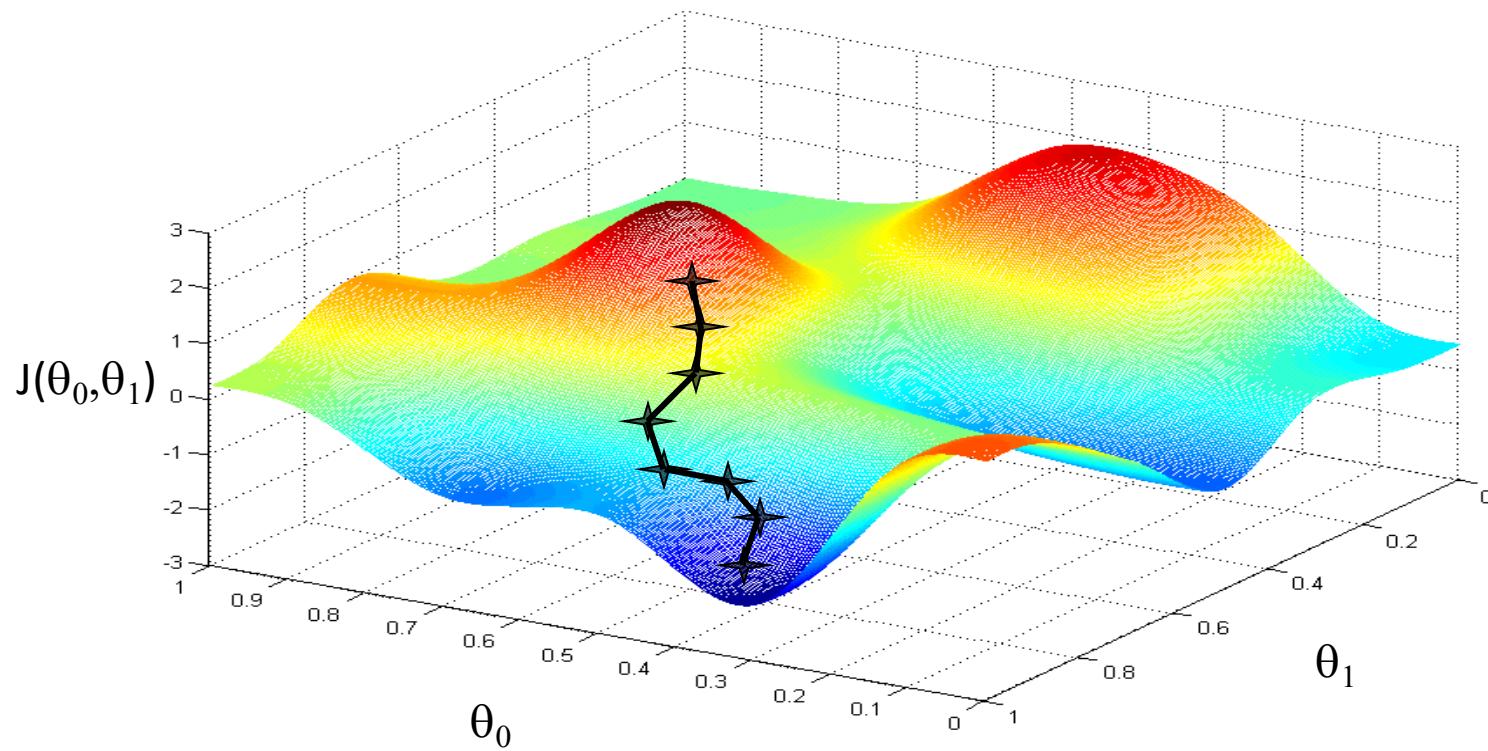
repeat until convergence {

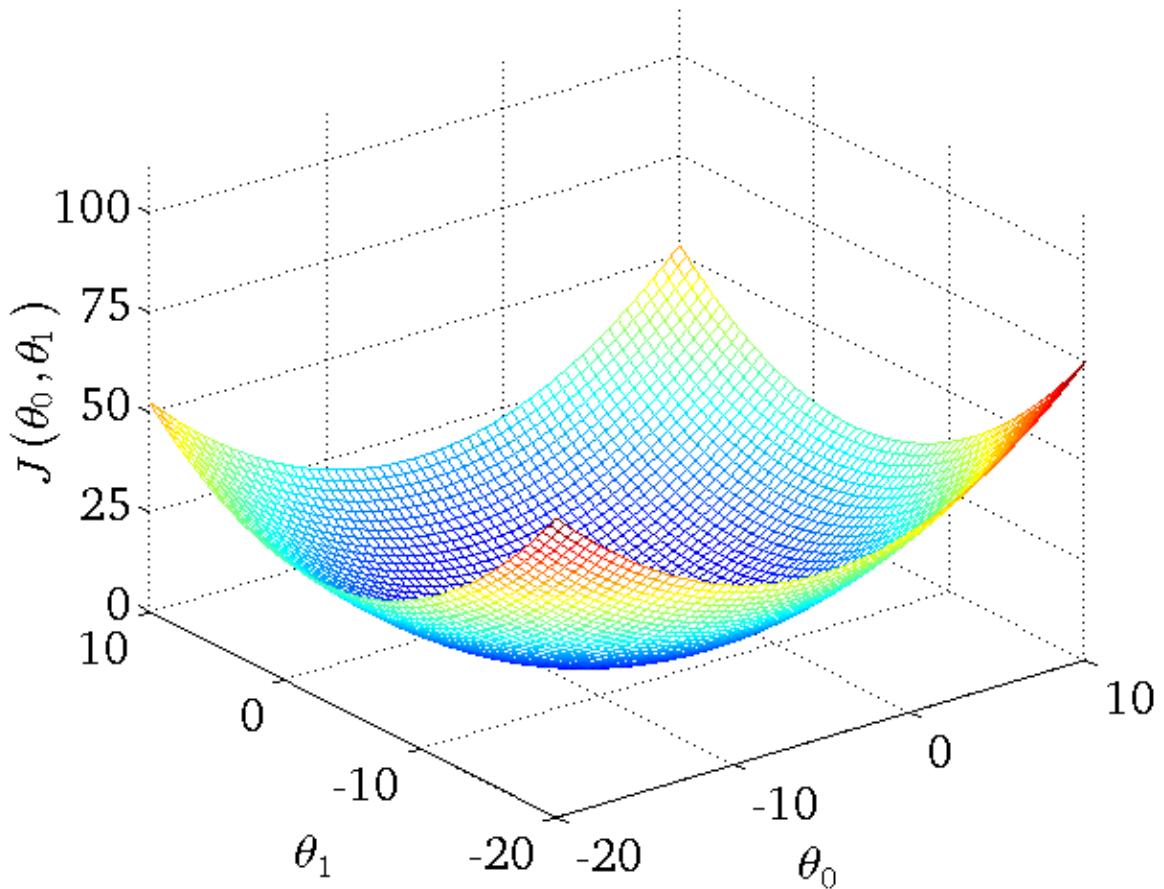
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$$

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

}

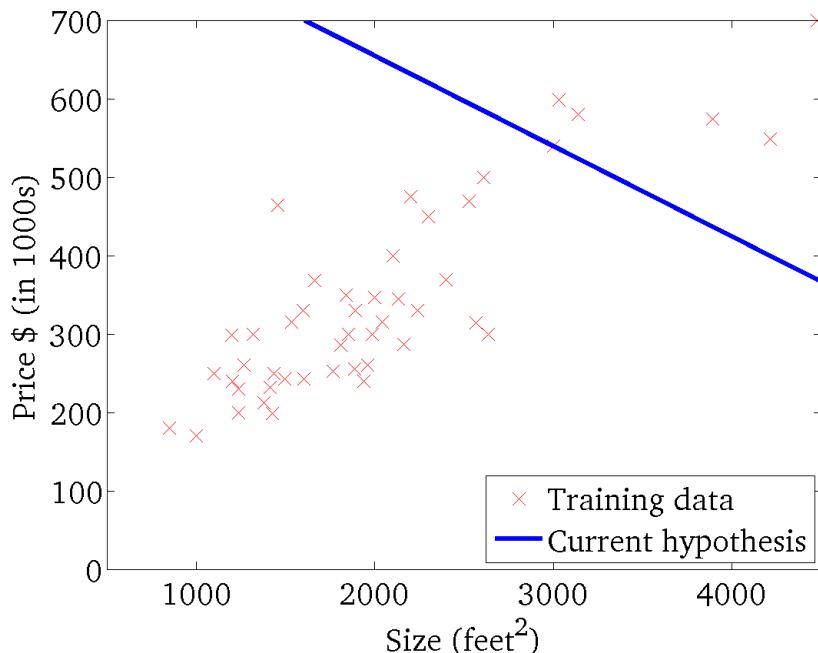
} update
 θ_0 and θ_1
simultaneously





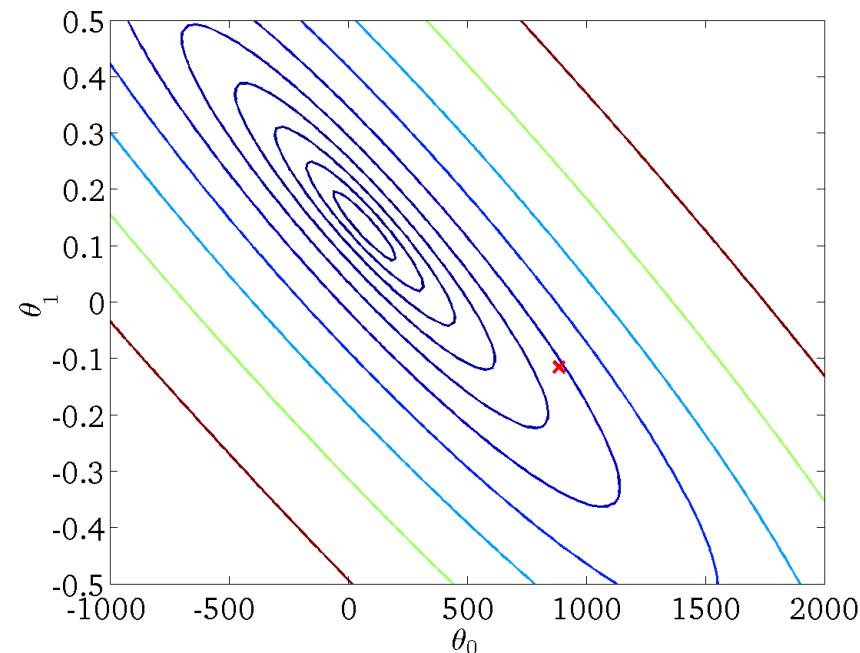
$$h_{\theta}(x)$$

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



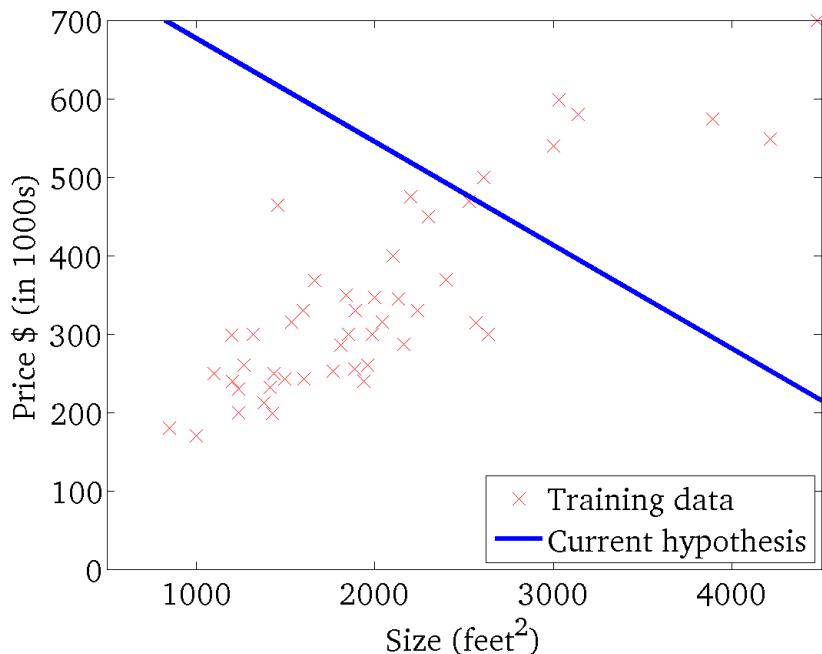
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



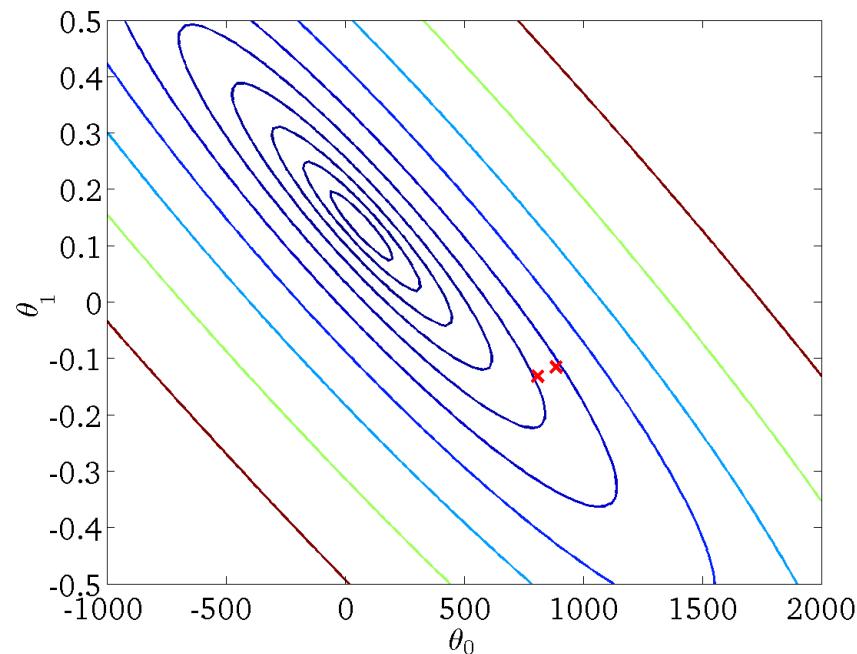
$$h_{\theta}(x)$$

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



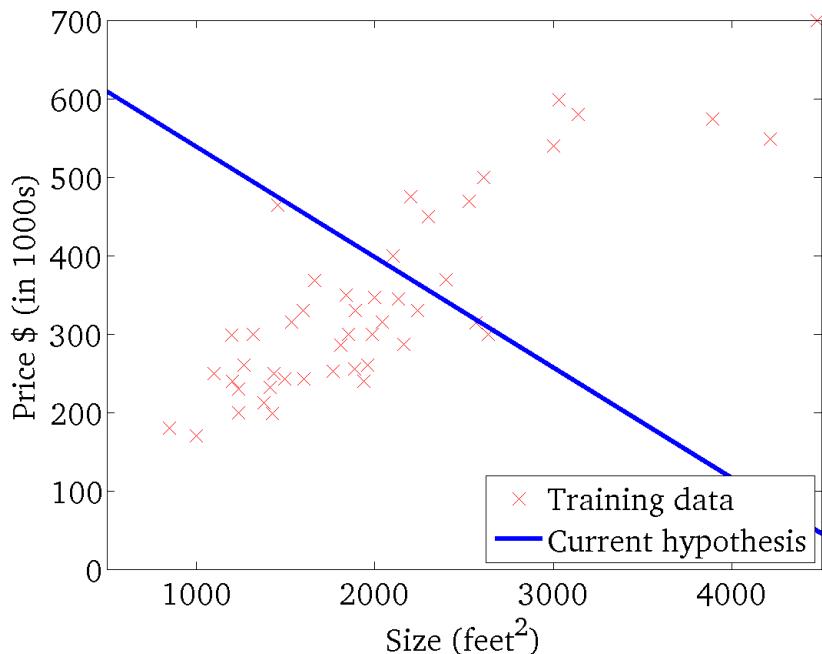
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



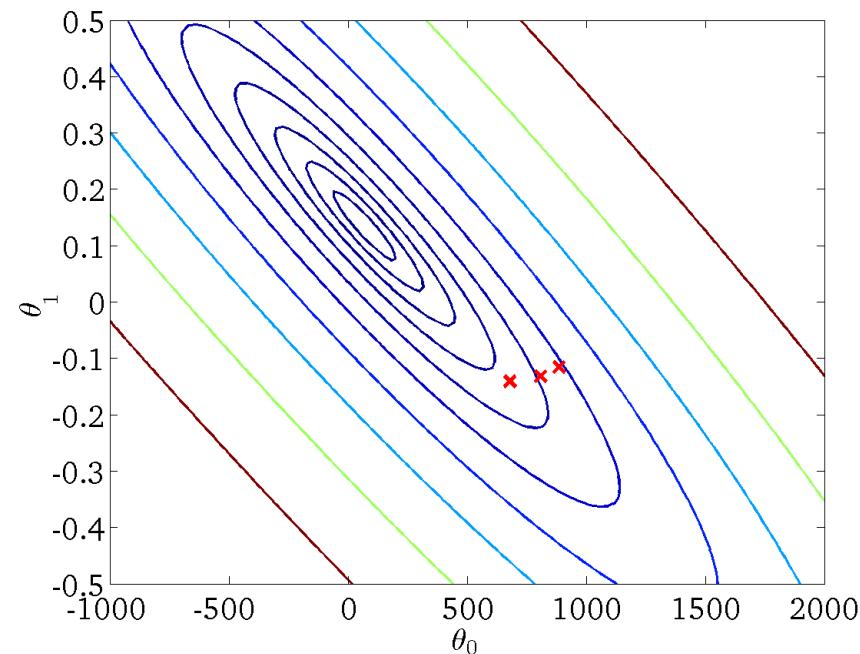
$$h_{\theta}(x)$$

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



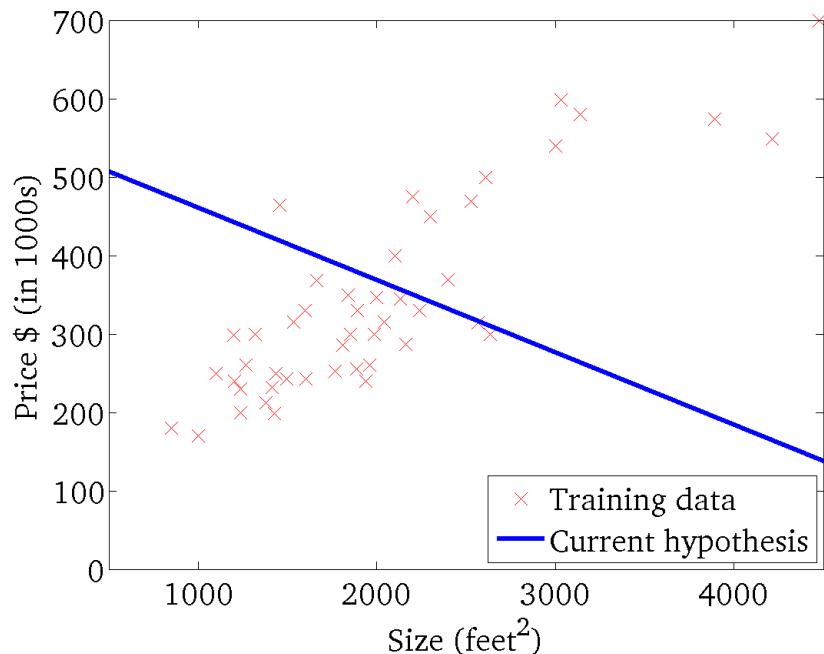
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



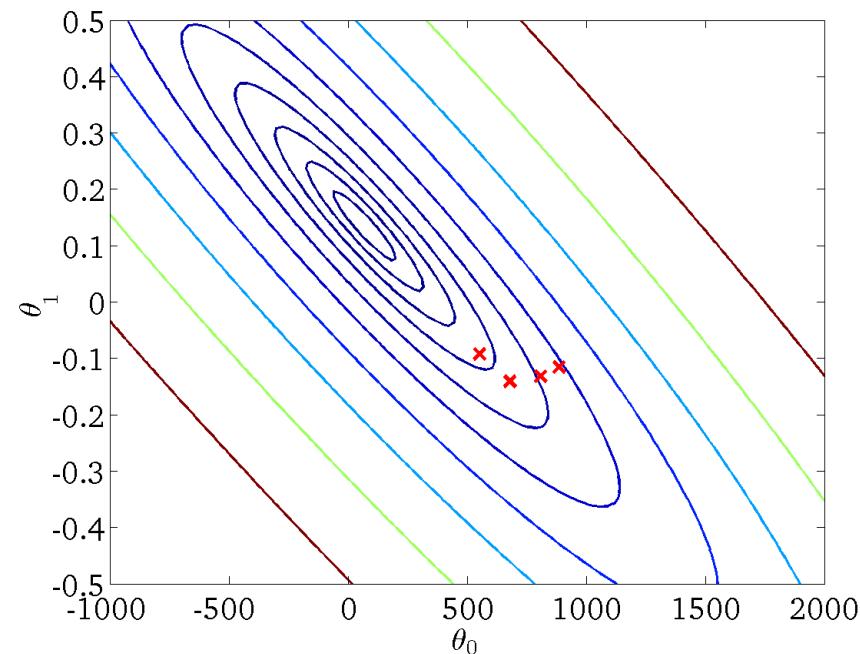
$$h_{\theta}(x)$$

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



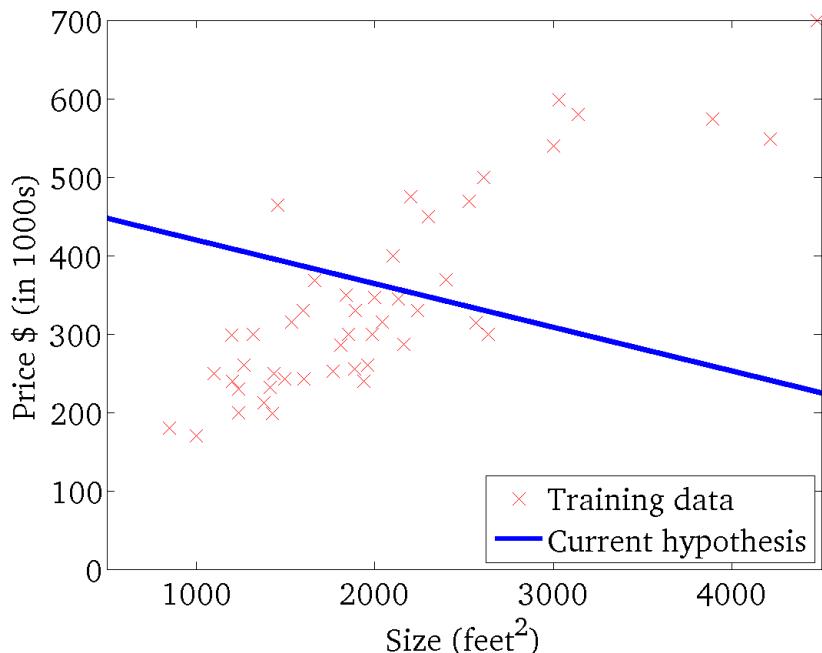
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



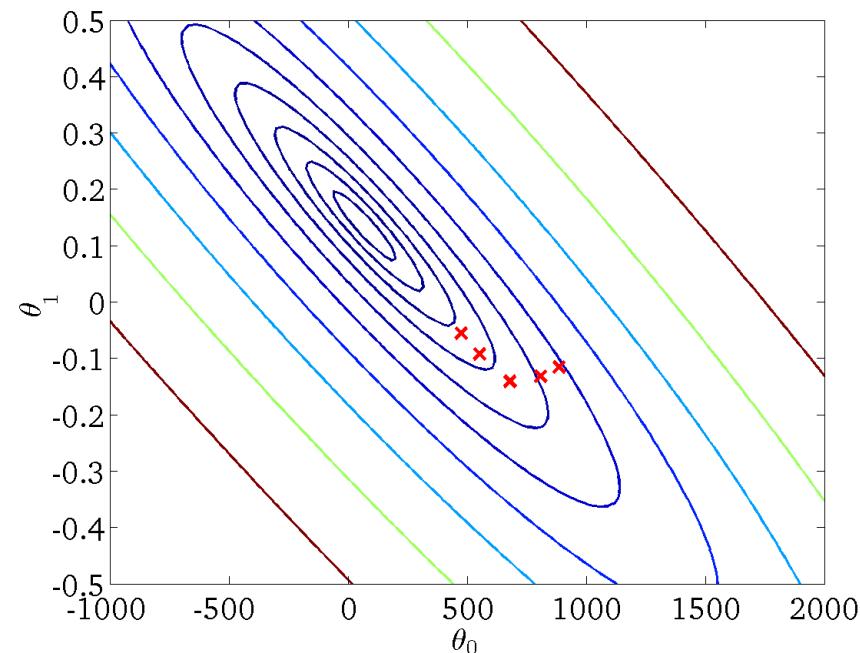
$$h_{\theta}(x)$$

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



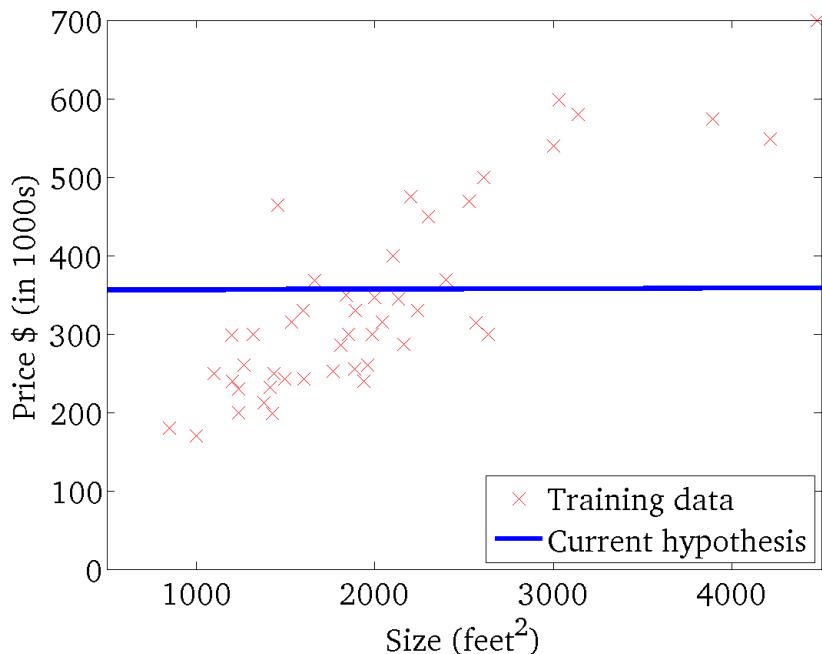
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



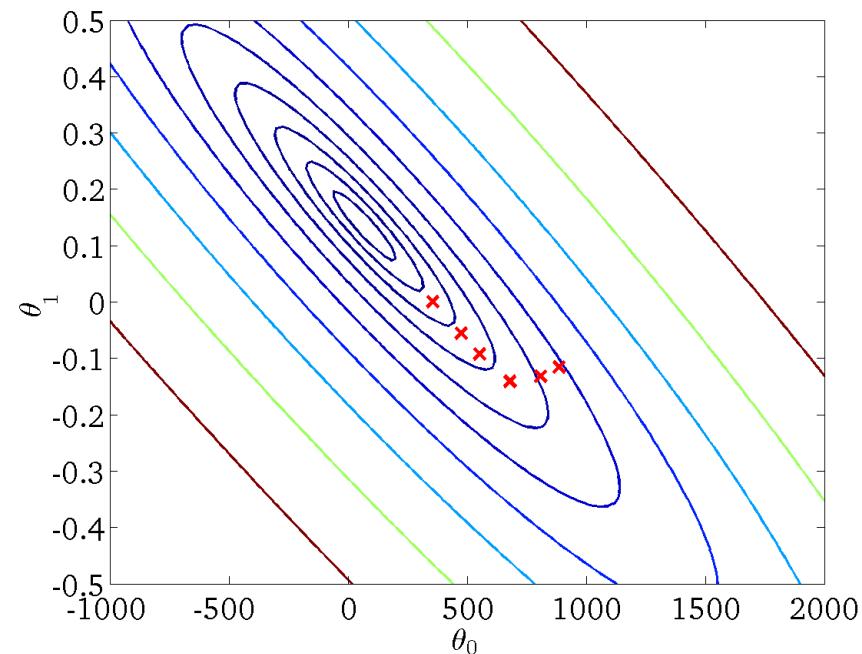
$$h_{\theta}(x)$$

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



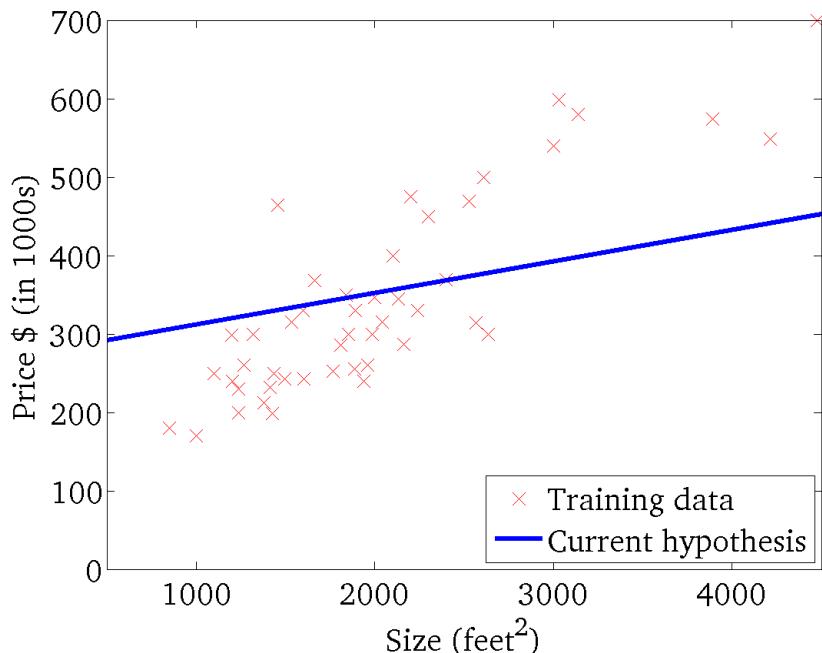
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



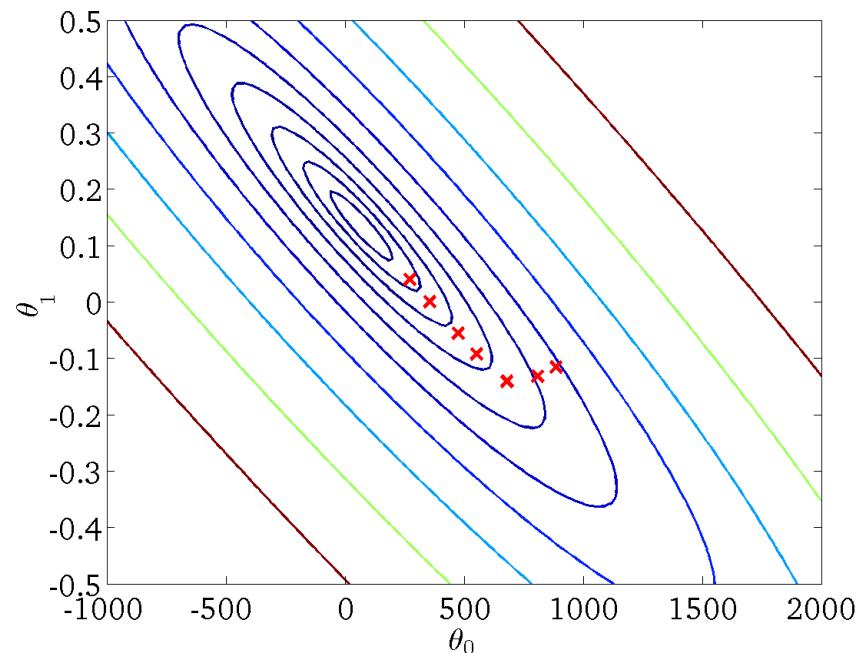
$$h_{\theta}(x)$$

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



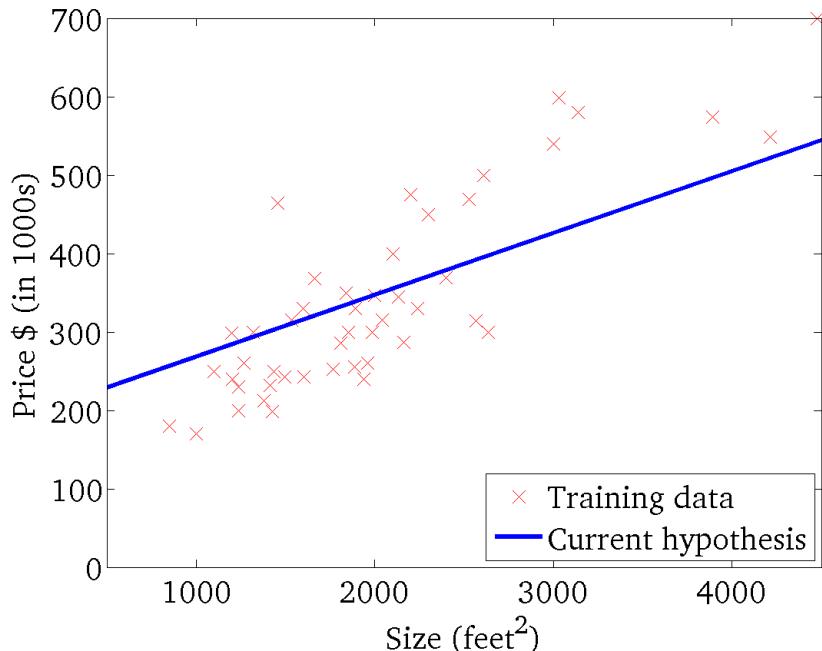
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



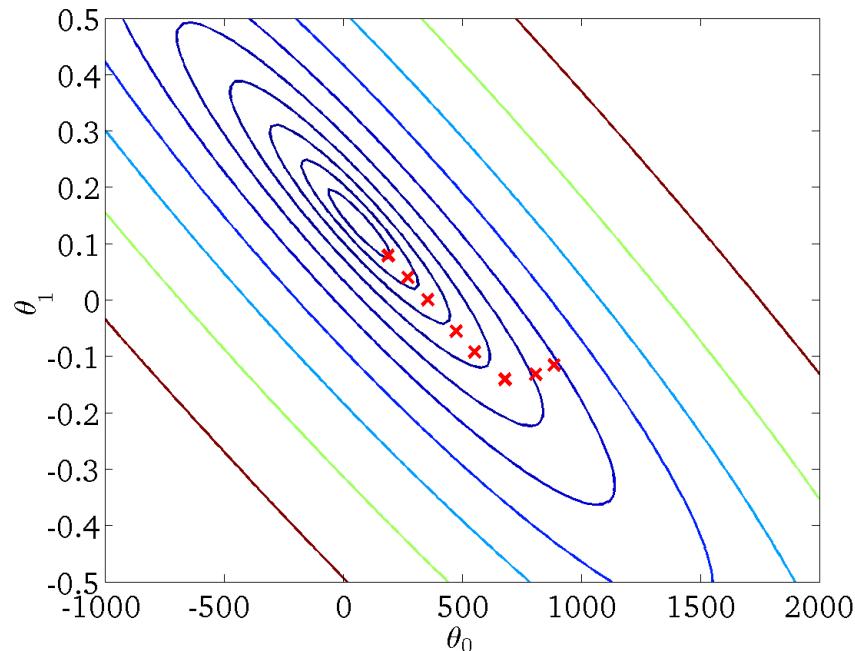
$$h_{\theta}(x)$$

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



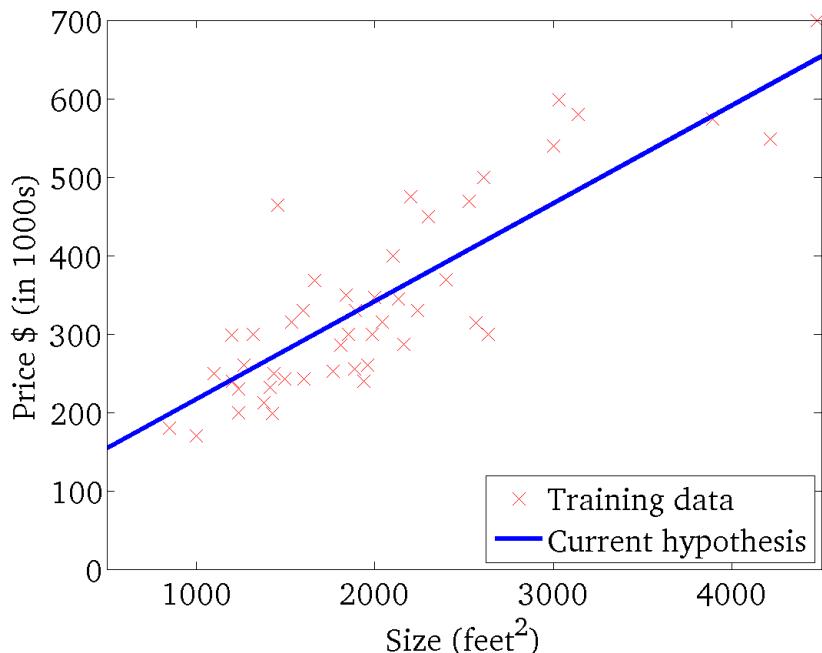
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



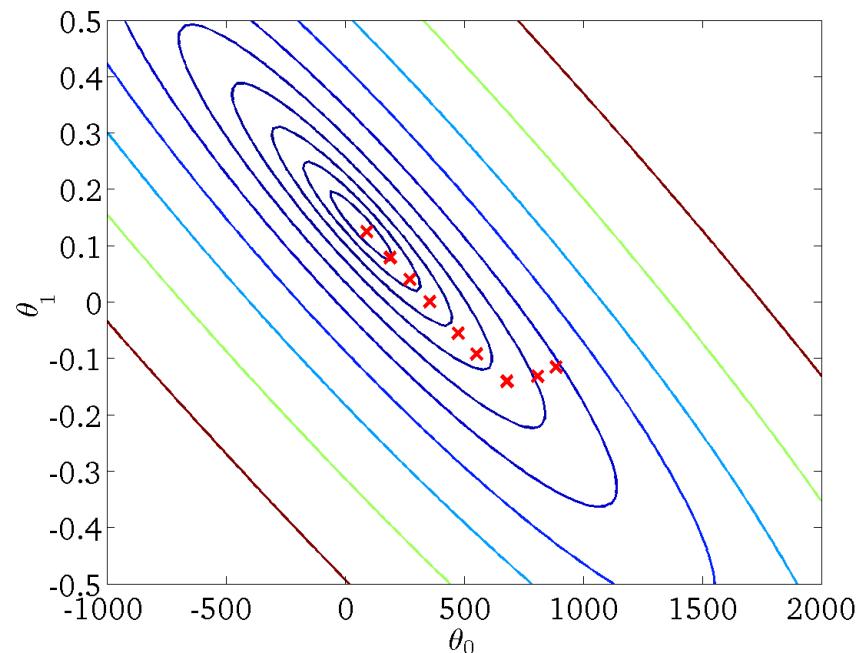
$$h_{\theta}(x)$$

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



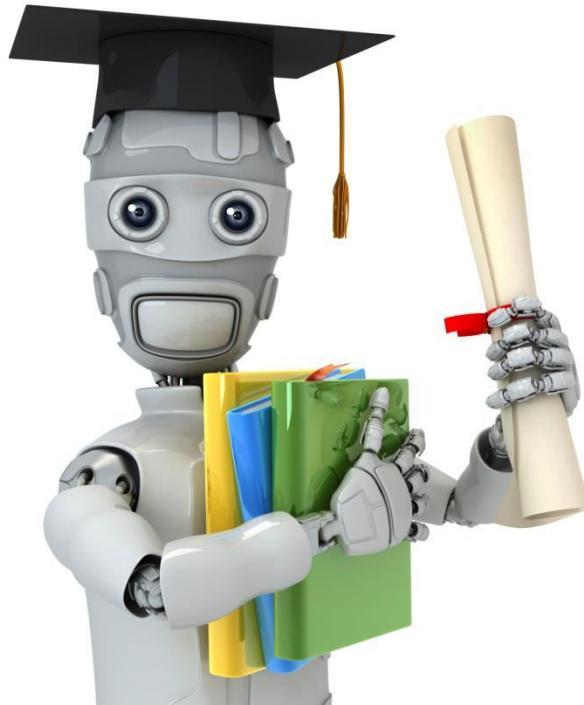
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



“Batch” Gradient Descent

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



Machine Learning

Linear Algebra review (optional)

Matrices and vectors

Matrix: Rectangular array of numbers:

Dimension of matrix: number of rows x number of columns

Matrix Elements (entries of matrix)

$$A = \begin{bmatrix} 1402 & 191 \\ 1371 & 821 \\ 949 & 1437 \\ 147 & 1448 \end{bmatrix}$$

A_{ij} = “ i, j entry” in the i^{th} row, j^{th} column.

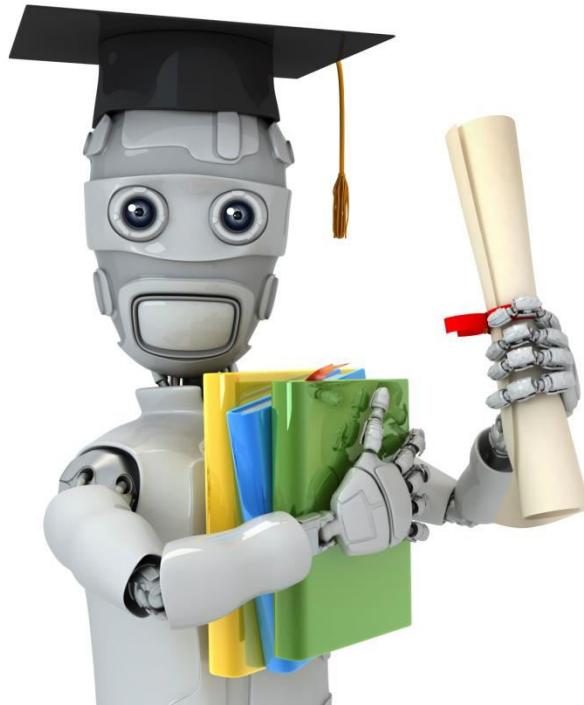
Vector: An $n \times 1$ matrix.

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$y_i = i^{th}$ element

1-indexed vs 0-indexed:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$



Machine Learning

Linear Algebra review (optional)

Addition and scalar multiplication

Matrix Addition

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \\ 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \\ 0 & 1 \end{bmatrix} =$$

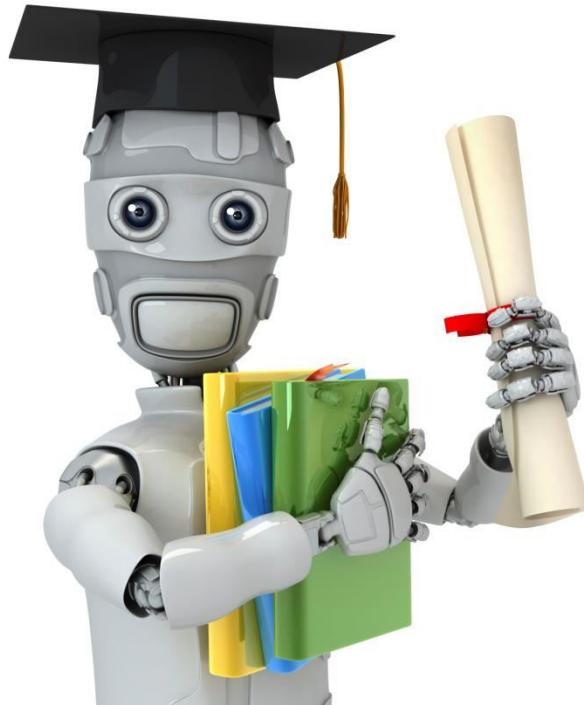
Scalar Multiplication

$$3 \times \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} / 4 =$$

Combination of Operands

$$3 \times \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} - \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix} / 3$$



Machine Learning

Linear Algebra review (optional)

Matrix-vector multiplication

Example

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \end{bmatrix} =$$

Details:

$$A \times x = y$$
$$\left[\quad \right] \times \left[\quad \right] = \left[\quad \right]$$

$m \times n$ matrix
(m rows,
 n columns) $n \times 1$ matrix
(n -dimensional
vector) m -dimensional
vector

To get y_i , multiply A 's i^{th} row with elements of vector x , and add them up.

Example

$$\begin{bmatrix} 1 & 2 & 1 & 5 \\ 0 & 3 & 0 & 4 \\ -1 & -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} =$$

House sizes:

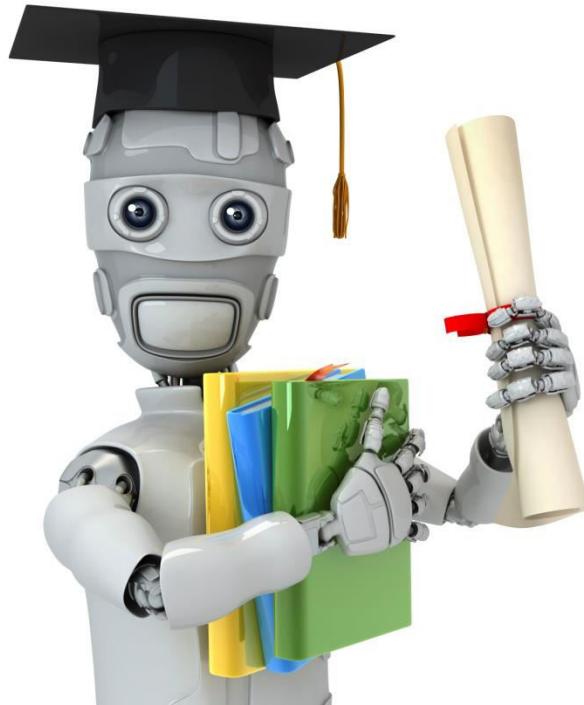
2104

$$h_{\theta}(x) = -40 + 0.25x$$

1416

1534

852



Machine Learning

Linear Algebra review (optional)

Matrix-matrix multiplication

Example

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} =$$

Details:

$$A \times B = C$$
$$\left[\quad \right] \times \left[\quad \right] = \left[\quad \right]$$

$m \times n$ matrix
(m rows,
 n columns)

$n \times o$ matrix
(n rows,
 o columns)

$m \times o$
matrix

The i^{th} column of the matrix C is obtained by multiplying A with the i^{th} column of B . (for $i = 1, 2, \dots, o$)

Example

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix} = \quad 7$$

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \quad 2 \quad 7$$

House sizes:

2104

1416

1534

852

Have 3 competing hypotheses:

$$1. h_{\theta}(x) = -40 + 0.25x$$

$$2. h_{\theta}(x) = 200 + 0.1x$$

$$3. h_{\theta}(x) = -150 + 0.4x$$

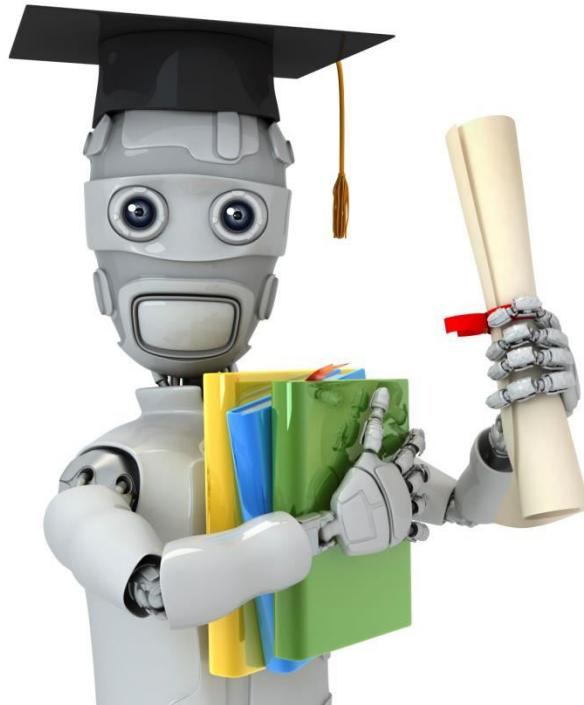
Matrix

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix}$$

Matrix

$$\times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} =$$

$$\begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$



Machine Learning

Linear Algebra review (optional)

Matrix multiplication properties

Let A and B be matrices. Then in general,
 $A \times B \neq B \times A$. (not commutative.)

E.g. $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$

$$\begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \end{bmatrix}$$

$A \times B \times C.$

Let $D = B \times C$. Compute $A \times D$.

Let $E = A \times B$. Compute $E \times C$.

Identity Matrix

Denoted I (or $I_{n \times n}$).

Examples of identity matrices:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2×2

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

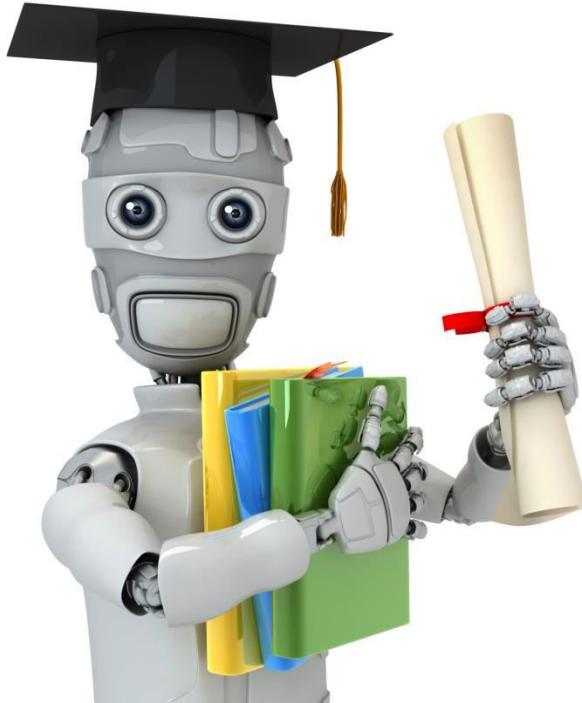
3×3

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4×4

For any matrix A ,

$$A \cdot I = I \cdot A = A$$



Machine Learning

Linear Algebra review (optional)

Inverse and transpose

Not all numbers have an inverse.

Matrix inverse:

If A is an $m \times m$ matrix, and if it has an inverse,

$$AA^{-1} = A^{-1}A = I.$$

Matrices that don't have an inverse are “singular” or “degenerate”

Matrix Transpose

Example:

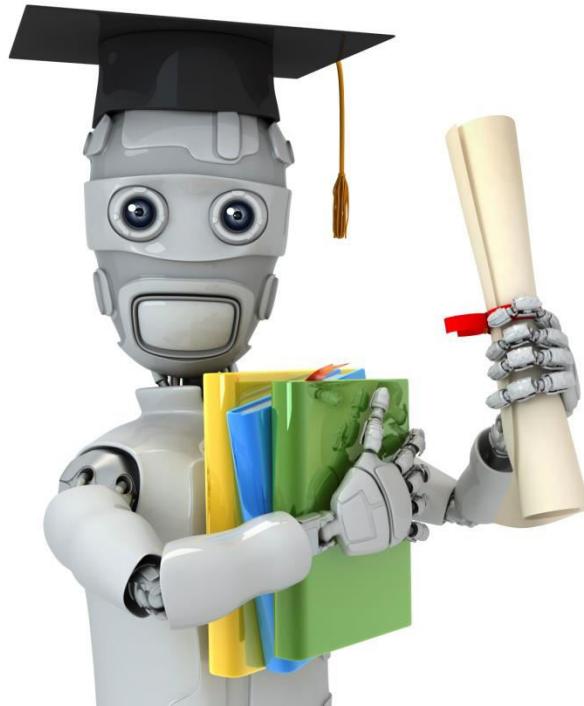
$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 5 & 9 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 0 & 9 \end{bmatrix}$$

Let A be an $m \times n$ matrix, and let $B = A^T$.

Then B is an $n \times m$ matrix, and

$$B_{ij} = A_{ji}.$$



Machine Learning

Linear Regression with
multiple variables

Multiple features

Multiple features (variables).

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

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

Multiple features (variables).

Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
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.

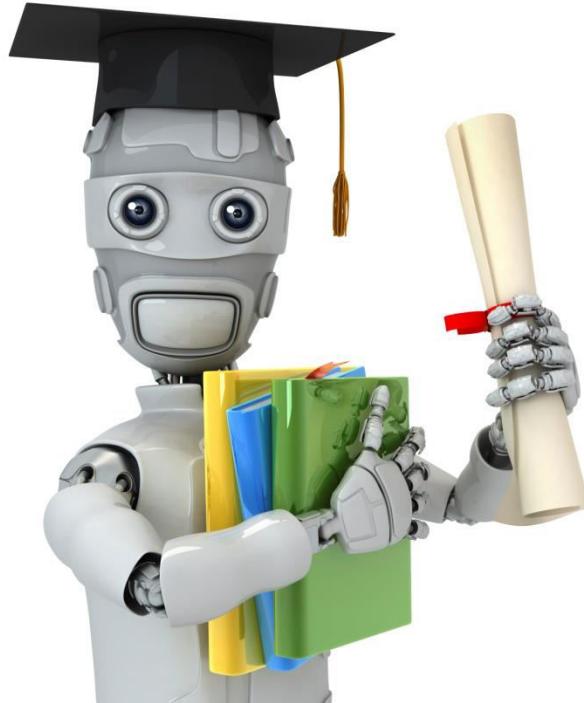
Hypothesis:

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

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

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

Multivariate linear regression.



Machine Learning

Linear Regression with multiple variables

Gradient descent for multiple variables

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: $\theta_0, \theta_1, \dots, \theta_n$

Cost function:

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

Gradient descent:

Repeat {

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

}

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

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 {

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

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

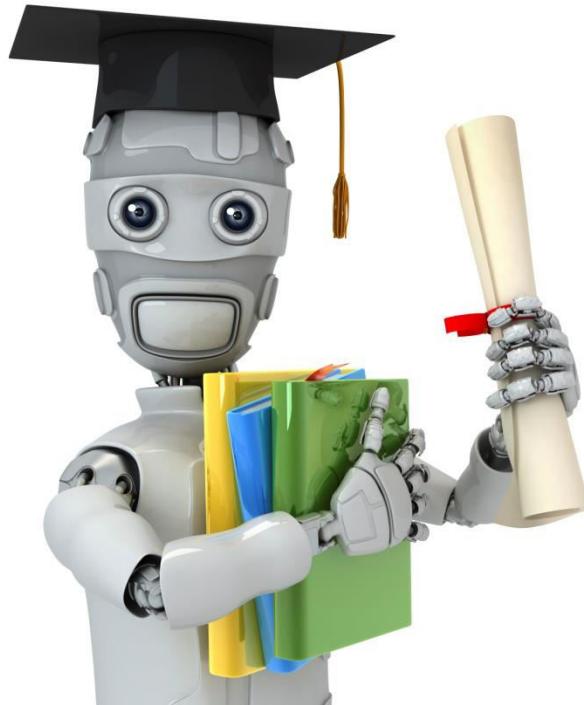
}

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

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

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

...



Machine Learning

Linear Regression with multiple variables

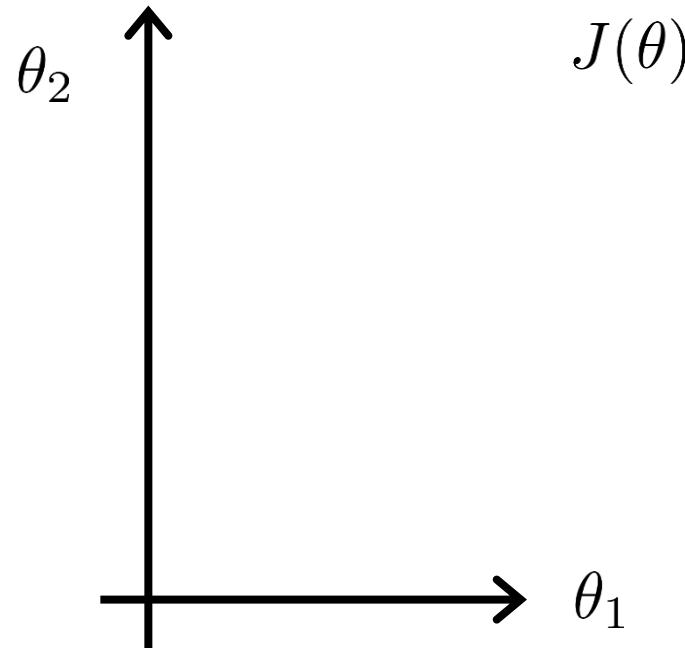
Gradient descent in practice I: Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

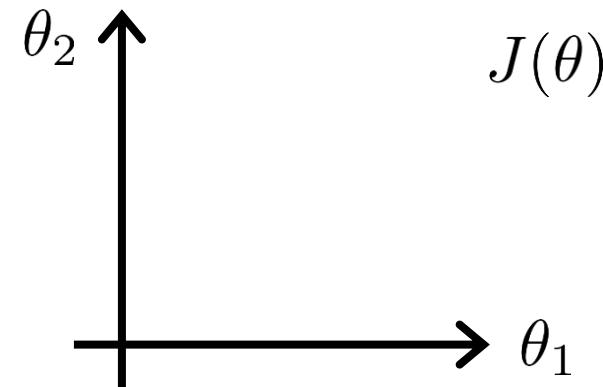
E.g. x_1 = size (0-2000 feet²)

x_2 = number of bedrooms (1-5)



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

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



Feature Scaling

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

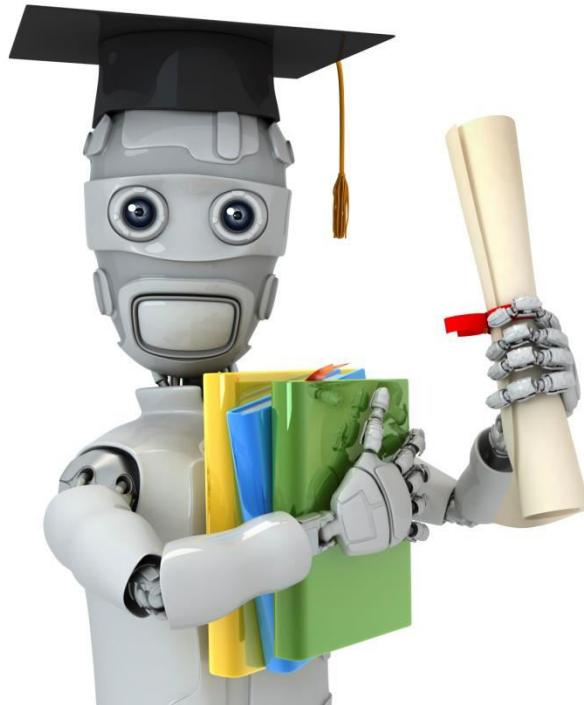
Mean normalization

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

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

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

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



Machine Learning

Linear Regression with multiple variables

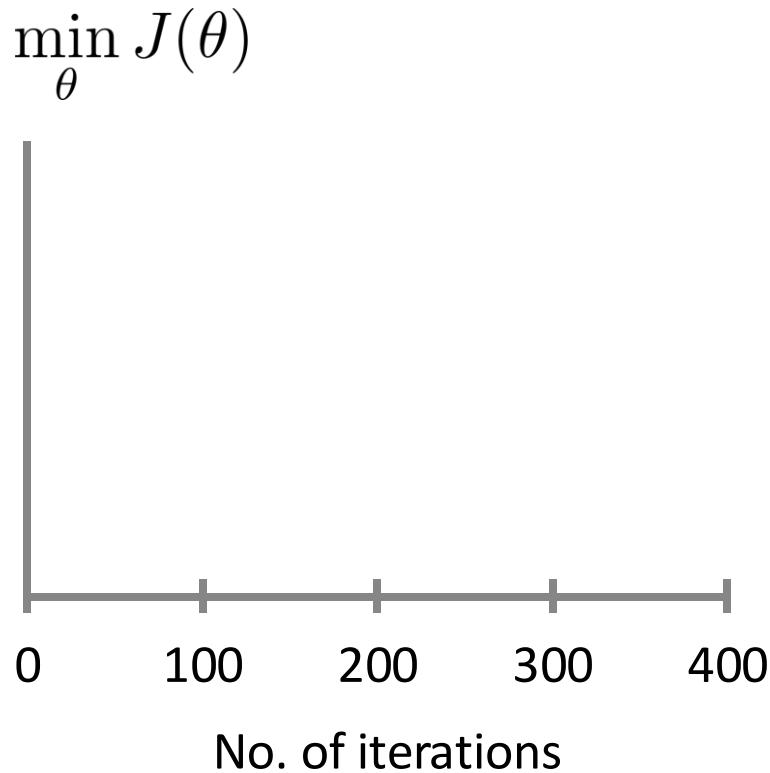
Gradient descent in practice II: Learning rate

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 .

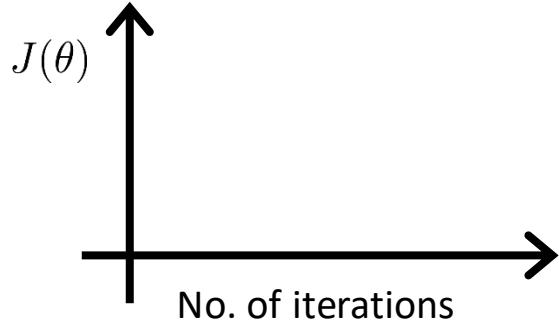
Making sure gradient descent is working correctly.



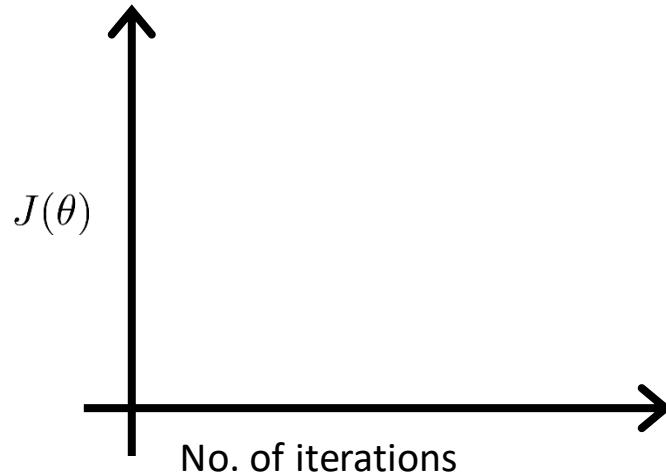
Example automatic convergence test:

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 .



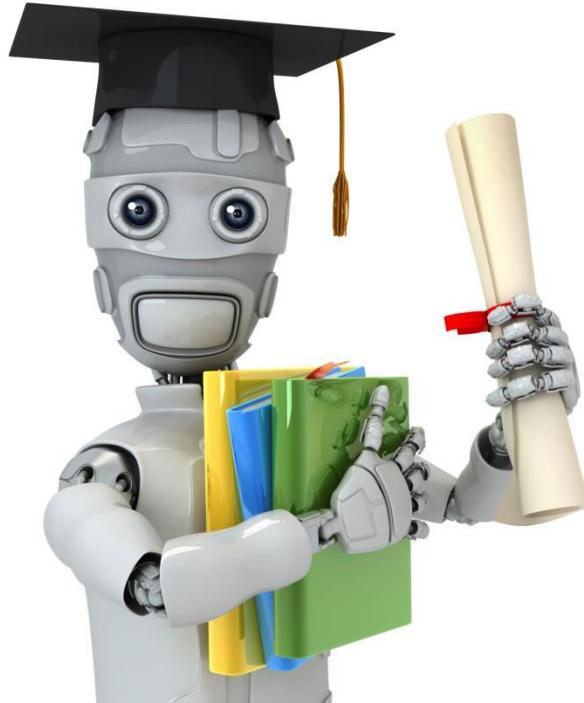
- For sufficiently small α , $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

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

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



Machine Learning

Linear Regression with multiple variables

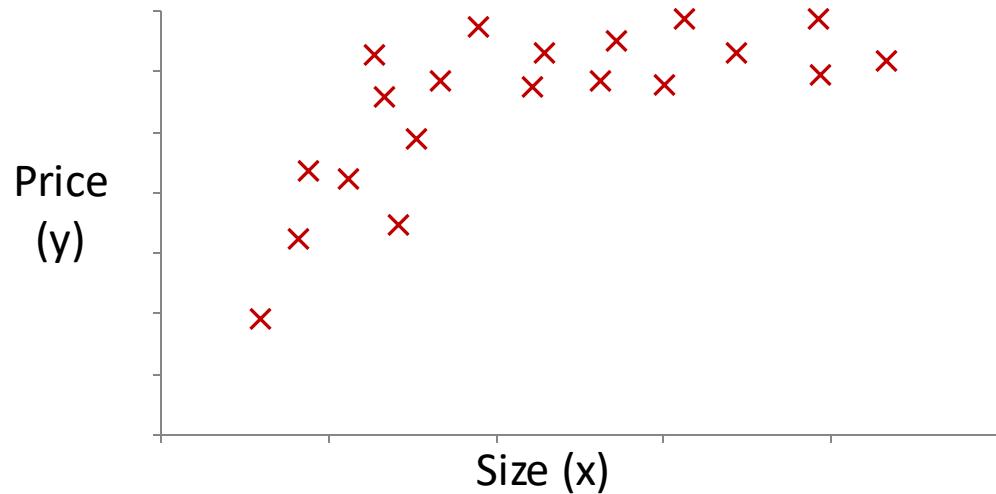
Features and
polynomial regression

Housing prices prediction

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



Polynomial regression



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

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

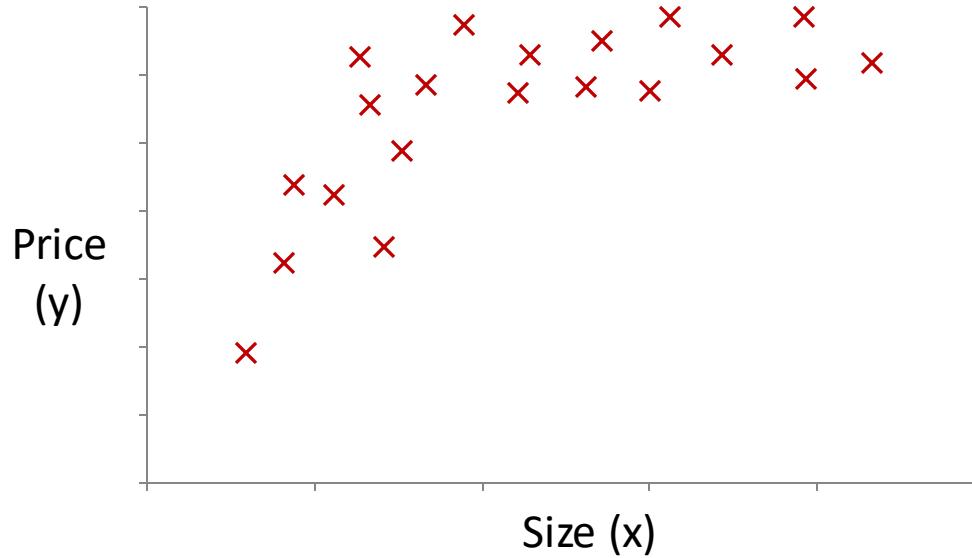
$$\begin{aligned}h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\&= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3\end{aligned}$$

$$x_1 = (\text{size})$$

$$x_2 = (\text{size})^2$$

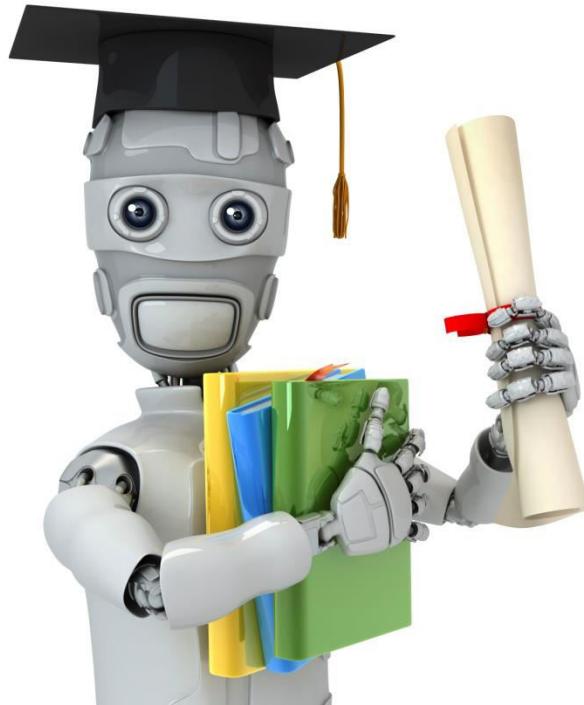
$$x_3 = (\text{size})^3$$

Choice of features



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

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

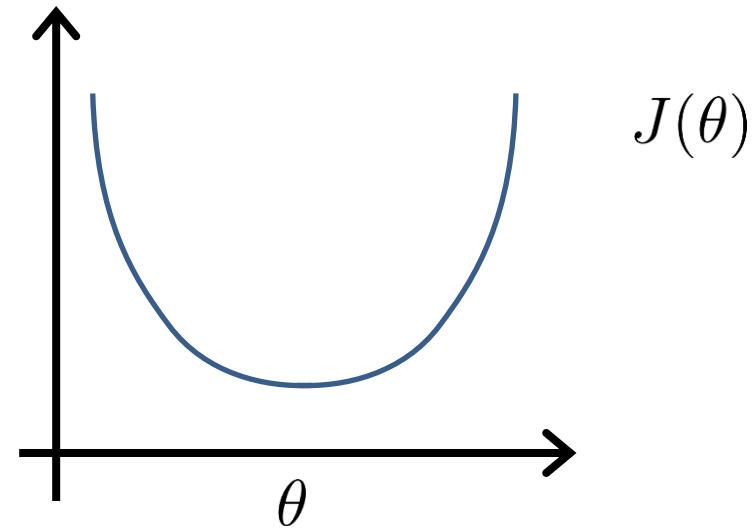


Machine Learning

Linear Regression with multiple variables

Normal equation

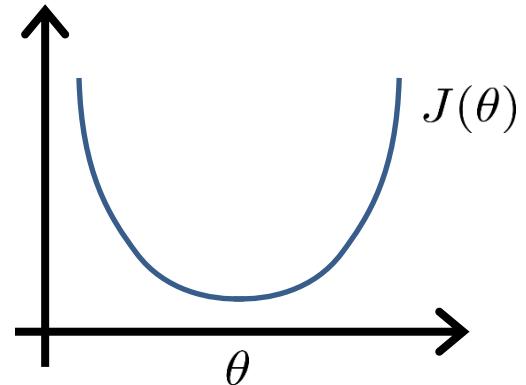
Gradient Descent



Normal equation: Method to solve for θ analytically.

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

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



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

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

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

Examples: $m = 4$.

	Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
x_0	x_1	x_2	x_3	x_4	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

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

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

m examples $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$; **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 $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

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

$(X^T X)^{-1}$ is inverse of matrix $X^T X$.

Octave: **pinv(X' *X) *X' *y**

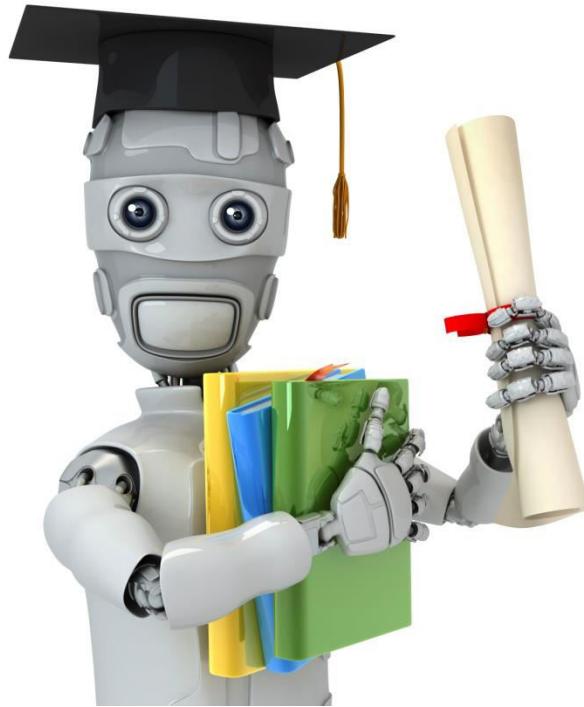
m training examples, n features.

Gradient Descent

- Need to choose α .
- Needs many iterations.
- Works well even when n is large.

Normal Equation

- No need to choose α .
- Don't need to iterate.
- Need to compute $(X^T X)^{-1}$
- Slow if n is very large.



Machine Learning

Linear Regression with multiple variables

Normal equation
and non-invertibility
(optional)

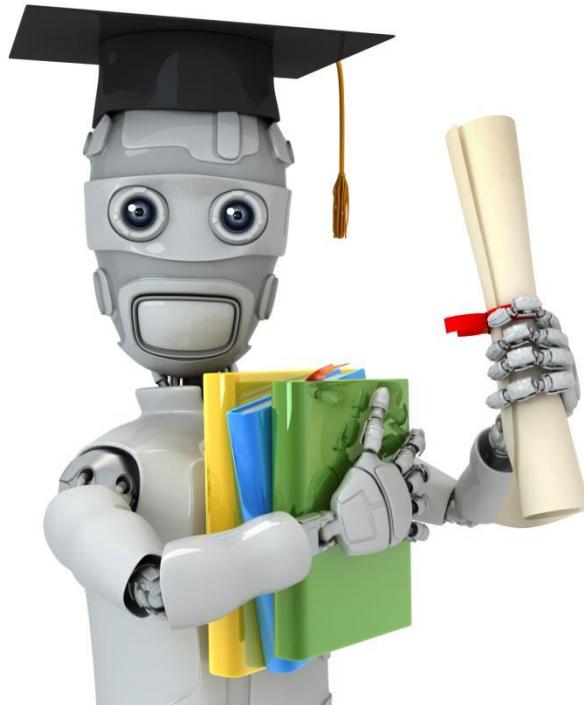
Normal equation

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

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

What if $X^T X$ is non-invertible?

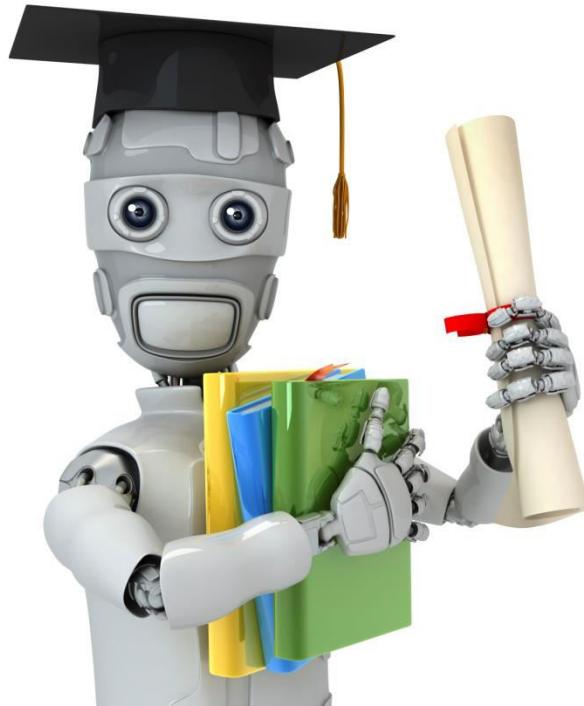
- Redundant features (linearly dependent).
E.g. $x_1 = \text{size in feet}^2$
 $x_2 = \text{size in m}^2$
- Too many features (e.g. $m \leq n$).
 - Delete some features, or use regularization.



Machine Learning

Octave Tutorial

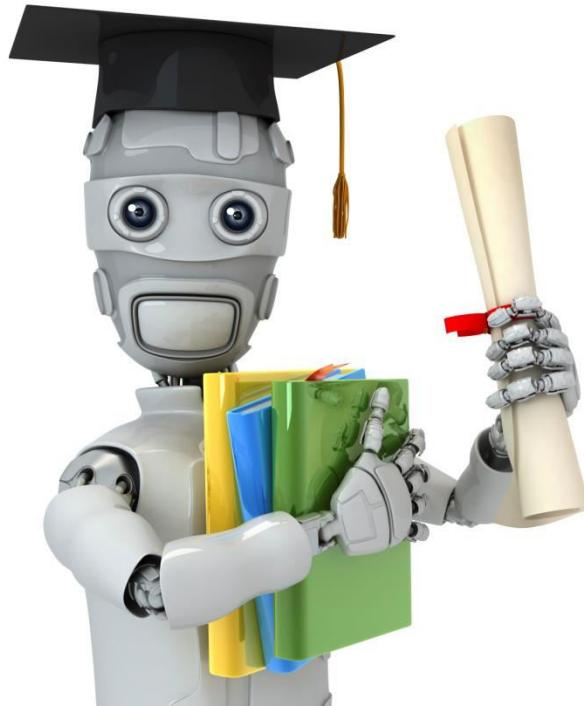
Basic operations



Machine Learning

Octave Tutorial

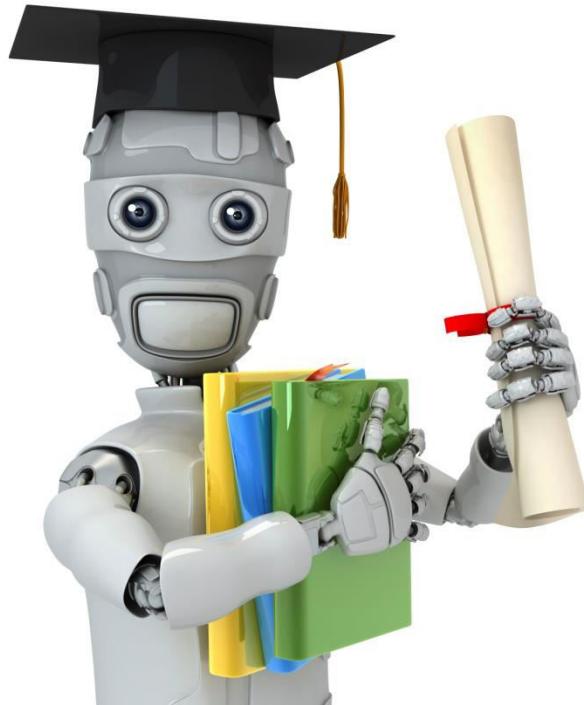
Moving data around



Machine Learning

Octave Tutorial

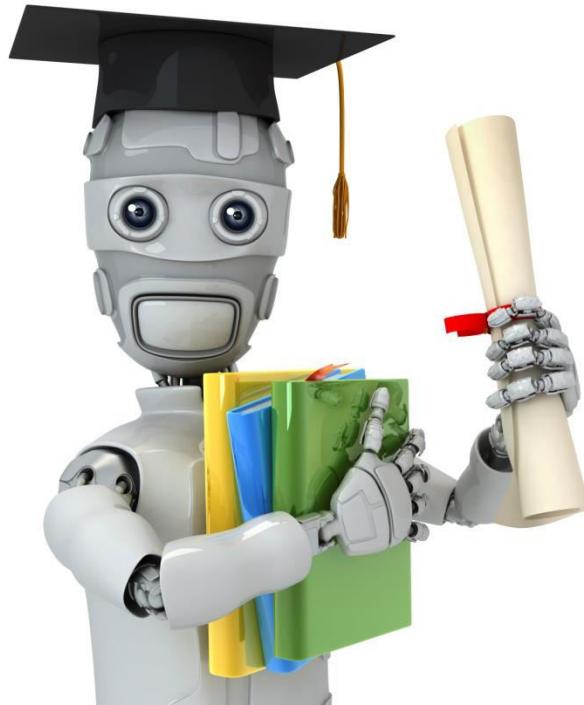
Computing on data



Machine Learning

Octave Tutorial

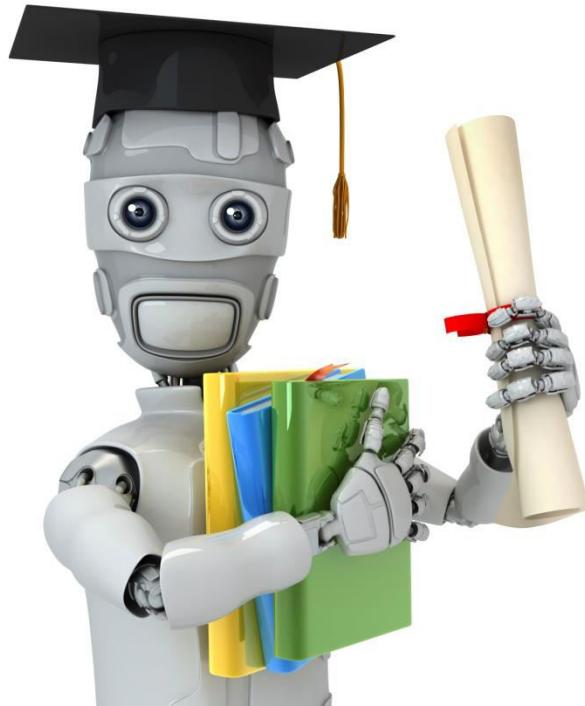
Plotting data



Machine Learning

Octave Tutorial

Control statements: for,
while, if statements



Machine Learning

Octave Tutorial

Vectorial implementation

Vectorization example.

$$h_{\theta}(x) = \sum_{j=0}^n \theta_j x_j \\ = \theta^T x$$

Unvectorized implementation

```
prediction = 0.0;  
for j = 1:n+1,  
    prediction = prediction +  
        theta(j) * x(y)  
end;
```

Vectorized implementation

```
prediction = theta' * x;
```

Vectorization example.

$$h_{\theta}(x) = \sum_{j=0}^n \theta_j x_j \\ = \theta^T x$$

Unvectorized implementation

```
double prediction = 0.0;  
for (int j = 0; j < n; j++)  
    prediction += theta[j] * x[y];
```

Vectorized implementation

```
double prediction  
= theta.transpose() * x;
```

Gradient descent

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

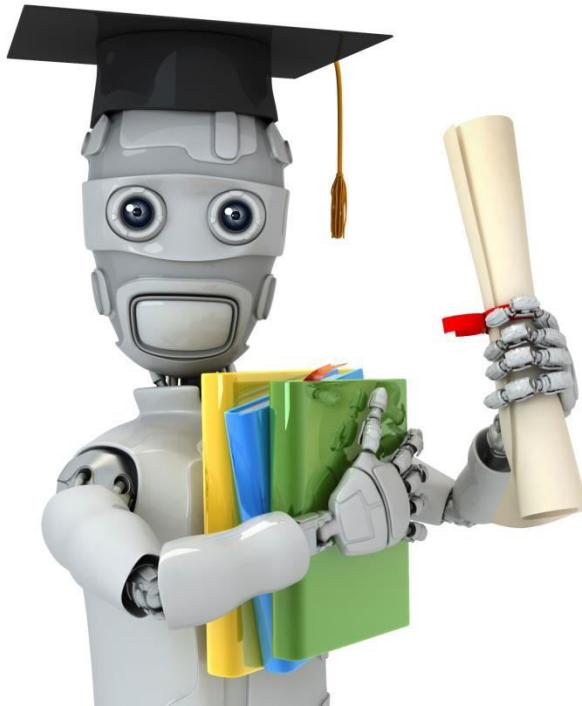
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

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

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

$$\begin{aligned}\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{\substack{i=1 \\ i=m}}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \\ \theta_2 &:= \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \\ (n = 2) &\end{aligned}$$

$$\left| \begin{array}{l} u(j) = 2v(j) + 5w(j) \quad (\text{for all } j) \\ u = 2v + 5w \end{array} \right.$$



Logistic Regression

Classification

Machine Learning

Classification

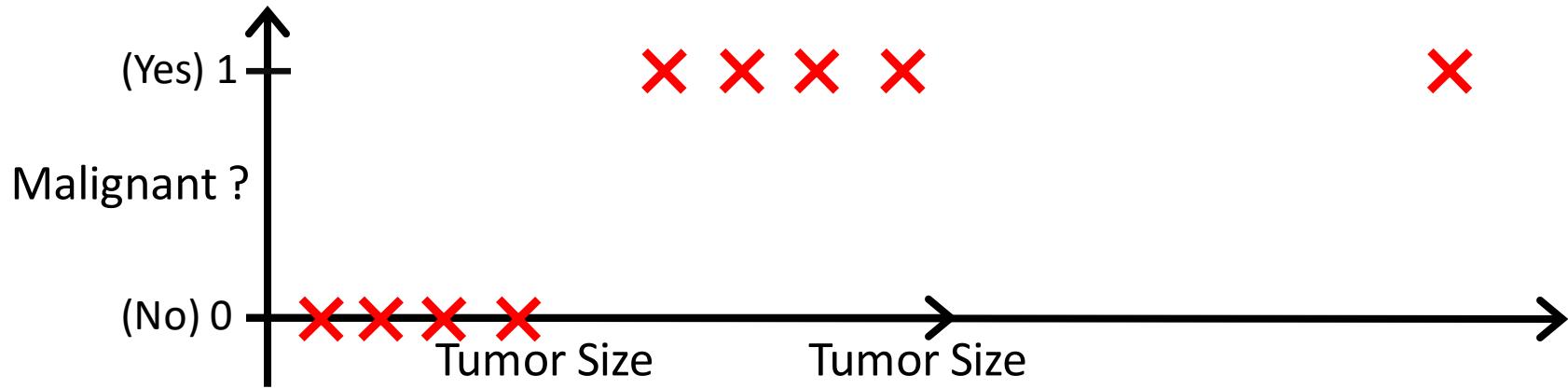
Email: Spam / Not Spam?

Online Transactions: Fraudulent (Yes / No)?

Tumor: Malignant / Benign ?

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

- 0: “Negative Class” (e.g., benign tumor)
- 1: “Positive Class” (e.g., malignant tumor)



Threshold classifier output $h_{\theta}(x)$ at 0.5:

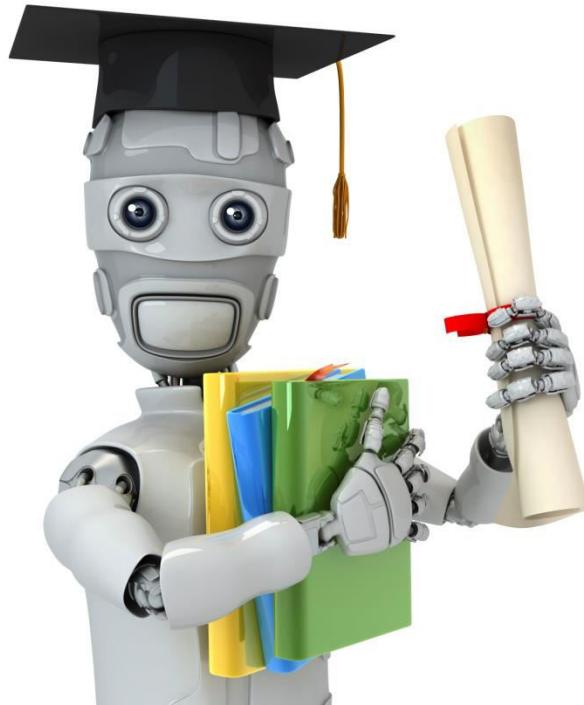
If $h_{\theta}(x) \geq 0.5$, predict “y = 1”

If $h_{\theta}(x) < 0.5$, predict “y = 0”

Classification: $y = 0$ or 1

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

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



Machine Learning

Logistic Regression

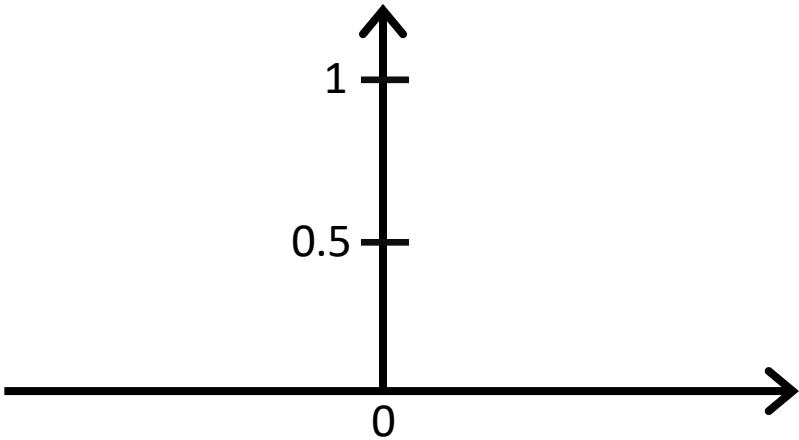
Hypothesis Representation

Logistic Regression Model

Want $0 \leq h_\theta(x) \leq 1$

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

Sigmoid function
Logistic function



Interpretation of Hypothesis Output

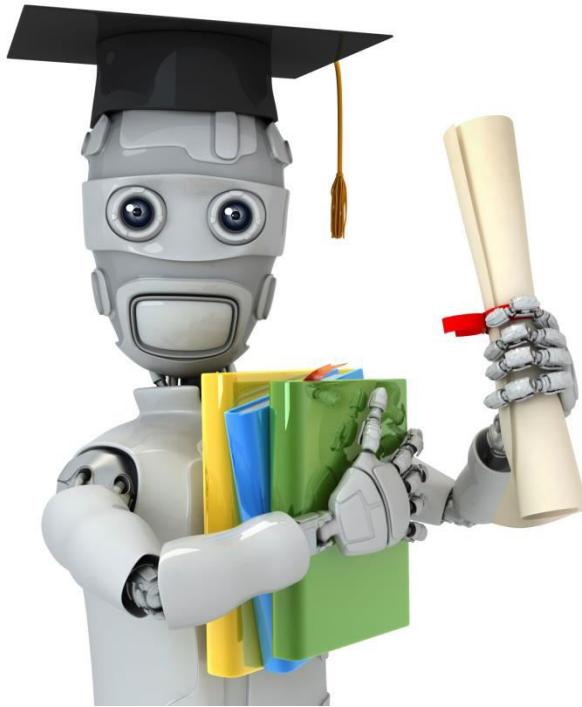
$h_{\theta}(x)$ = estimated probability that $y = 1$ on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$
 $h_{\theta}(x) = 0.7$

Tell patient that 70% chance of tumor being malignant

“probability that $y = 1$, given x ,
parameterized by θ ”

$$P(y = 0|x; \theta) + P(y = 1|x; \theta) = 1$$
$$P(y = 0|x; \theta) = 1 - P(y = 1|x; \theta)$$



Machine Learning

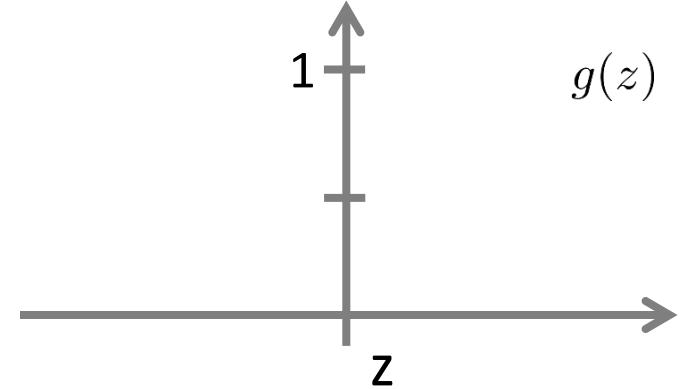
Logistic Regression

Decision boundary

Logistic regression

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

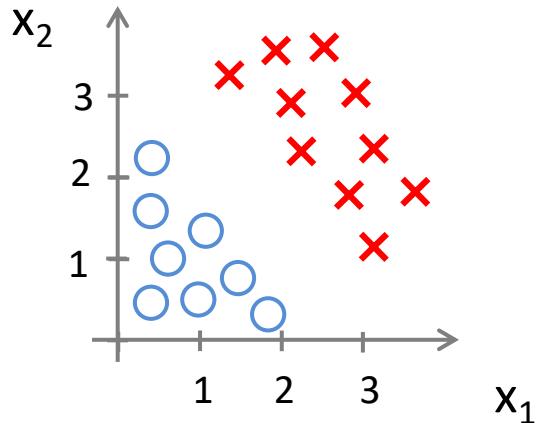
$$g(z) = \frac{1}{1+e^{-z}}$$



Suppose predict “ $y = 1$ “ if $h_{\theta}(x) \geq 0.5$

predict “ $y = 0$ “ if $h_{\theta}(x) < 0.5$

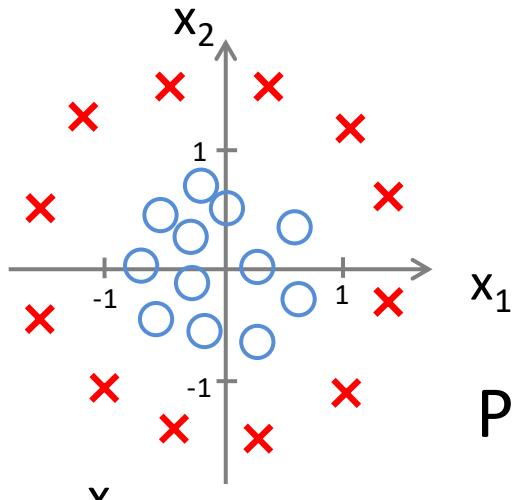
Decision Boundary



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

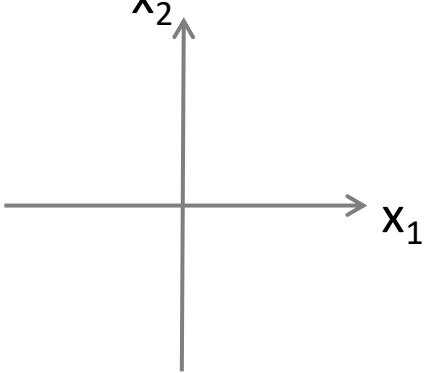
Predict “ $y = 1$ “ if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundaries

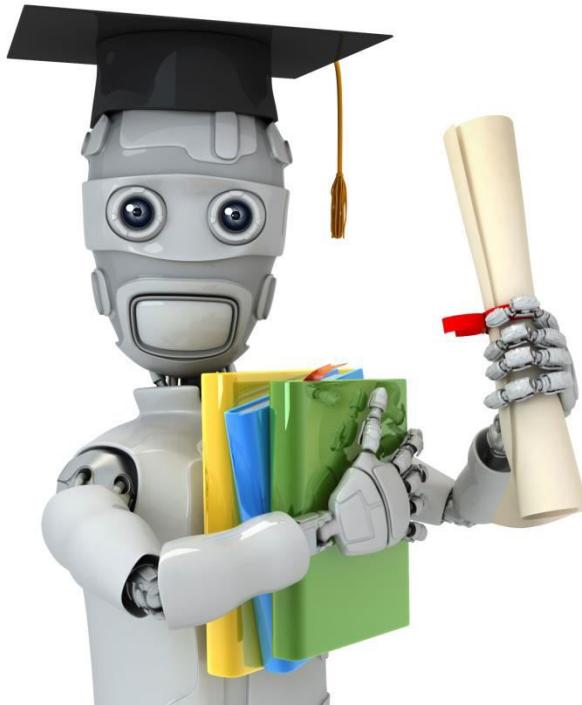


$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

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



$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$



Machine Learning

Logistic Regression

Cost function

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

m examples

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

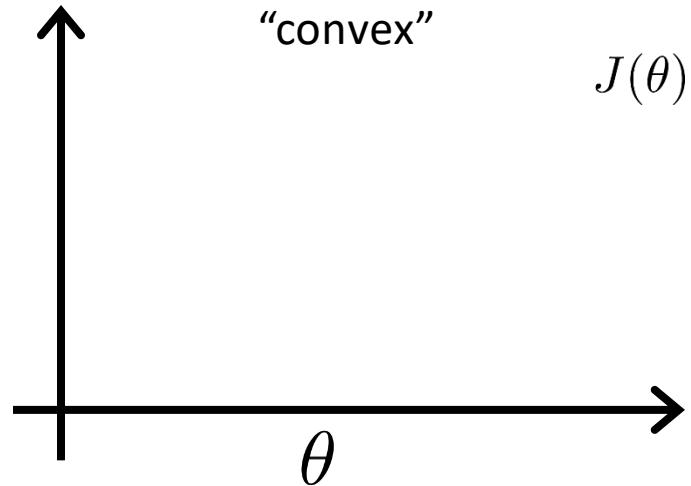
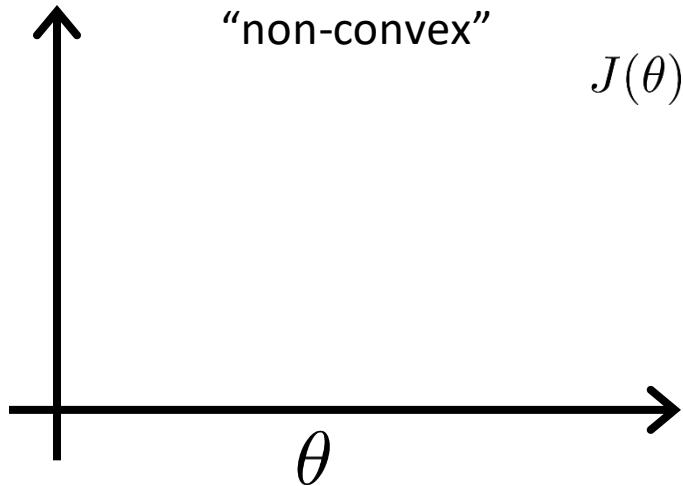
$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters θ ?

Cost function

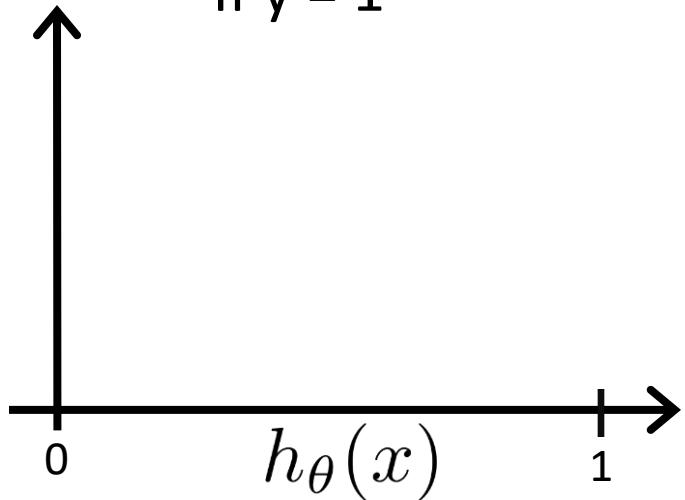
Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$



Logistic regression cost function

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



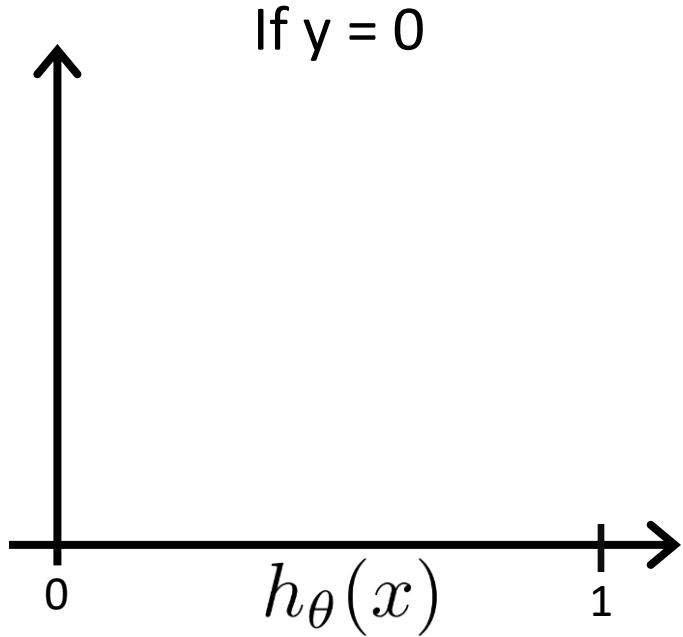
If $y = 1$

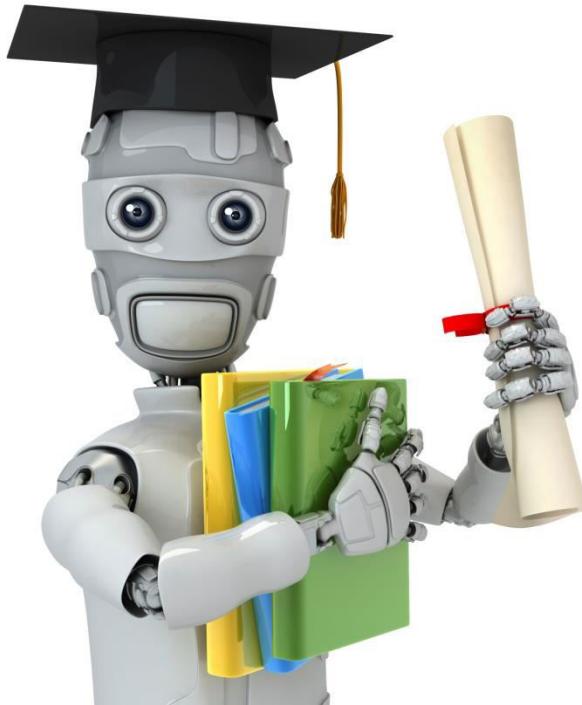
Cost = 0 if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \rightarrow 0$
 $Cost \rightarrow \infty$

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.

Logistic regression cost function

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





Machine Learning

Logistic Regression

Simplified cost function
and gradient descent

Logistic regression cost function

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

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

Note: $y = 0$ or 1 always

Logistic regression cost function

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

To fit parameters θ :

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

To make a prediction given new x :

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

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 {

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

}

(simultaneously update all θ_j)

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

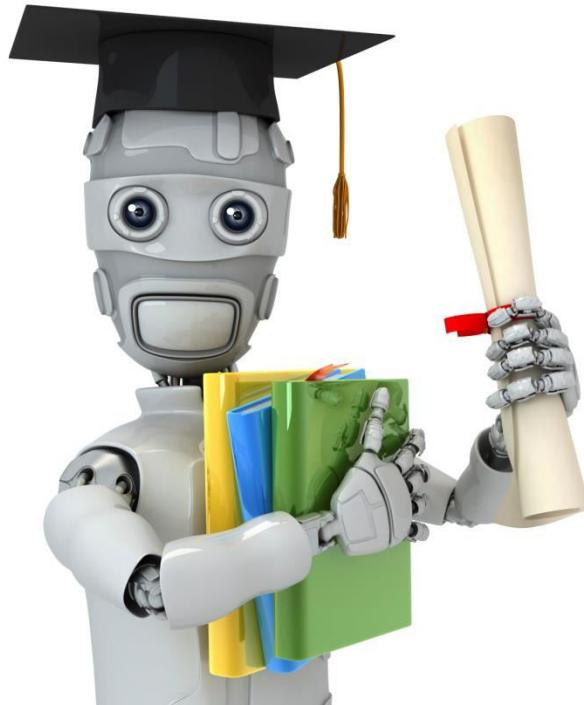
Repeat {

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

}

(simultaneously update all θ_j)

Algorithm looks identical to linear regression!



Machine Learning

Logistic Regression

Advanced optimization

Optimization algorithm

Cost function $J(\theta)$. Want $\min_{\theta} J(\theta)$.

Given θ , we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \dots, n$)

Gradient descent:

Repeat {

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

}

Optimization algorithm

Given θ , we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \dots, n$)

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:

- More complex

Example:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
= fminunc(@costFunction, initialTheta, options);
```

```
function [jVal, gradient]
    = costFunction(theta)
jVal = (theta(1)-5)^2 + ...
       (theta(2)-5)^2;
gradient = zeros(2,1);
gradient(1) = 2*(theta(1)-5);
gradient(2) = 2*(theta(2)-5);
```

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

```
function [jVal, gradient] = costFunction(theta)
```

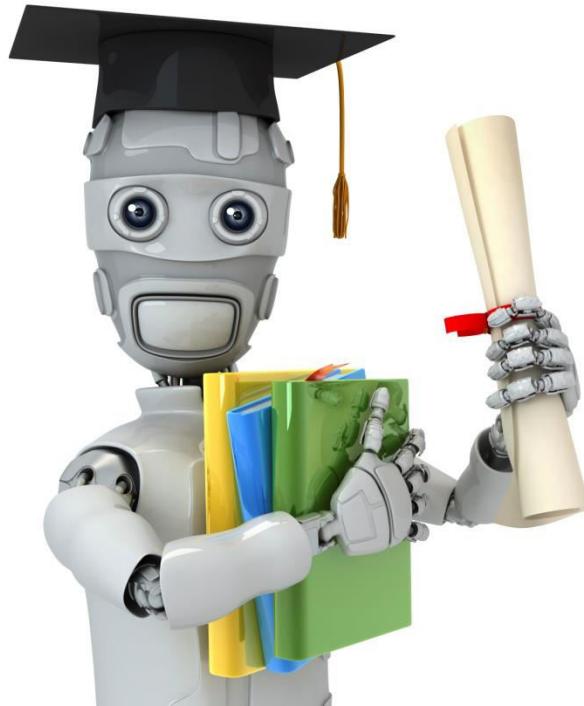
```
jVal = [ code to compute  $J(\theta)$ ] ;
```

```
gradient(1) = [ code to compute  $\frac{\partial}{\partial \theta_0} J(\theta)$ ] ;
```

```
gradient(2) = [ code to compute  $\frac{\partial}{\partial \theta_1} J(\theta)$ ] ;
```

```
⋮
```

```
gradient(n+1) = [ code to compute  $\frac{\partial}{\partial \theta_n} J(\theta)$ ] ;
```



Machine Learning

Logistic Regression

Multi-class classification:
One-vs-all

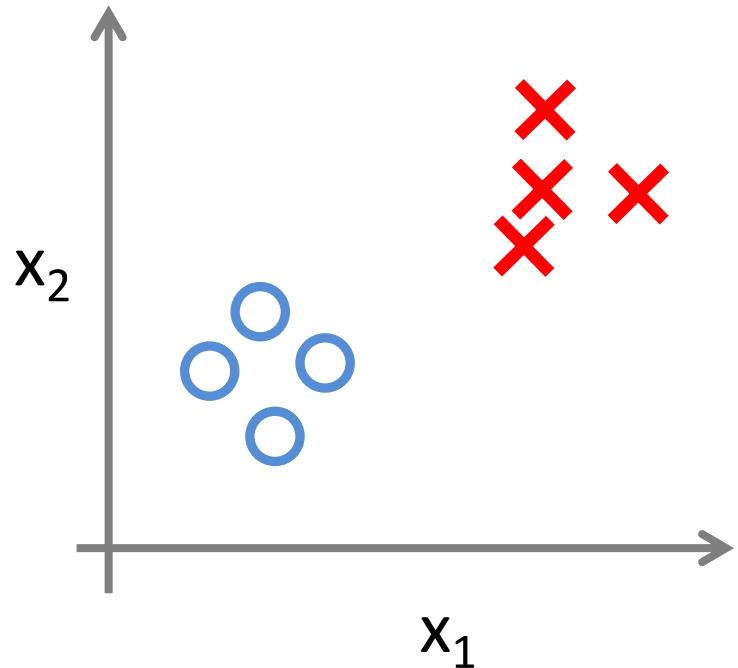
Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

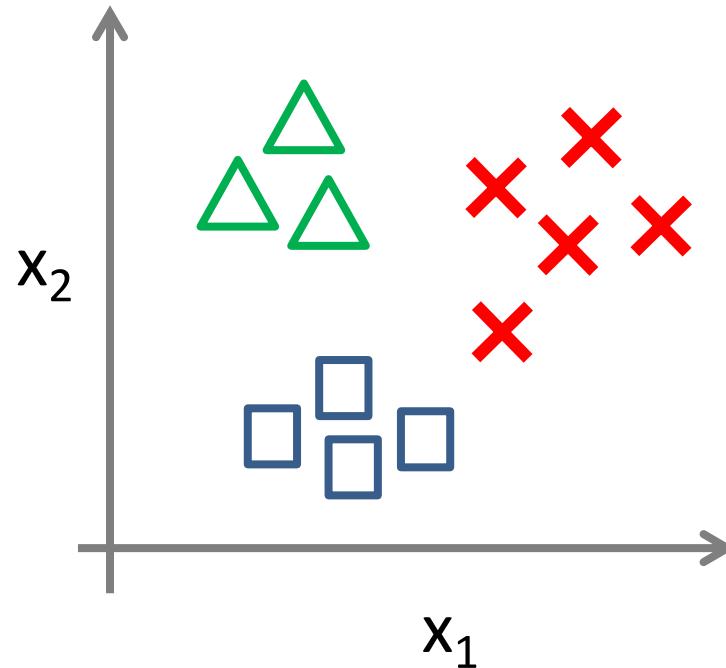
Medical diagrams: Not ill, Cold, Flu

Weather: Sunny, Cloudy, Rain, Snow

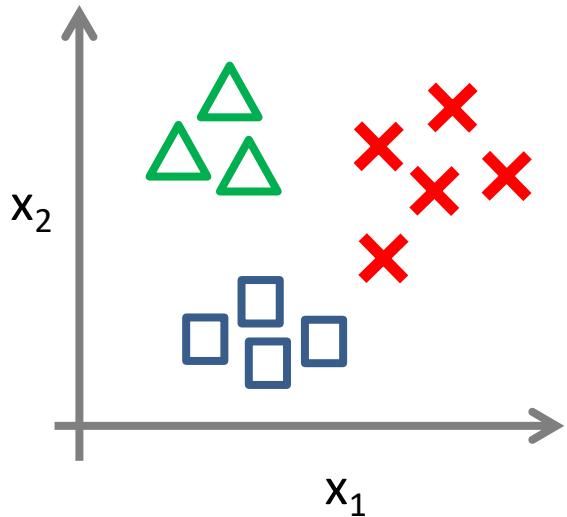
Binary classification:



Multi-class classification:



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

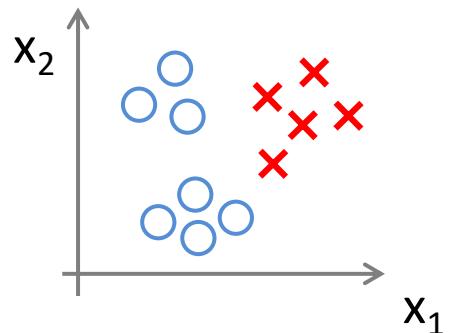
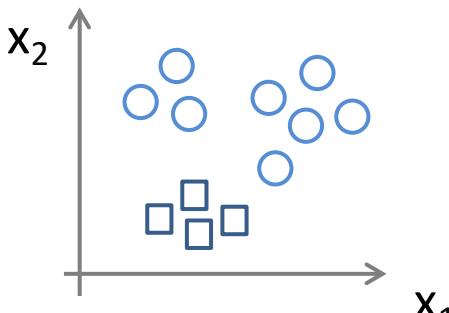
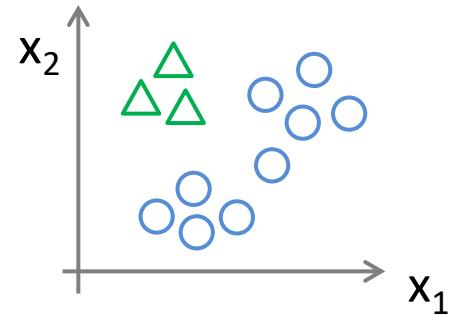
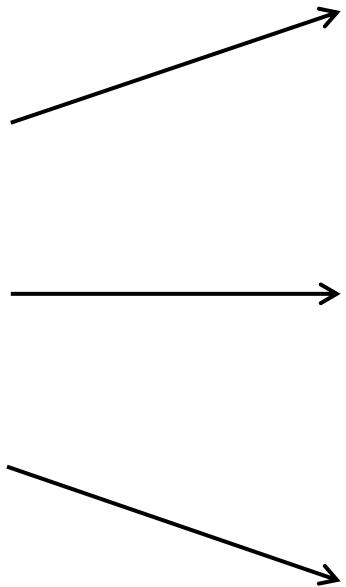


Class 1:

Class 2:

Class 3:

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

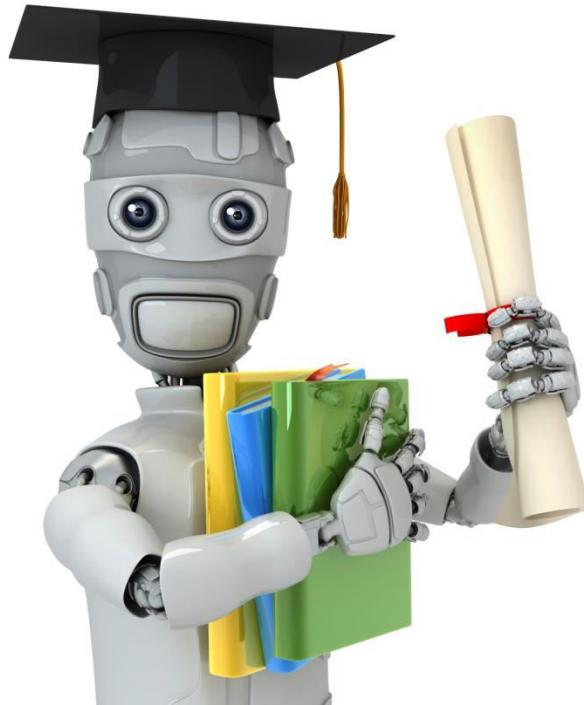


One-vs-all

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

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

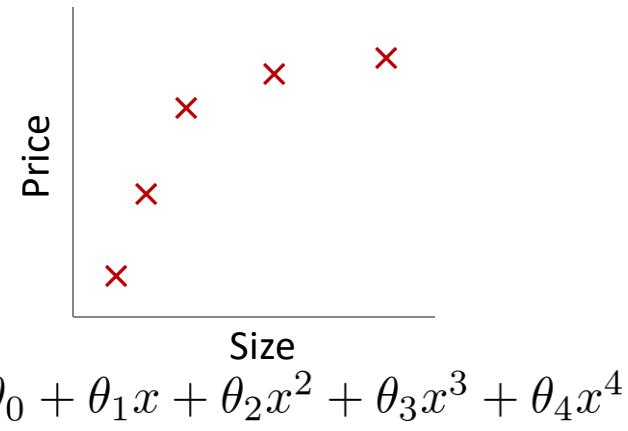
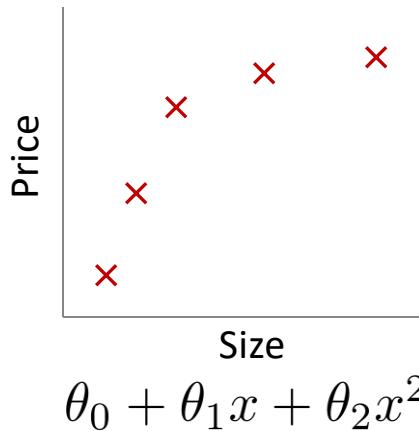
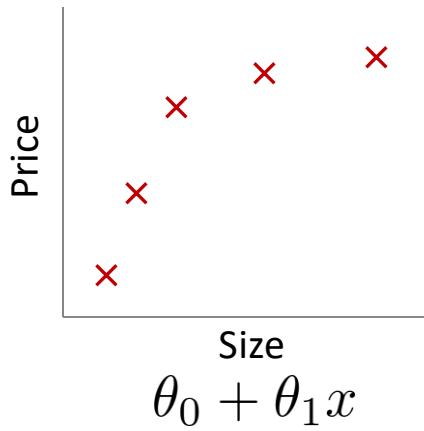


Machine Learning

Regularization

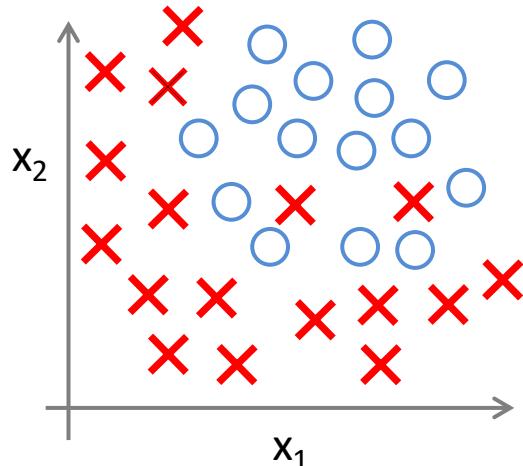
The problem of overfitting

Example: Linear regression (housing prices)



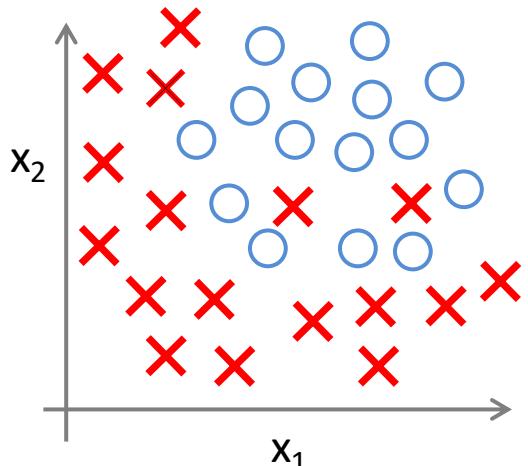
Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

Example: Logistic regression

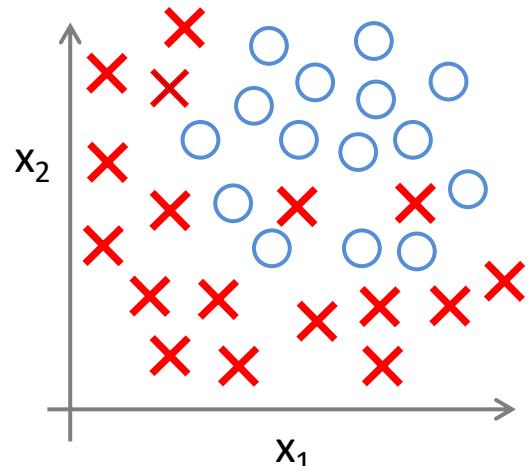


$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

(g = sigmoid function)



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



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

Addressing overfitting:

x_1 = size of house

x_2 = no. of bedrooms

x_3 = no. of floors

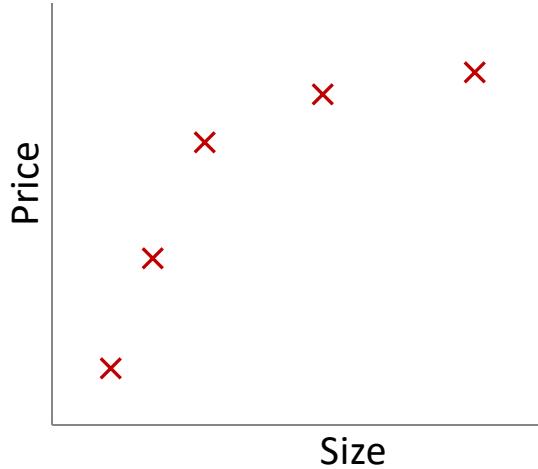
x_4 = age of house

x_5 = average income in neighborhood

x_6 = kitchen size

⋮

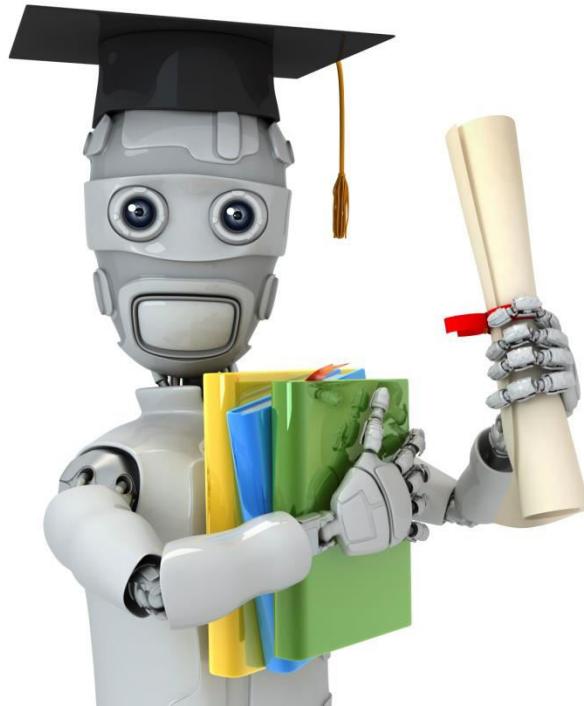
x_{100}



Addressing overfitting:

Options:

1. Reduce number of features.
 - Manually select which features to keep.
 - Model selection algorithm (later in course).
2. Regularization.
 - Keep all the features, but reduce magnitude/values of parameters θ_j
 - Works well when we have a lot of features, each of which contributes a bit to predicting y .

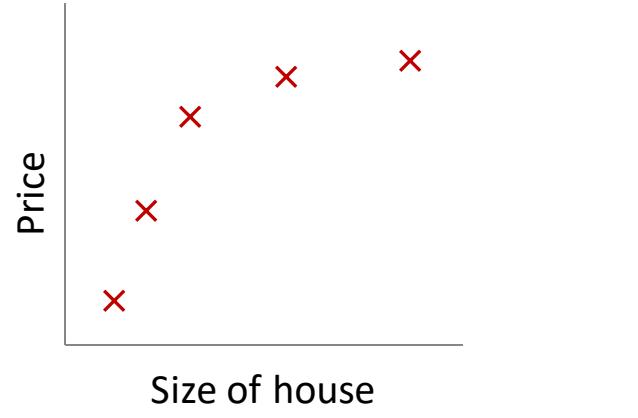
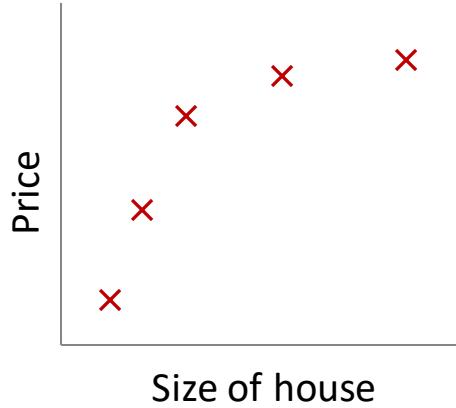


Machine Learning

Regularization

Cost function

Intuition



Suppose we penalize and make θ_3, θ_4 really small.

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

Regularization.

Small values for parameters $\theta_0, \theta_1, \dots, \theta_n$

- “Simpler” hypothesis
- Less prone to overfitting

Housing:

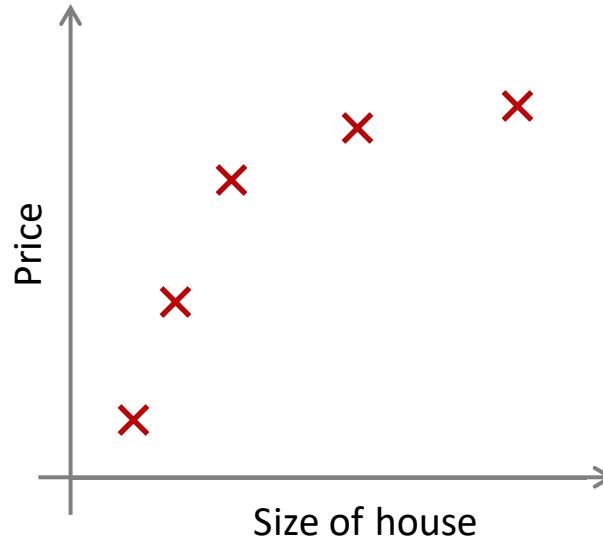
- Features: x_1, x_2, \dots, x_{100}
- Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

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

Regularization.

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

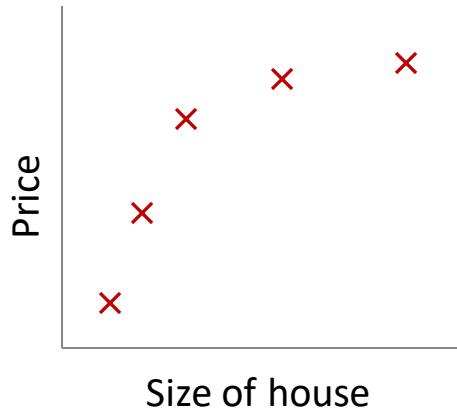
$$\min_{\theta} J(\theta)$$



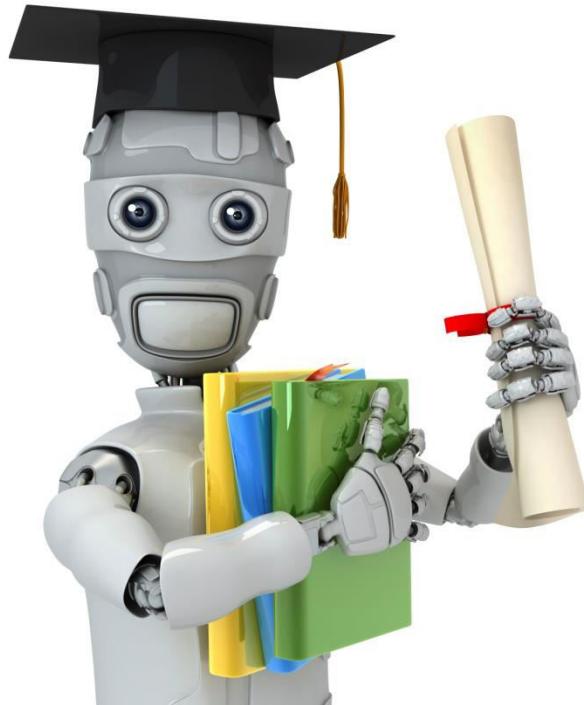
In regularized linear regression, we choose θ to minimize

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$



Machine Learning

Regularization

Regularized linear regression

Regularized linear regression

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

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

Gradient descent

Repeat {

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

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

$(j = \cancel{X}, 1, 2, 3, \dots, n)$

}

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

Normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

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

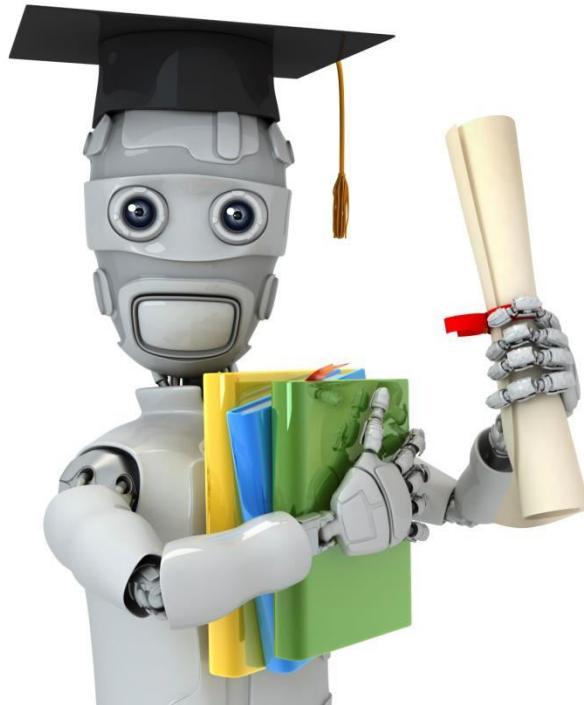
Non-invertibility (optional/advanced).

Suppose $m \leq n$,
(#examples) (#features)

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

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

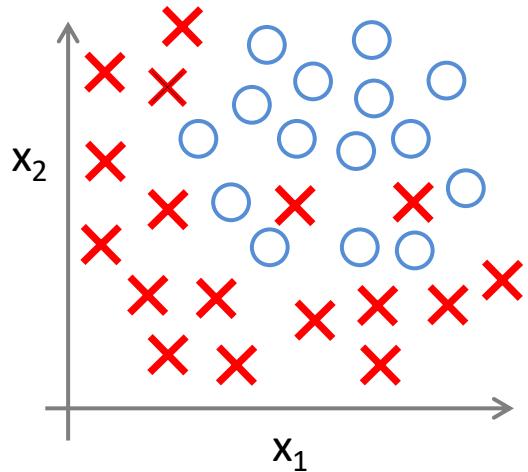


Machine Learning

Regularization

Regularized logistic regression

Regularized logistic regression.



$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

Cost function:

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

Gradient descent

Repeat {

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

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad (j = \textcolor{red}{X}, 1, 2, 3, \dots, n)$$

}

Advanced optimization

```
function [jVal, gradient] = costFunction(theta)
```

jVal = [code to compute $J(\theta)$] ;

$$J(\theta) = \left[-\frac{1}{m} \sum_{i=1}^m y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log 1 - h_\theta(x^{(i)}) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

gradient(1) = [code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$] ;

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

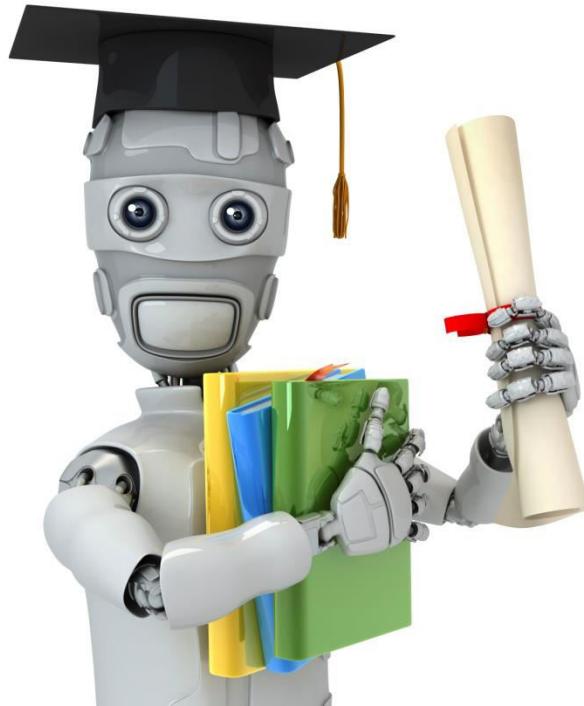
gradient(2) = [code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$] ;

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

gradient(3) = [code to compute $\frac{\partial}{\partial \theta_2} J(\theta)$] ;

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

gradient(n+1) = [code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$] ;

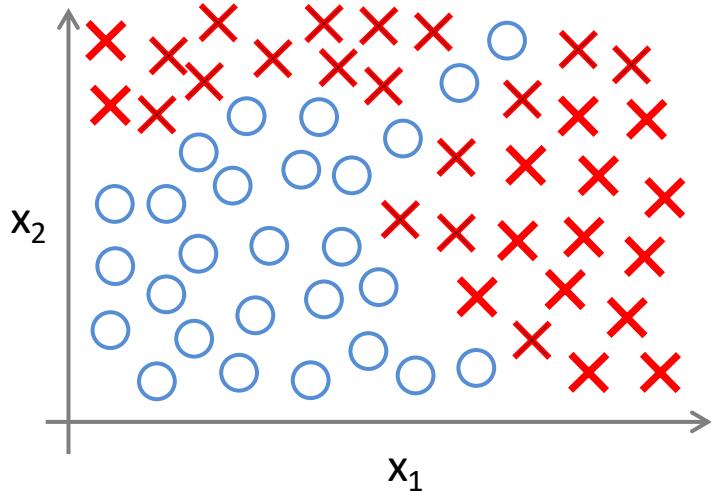


Machine Learning

Neural Networks: Representation

Non-linear hypotheses

Non-linear Classification



x_1 = size

x_2 = # bedrooms

x_3 = # floors

x_4 = age

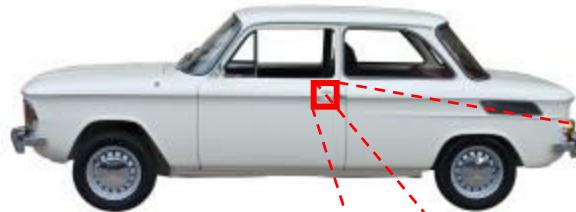
...

x_{100}

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

What is this?

You see this:



But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Computer Vision: Car detection



Cars

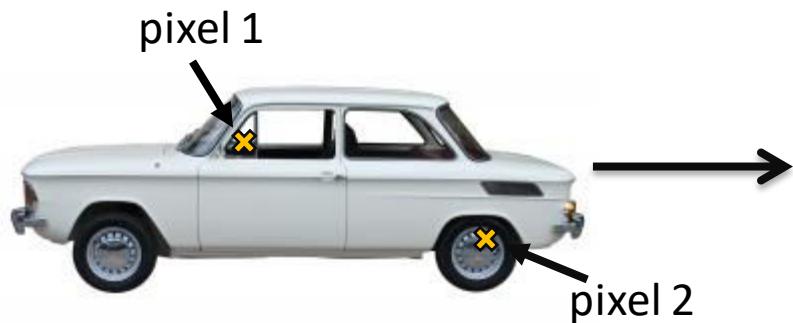


Not a car

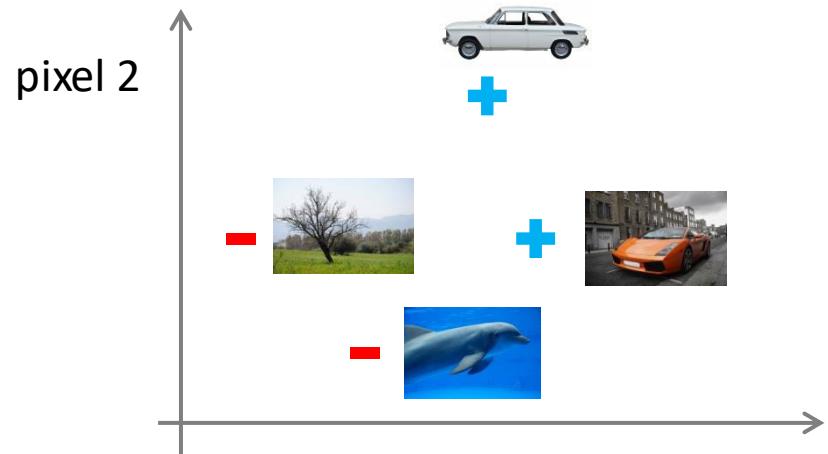
Testing:



What is this?



Learning
Algorithm



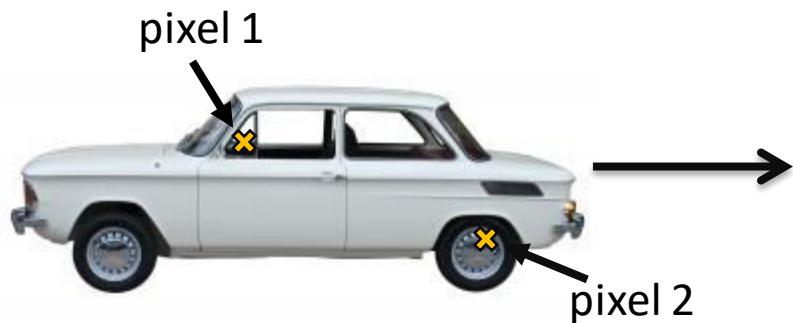
+

Cars

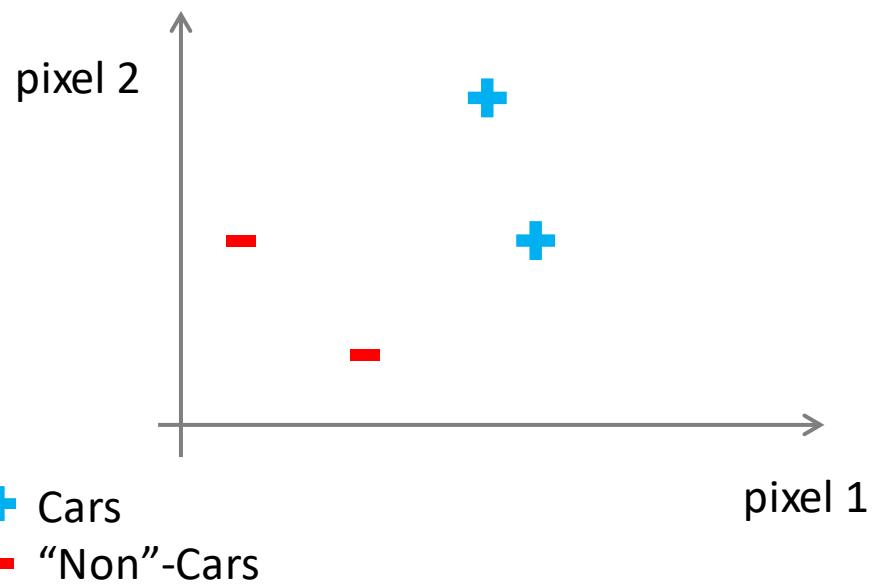
-

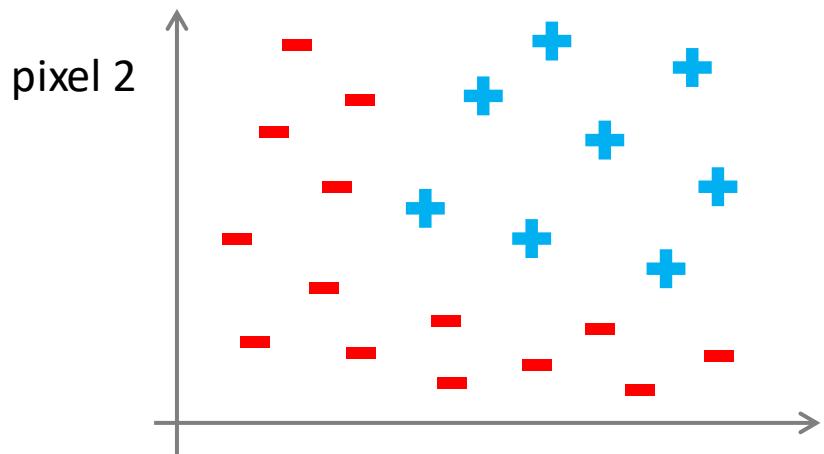
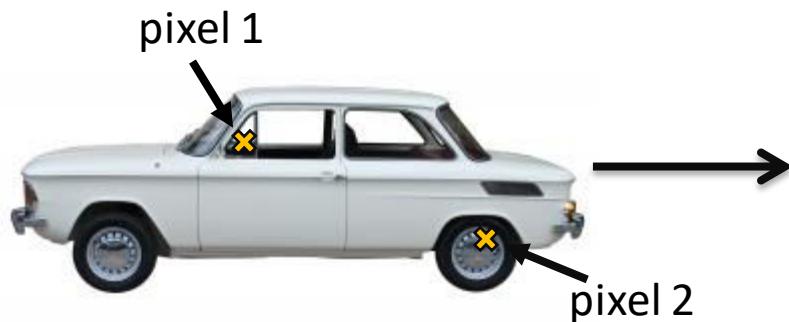
"Non"-Cars

pixel 1



Learning
Algorithm





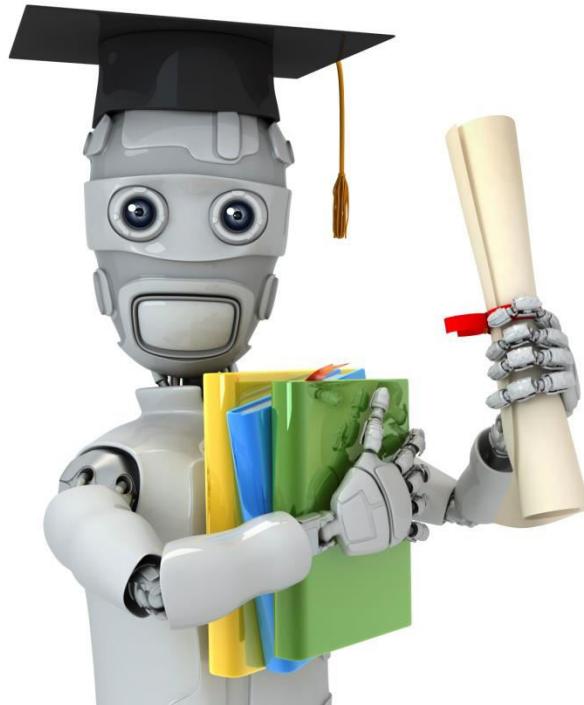
50×50 pixel images \rightarrow 2500 pixels
 $n = 2500$ (7500 if RGB)

$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

0-255

+ Cars
- "Non"-Cars

Quadratic features ($x_i \times x_j$): 3 million features



Machine Learning

Neural Networks: Representation

Neurons and the brain

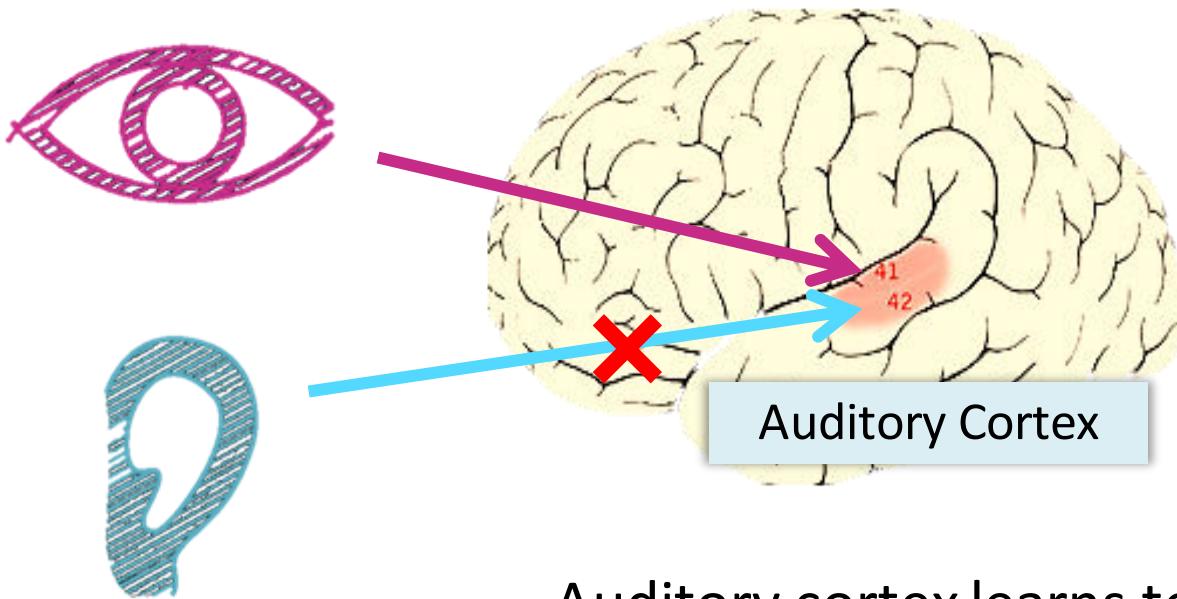
Neural Networks

Origins: Algorithms that try to mimic the brain.

Was very widely used in 80s and early 90s; popularity diminished in late 90s.

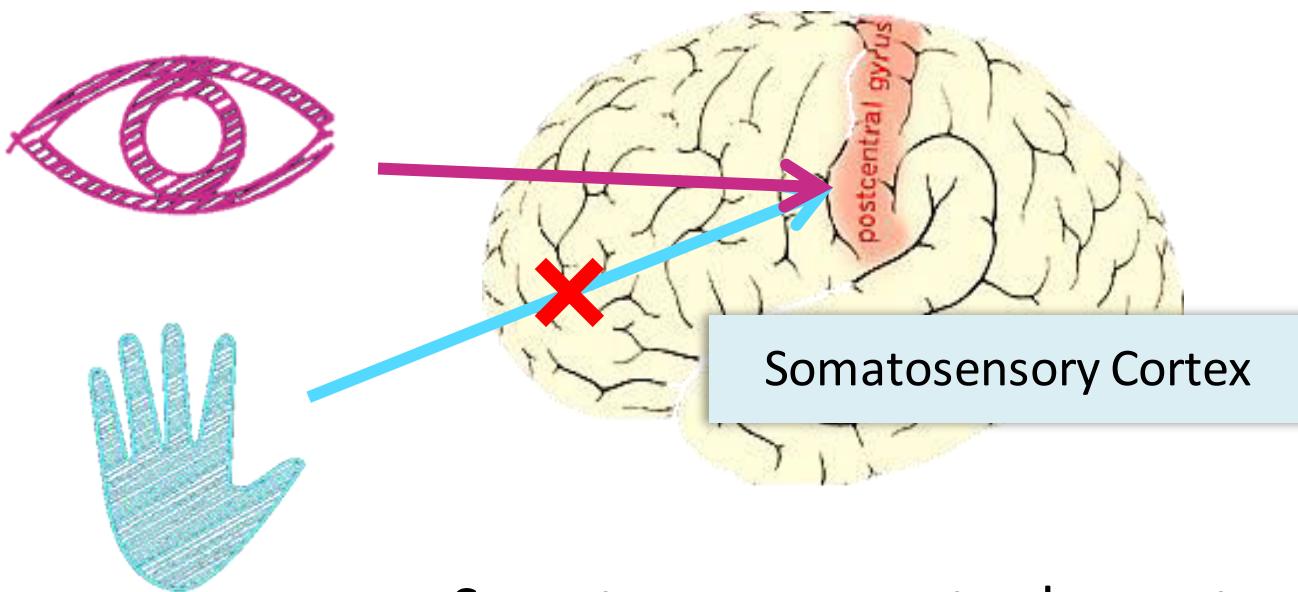
Recent resurgence: State-of-the-art technique for many applications

The “one learning algorithm” hypothesis



Auditory cortex learns to see

The “one learning algorithm” hypothesis



Somatosensory cortex learns to see

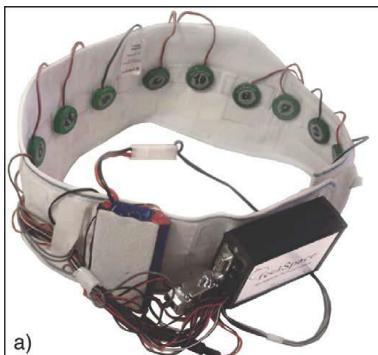
Sensor representations in the brain



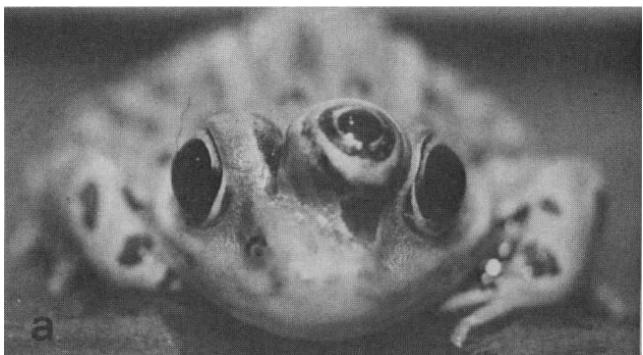
Seeing with your tongue



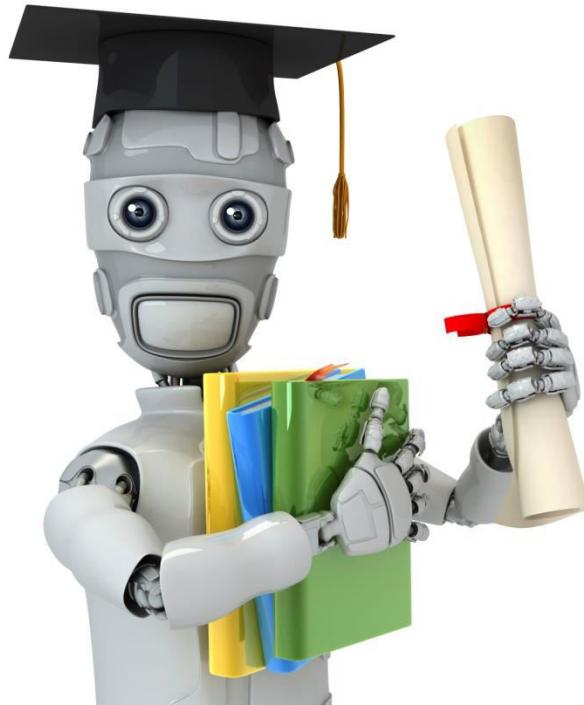
Human echolocation (sonar)



Haptic belt: Direction sense



Implanting a 3rd eye

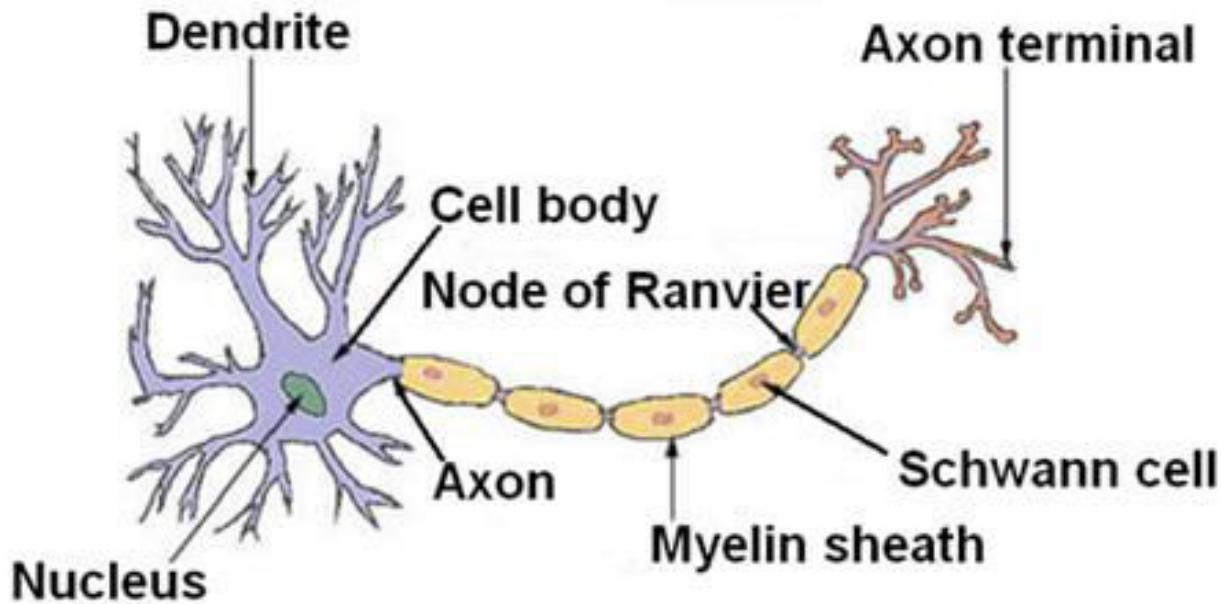


Machine Learning

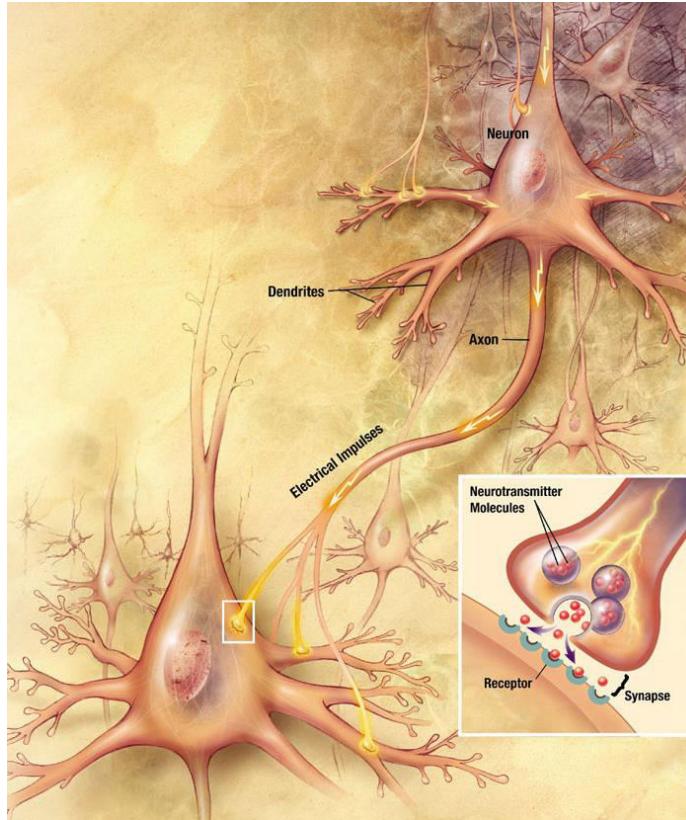
Neural Networks: Representation

Model representation I

Neuron in the brain



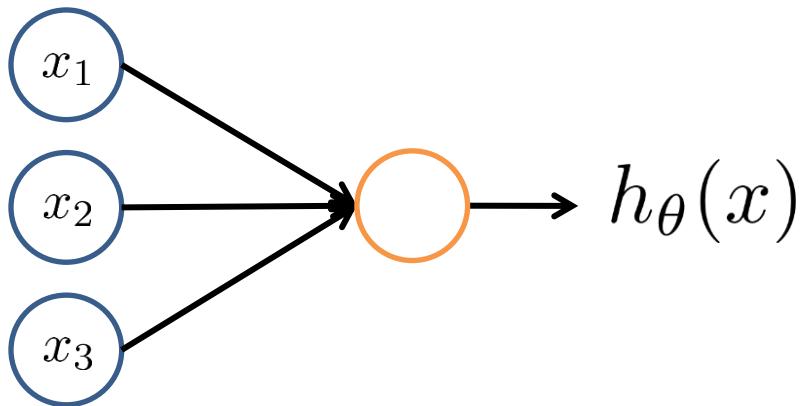
Neurons in the brain



[Credit: US National Institutes of Health, National Institute on Aging]

Andrew Ng

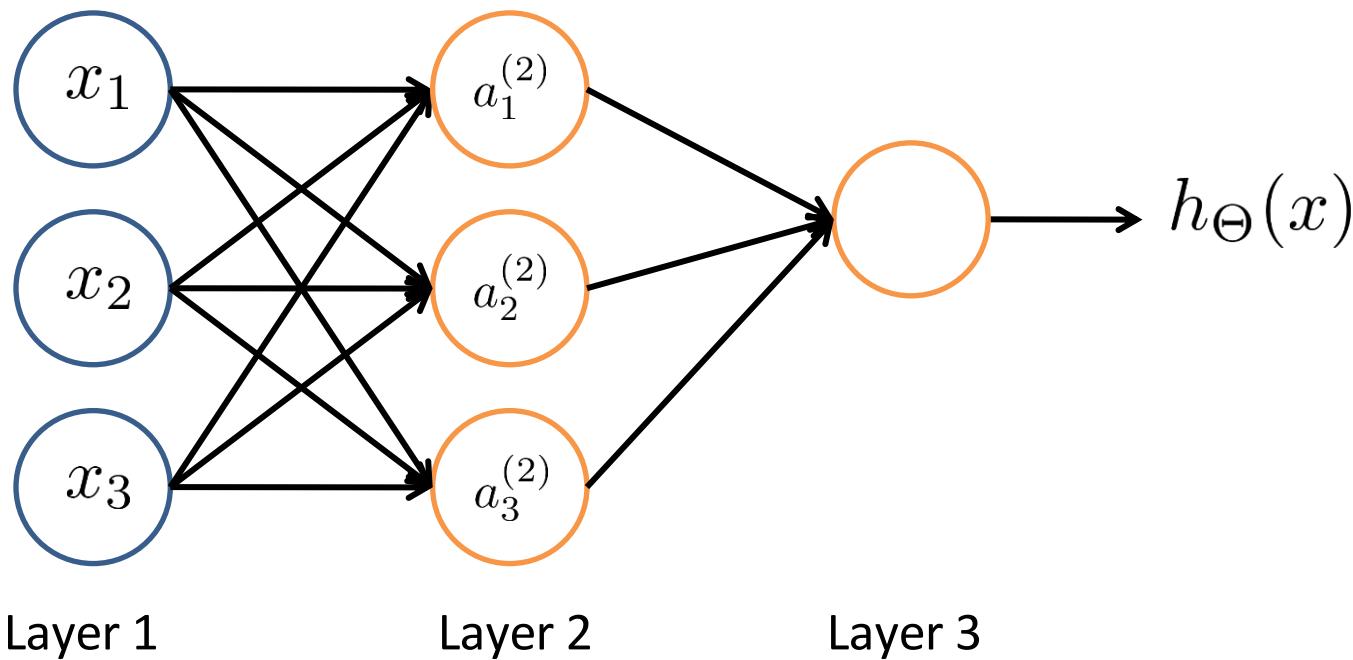
Neuron model: Logistic unit



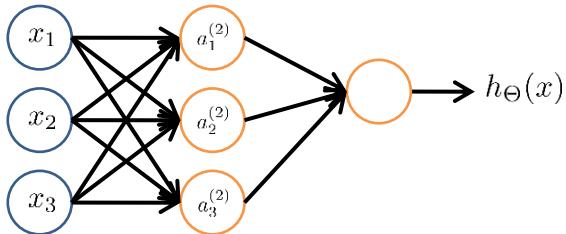
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

Sigmoid (logistic) activation function.

Neural Network



Neural Network



$a_i^{(j)}$ = “activation” of unit i in layer j

$\Theta^{(j)}$ = matrix of weights controlling
function mapping from layer j to
layer $j + 1$

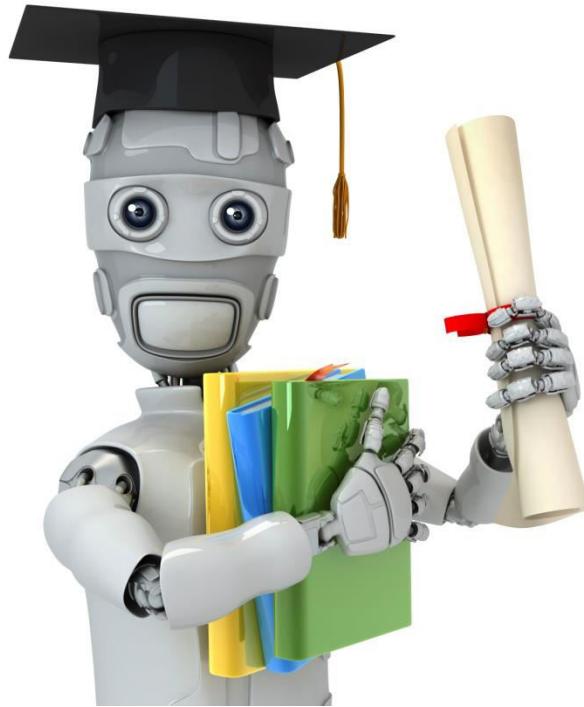
$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

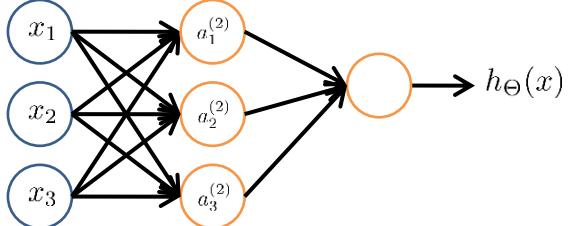
If network has s_j units in layer j , s_{j+1} units in layer $j + 1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.



Machine Learning

Neural Networks: Representation Model representation II

Forward propagation: Vectorized implementation



$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$z^{(2)} = \Theta^{(1)} x$$

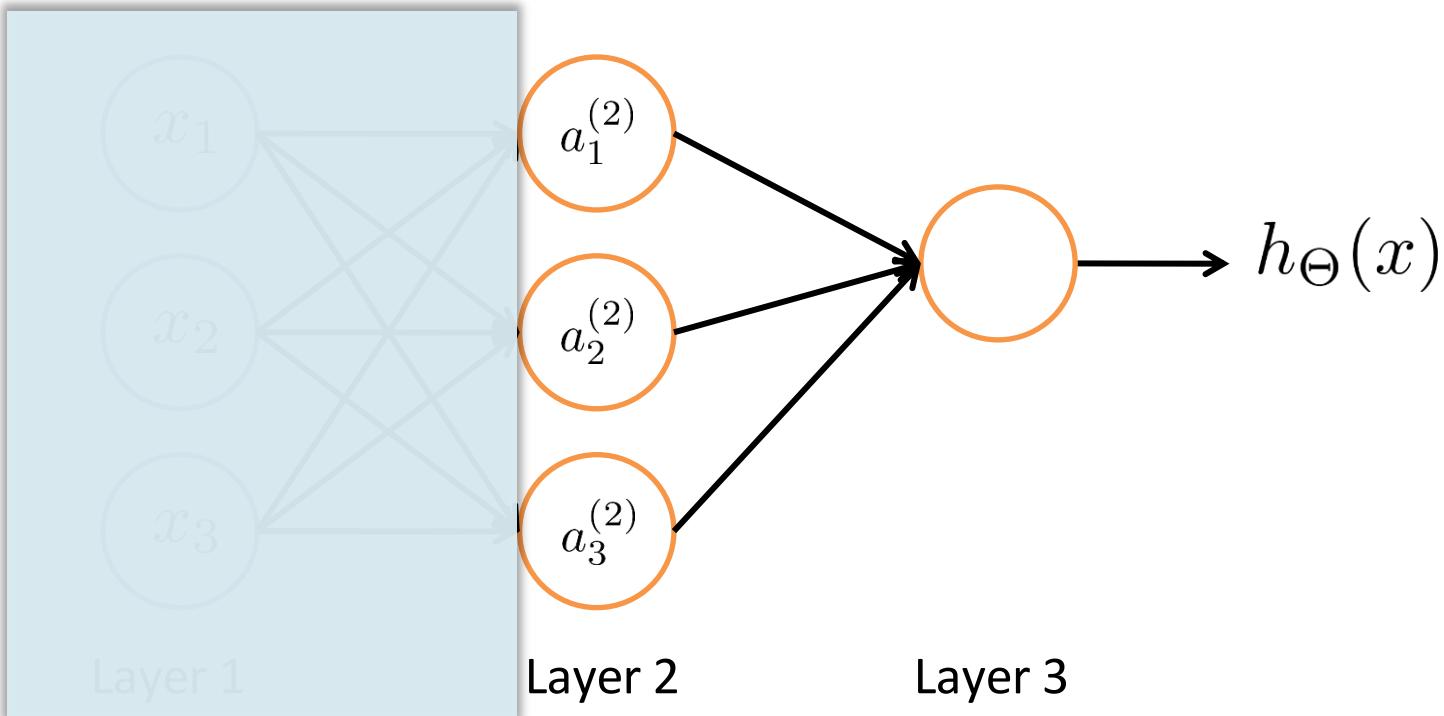
$$a^{(2)} = g(z^{(2)})$$

Add $a_0^{(2)} = 1$.

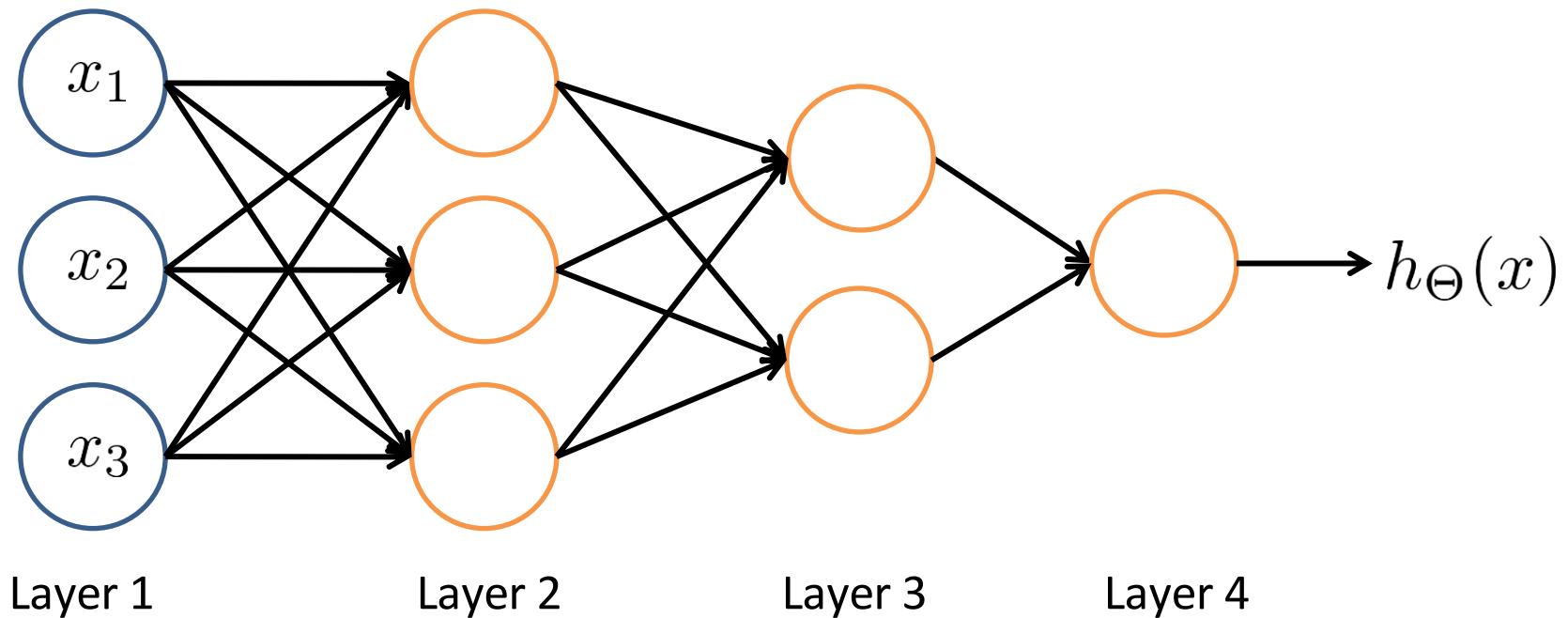
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

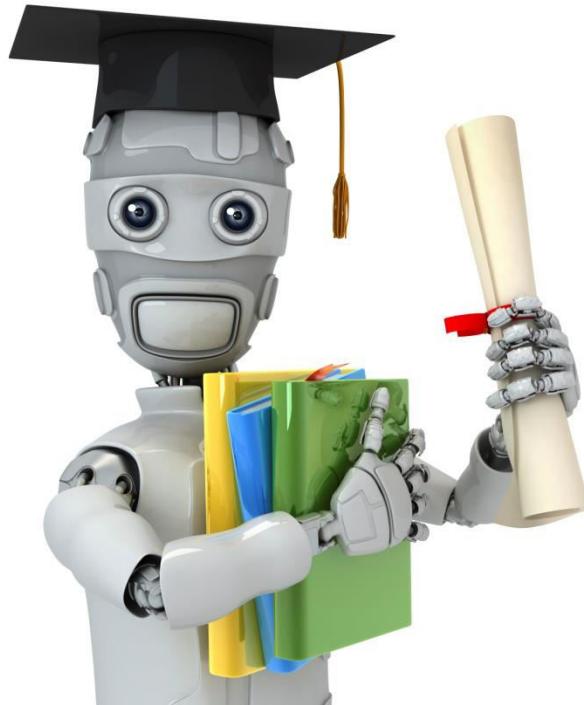
$$h_{\Theta}(x) = a^{(3)} = g(z^{(3)})$$

Neural Network learning its own features



Other network architectures





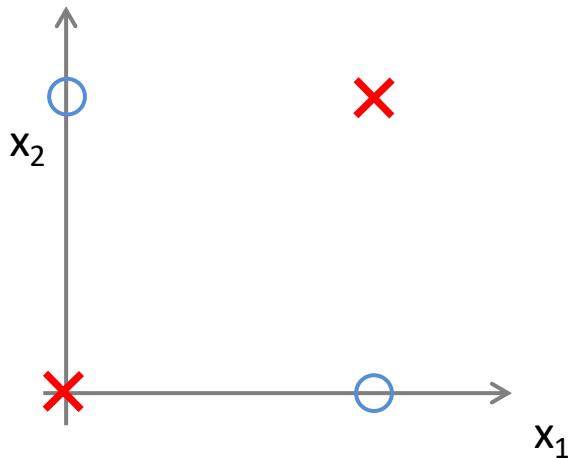
Machine Learning

Neural Networks: Representation

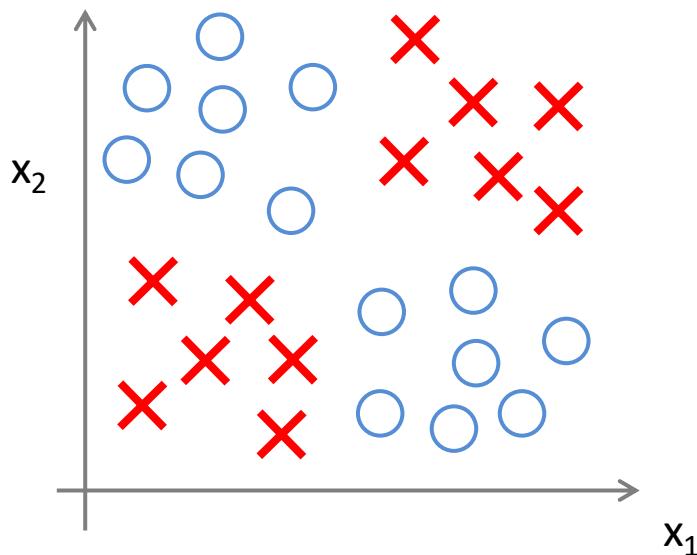
Examples and intuitions I

Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).



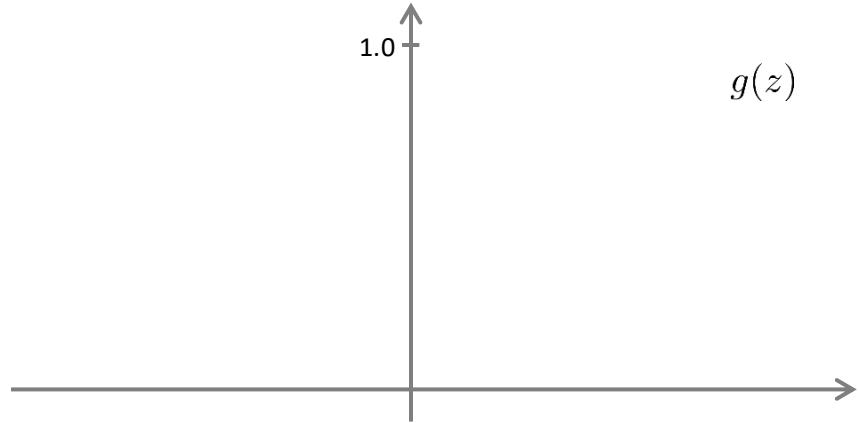
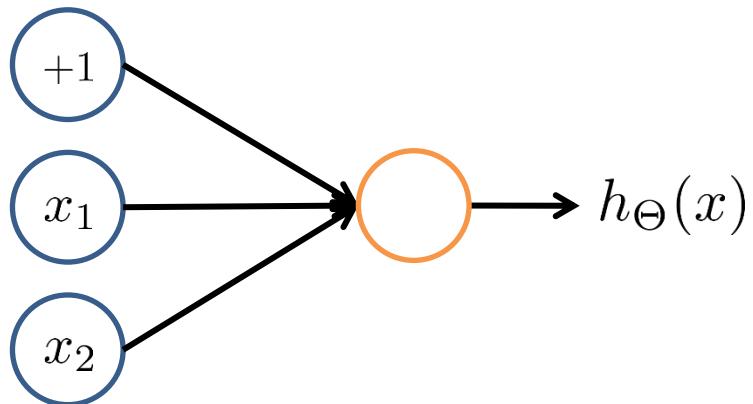
$$\begin{aligned}y &= x_1 \text{ XOR } x_2 \\x_1 \text{ XNOR } x_2 \\&\text{NOT } (x_1 \text{ XOR } x_2)\end{aligned}$$



Simple example: AND

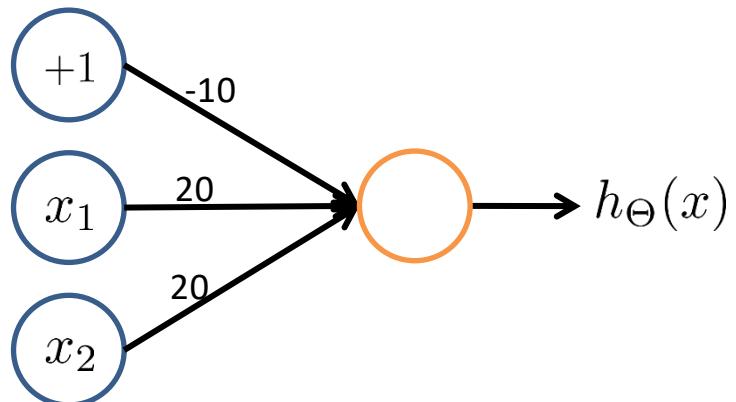
$$x_1, x_2 \in \{0, 1\}$$

$$y = x_1 \text{ AND } x_2$$

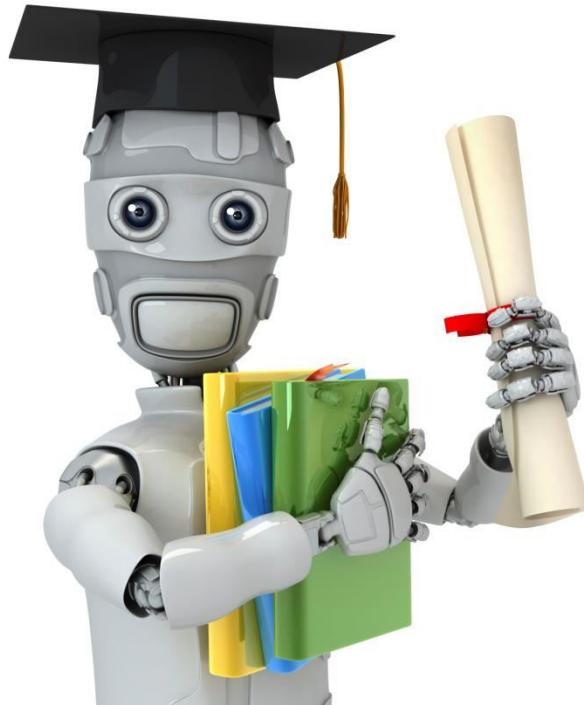


x_1	x_2	$h_{\Theta}(x)$
0	0	0
0	1	0
1	0	0
1	1	1

Example: OR function



x_1	x_2	$h_{\Theta}(x)$
0	0	0
0	1	1
1	0	1
1	1	1



Machine Learning

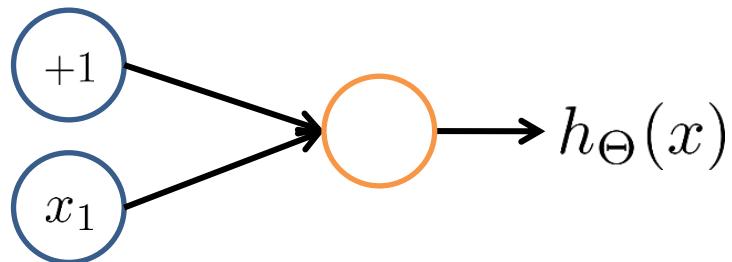
Neural Networks: Representation

Examples and intuitions II

x_1 AND x_2

x_1 OR x_2

Negation:

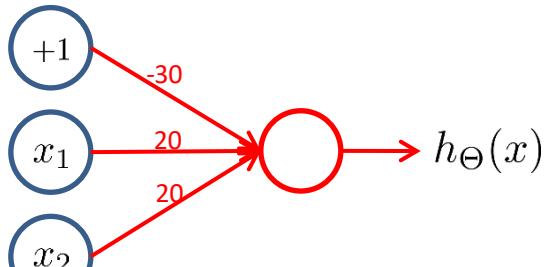


x_1	$h_{\Theta}(x)$
0	1
1	0

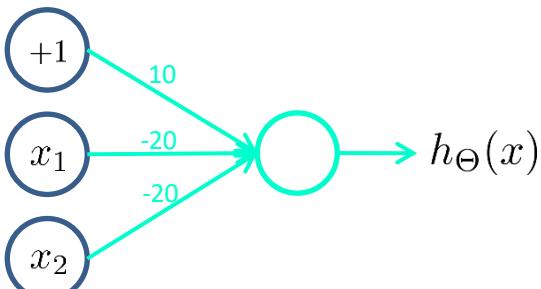
$$h_{\Theta}(x) = g(10 - 20x_1)$$

(NOT x_1) AND (NOT x_2)

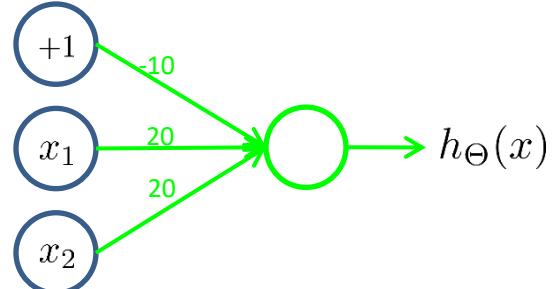
Putting it together: x_1 XNOR x_2



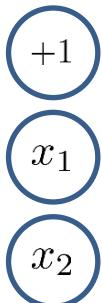
x_1 AND x_2



(NOT x_1) AND (NOT x_2)

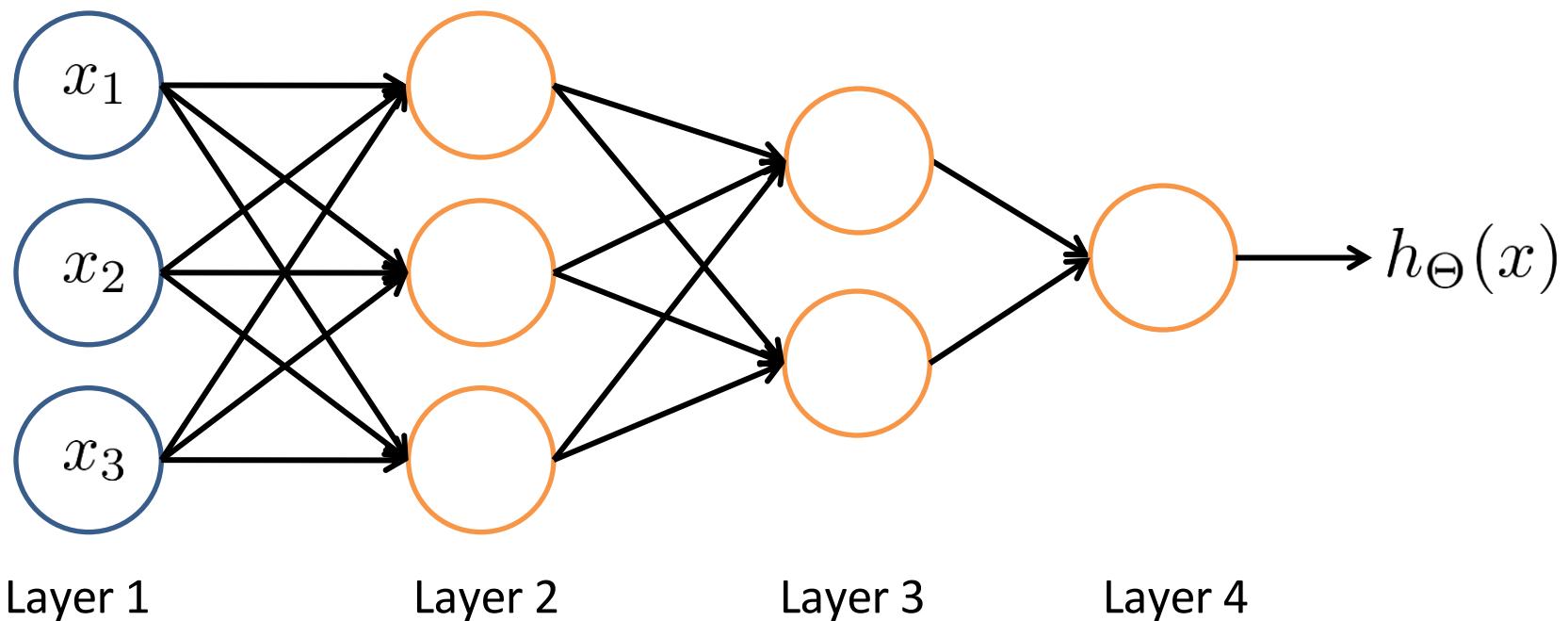


x_1 OR x_2

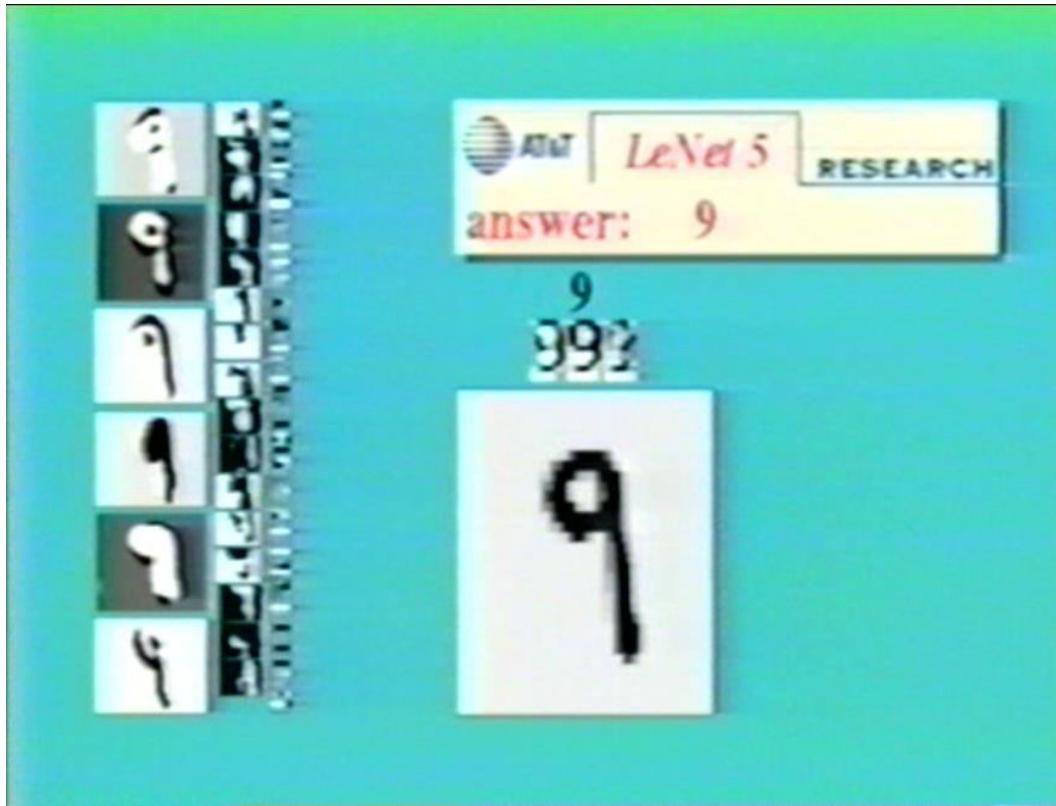


x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_{\Theta}(x)$
0	0			
0	1			
1	0			
1	1			

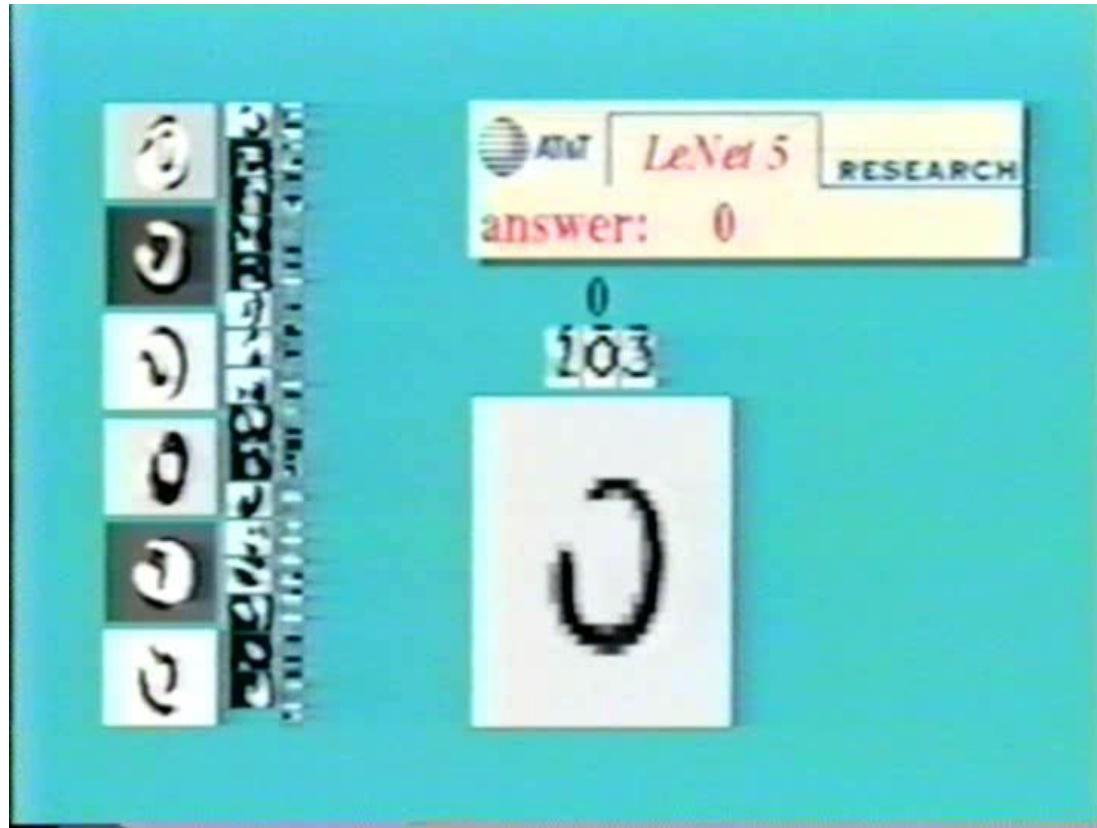
Neural Network intuition

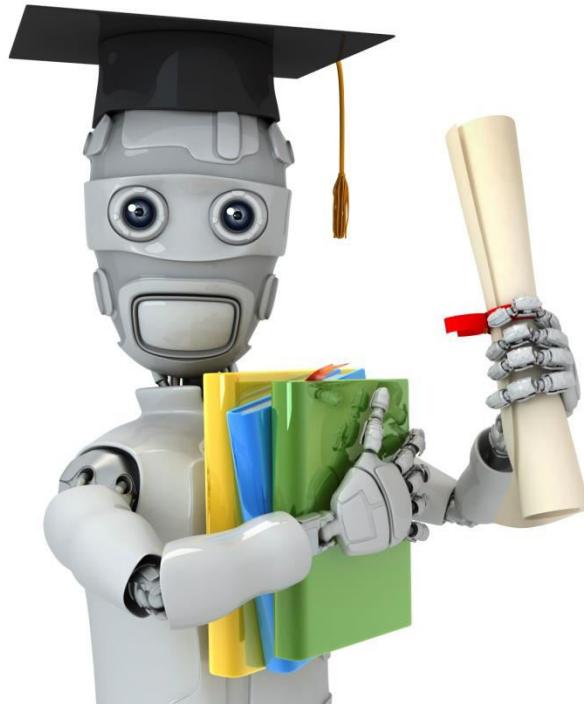


Handwritten digit classification



Handwritten digit classification





Machine Learning

Neural Networks: Representation

Multi-class classification

Multiple output units: One-vs-all.



Pedestrian



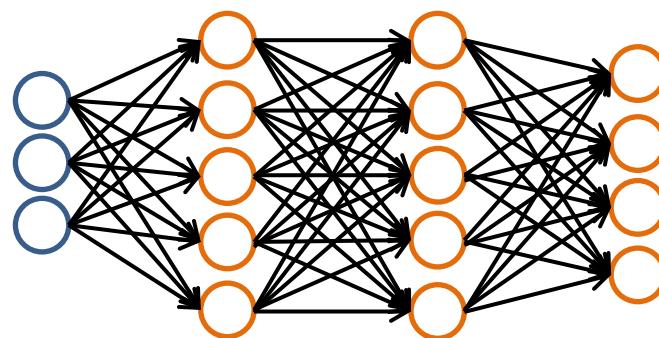
Car



Motorcycle



Truck

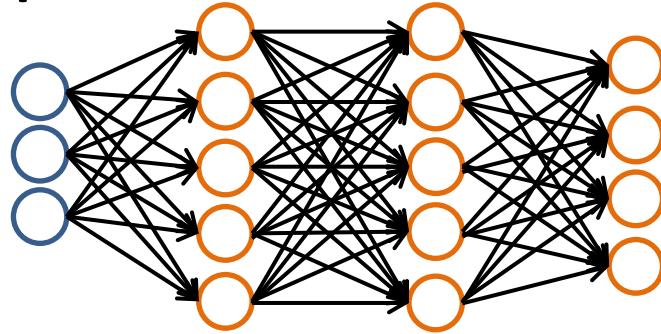


$$h_{\Theta}(x) \in \mathbb{R}^4$$

Want $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Multiple output units: One-vs-all.



$$h_{\Theta}(x) \in \mathbb{R}^4$$

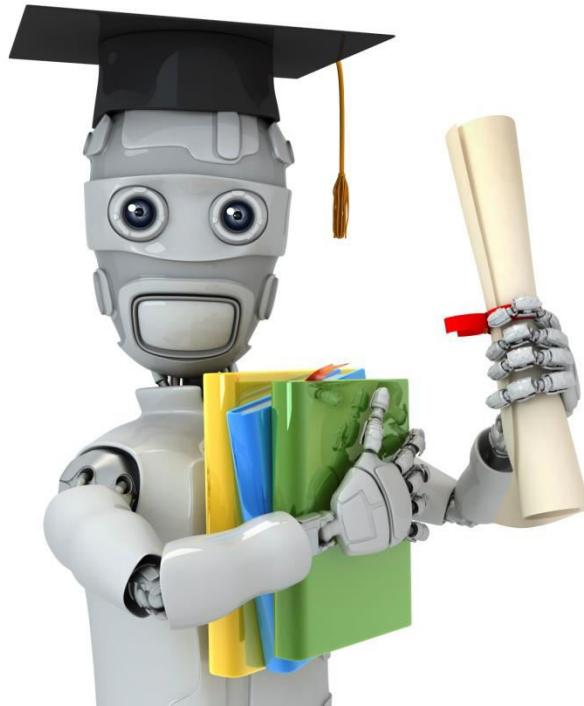
Want $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle

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

$y^{(i)}$ one of $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

pedestrian car motorcycle truck

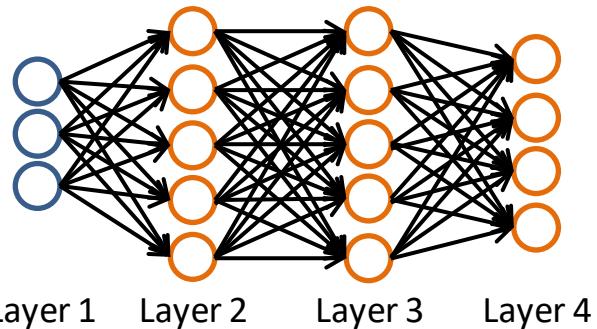


Machine Learning

Neural Networks: Learning

Cost function

Neural Network (Classification)



Binary classification

$y = 0$ or 1

1 output unit

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

L = total no. of layers in network

s_l = no. of units (not counting bias unit) in layer l

Multi-class classification (K classes)

$$y \in \mathbb{R}^K \text{ E.g. } \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

pedestrian car motorcycle truck

K output units

Cost function

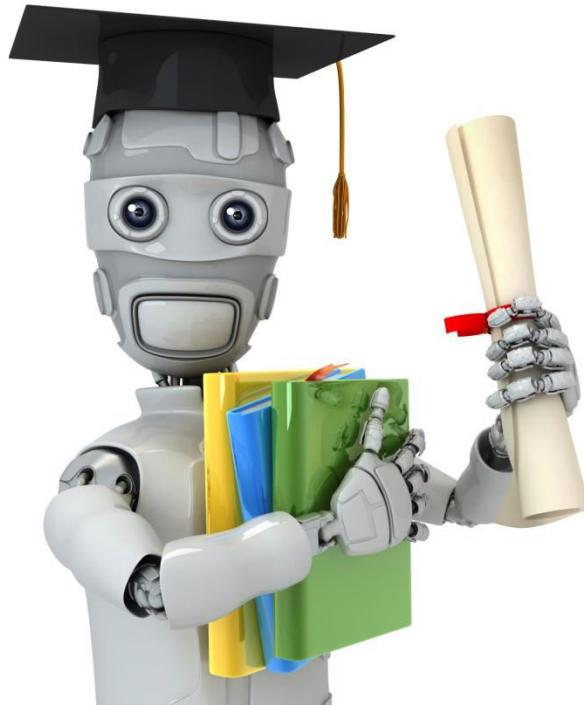
Logistic regression:

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Neural network:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$\begin{aligned} J(\Theta) &= -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k) \right] \\ &\quad + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2 \end{aligned}$$



Machine Learning

Neural Networks: Learning

Backpropagation algorithm

Gradient computation

$$J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log h_\theta(x^{(i)})_k + (1 - y_k^{(i)}) \log(1 - h_\theta(x^{(i)})_k) \right] \\ + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_j^{(l)})^2$$

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

Need code to compute:

- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Gradient computation

Given one training example (x, y):

Forward propagation:

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

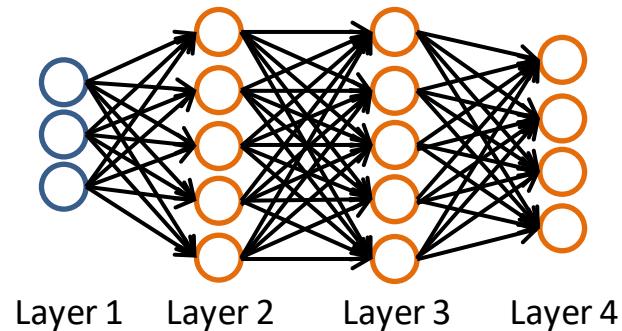
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$$

$$z^{(4)} = \Theta^{(3)} a^{(3)}$$

$$a^{(4)} = h_{\Theta}(x) = g(z^{(4)})$$



Gradient computation: Backpropagation algorithm

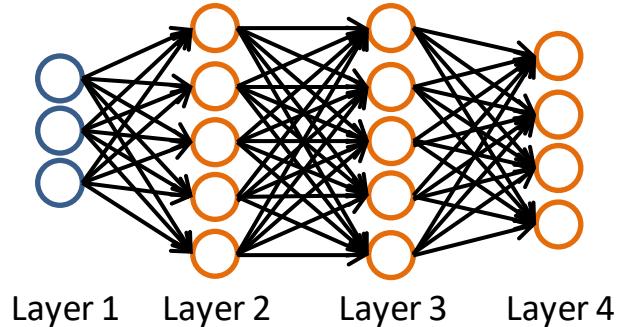
Intuition: $\delta_j^{(l)}$ = “error” of node j in layer l .

For each output unit (layer $L = 4$)

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} * g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} * g'(z^{(2)})$$



Backpropagation algorithm

Training set $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j).

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \dots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

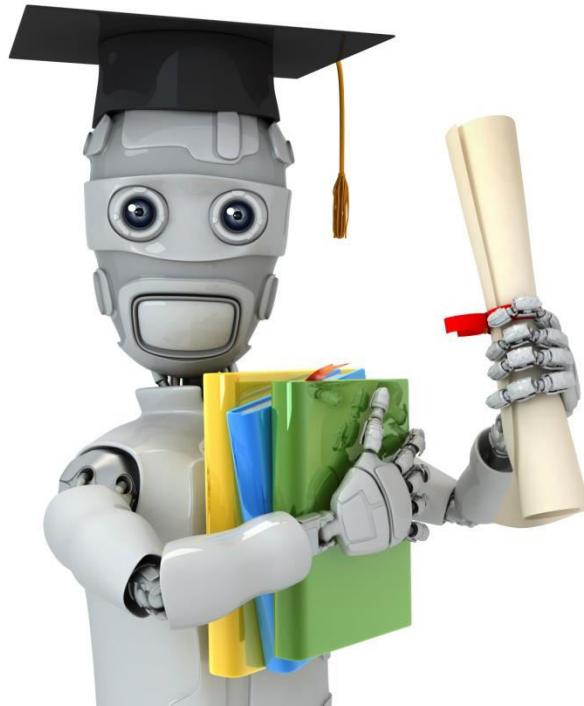
Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} \quad \text{if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \quad \text{if } j = 0$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

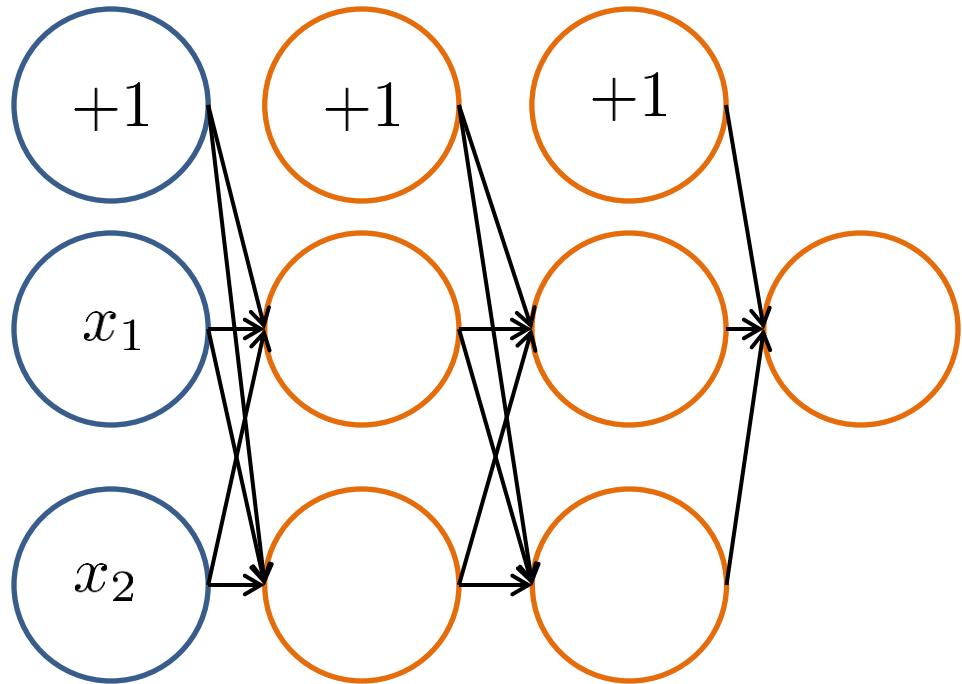


Machine Learning

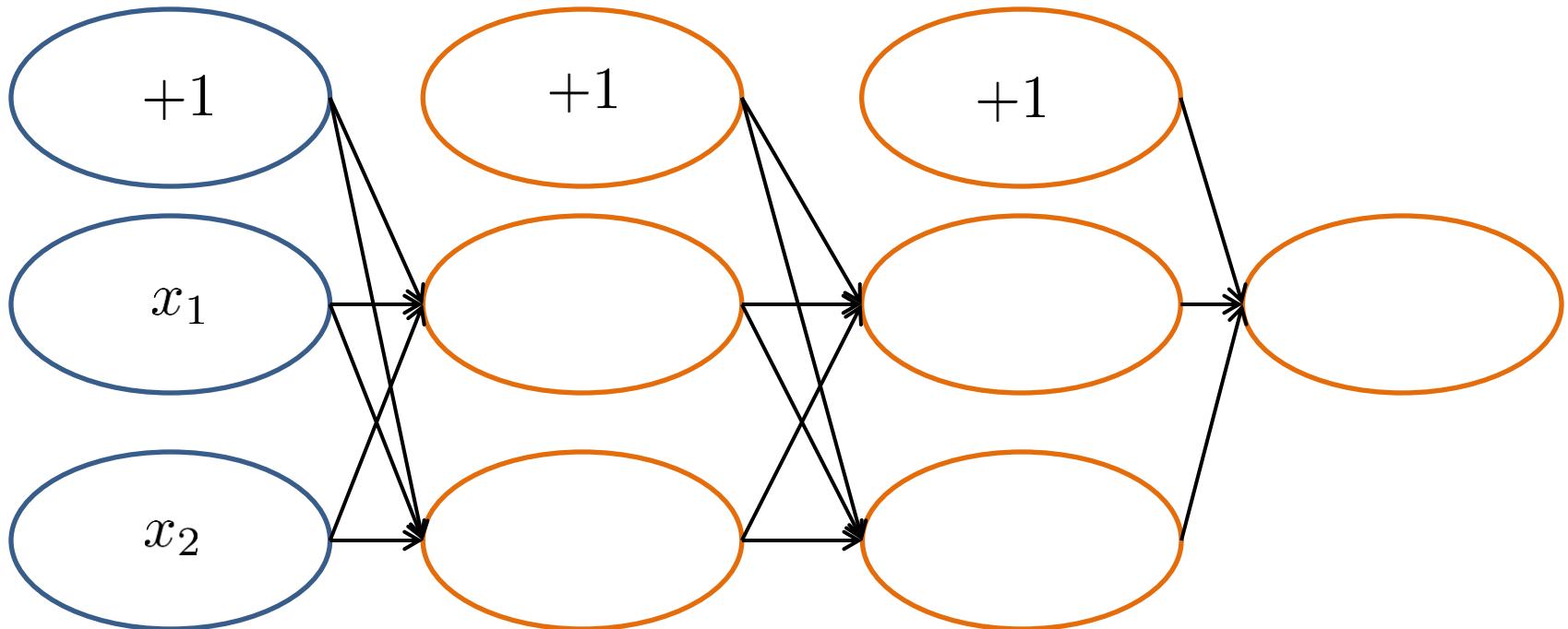
Neural Networks: Learning

Backpropagation intuition

Forward Propagation



Forward Propagation



What is backpropagation doing?

$$J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h_\Theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - (h_\Theta(x^{(i)}))) \right] \\ + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

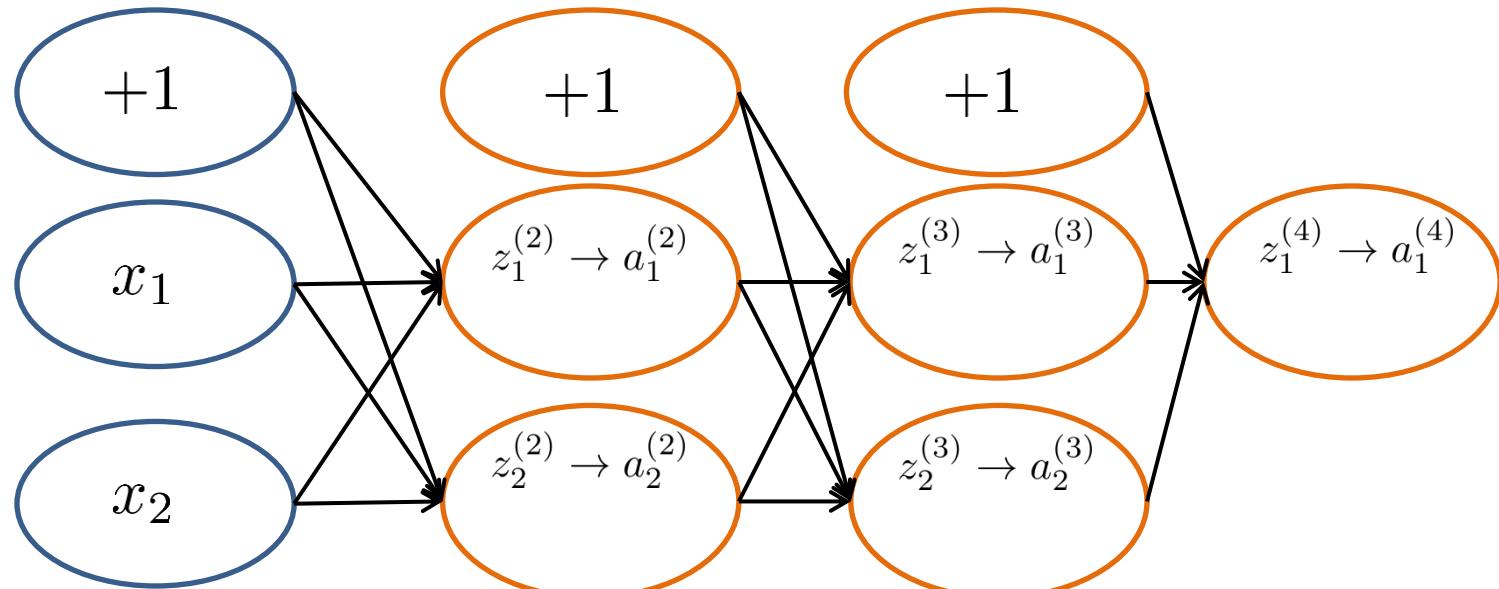
Focusing on a single example $x^{(i)}$, $y^{(i)}$, the case of 1 output unit, and ignoring regularization ($\lambda = 0$),

$$\text{cost}(i) = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log h_\Theta(x^{(i)})$$

(Think of $\text{cost}(i) \approx (h_\Theta(x^{(i)}) - y^{(i)})^2$)

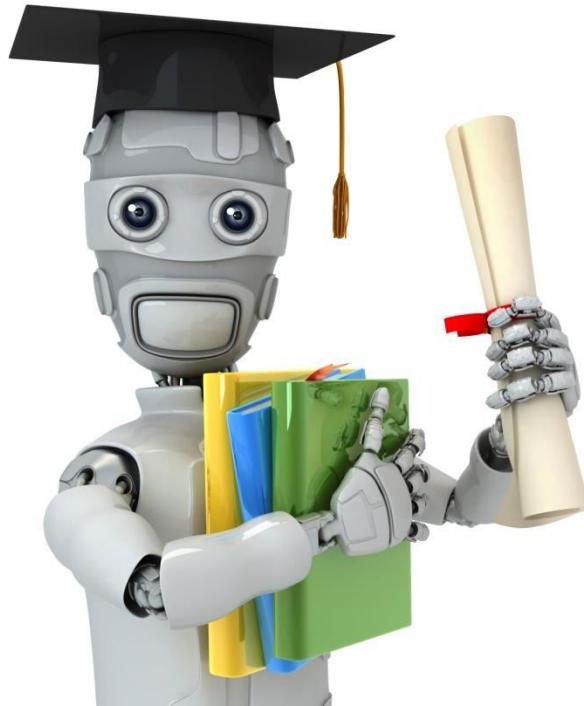
I.e. how well is the network doing on example i?

Forward Propagation



$\delta_j^{(l)}$ = “error” of cost for $a_j^{(l)}$ (unit j in layer l).

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(i)$ (for $j \geq 0$), where
 $\text{cost}(i) = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\Theta(x^{(i)}))$



Machine Learning

Neural Networks: Learning

Implementation note:
Unrolling parameters

Advanced optimization

```
function [jVal, gradient] = costFunction(theta)
...
optTheta = fminunc(@costFunction, initialTheta, options)
```

Neural Network (L=4):

$\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$ - matrices (**Theta1**, **Theta2**, **Theta3**)

$D^{(1)}, D^{(2)}, D^{(3)}$ - matrices (**D1**, **D2**, **D3**)

“Unroll” into vectors

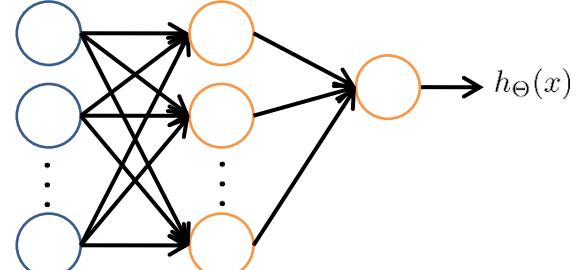
Example

$$s_1 = 10, s_2 = 10, s_3 = 1$$

$$\Theta^{(1)} \in \mathbb{R}^{10 \times 11}, \Theta^{(2)} \in \mathbb{R}^{10 \times 11}, \Theta^{(3)} \in \mathbb{R}^{1 \times 11}$$

$$D^{(1)} \in \mathbb{R}^{10 \times 11}, D^{(2)} \in \mathbb{R}^{10 \times 11}, D^{(3)} \in \mathbb{R}^{1 \times 11}$$

```
thetaVec = [ Theta1(:); Theta2(:); Theta3(:) ];
DVec = [D1(:); D2(:); D3(:)];  
  
Theta1 = reshape(thetaVec(1:110),10,11);
Theta2 = reshape(thetaVec(111:220),10,11);
Theta3 = reshape(thetaVec(221:231),1,11);
```



Learning Algorithm

Have initial parameters $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$.

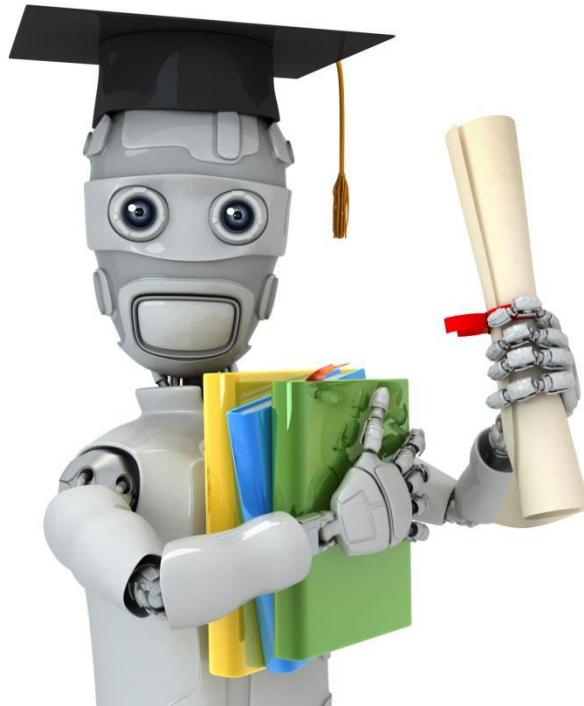
Unroll to get `initialTheta` to pass to
`fminunc(@costFunction, initialTheta, options)`

```
function [jval, gradientVec] = costFunction(thetaVec)
```

From `thetaVec`, get $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$.

Use forward prop/back prop to compute $D^{(1)}, D^{(2)}, D^{(3)}$ and $J(\Theta)$.

Unroll $D^{(1)}, D^{(2)}, D^{(3)}$ to get `gradientVec`.

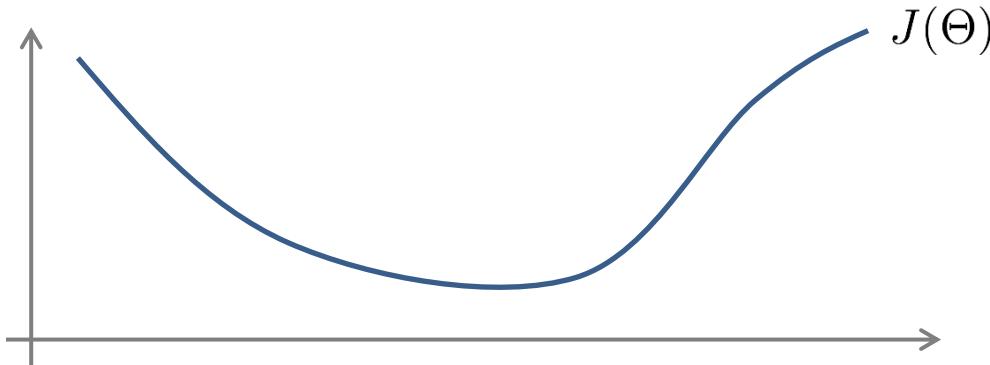


Machine Learning

Neural Networks: Learning

Gradient checking

Numerical estimation of gradients



```
Implement: gradApprox = (J(theta + EPSILON) - J(theta - EPSILON))  
/ (2*EPSILON)
```

Parameter vector θ

$\theta \in \mathbb{R}^n$ (E.g. θ is “unrolled” version of $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$)

$$\theta = \theta_1, \theta_2, \theta_3, \dots, \theta_n$$

$$\frac{\partial}{\partial \theta_1} J(\theta) \approx \frac{J(\theta_1 + \epsilon, \theta_2, \theta_3, \dots, \theta_n) - J(\theta_1 - \epsilon, \theta_2, \theta_3, \dots, \theta_n)}{2\epsilon}$$

$$\frac{\partial}{\partial \theta_2} J(\theta) \approx \frac{J(\theta_1, \theta_2 + \epsilon, \theta_3, \dots, \theta_n) - J(\theta_1, \theta_2 - \epsilon, \theta_3, \dots, \theta_n)}{2\epsilon}$$

⋮

$$\frac{\partial}{\partial \theta_n} J(\theta) \approx \frac{J(\theta_1, \theta_2, \theta_3, \dots, \theta_n + \epsilon) - J(\theta_1, \theta_2, \theta_3, \dots, \theta_n - \epsilon)}{2\epsilon}$$

```
for i = 1:n,
    thetaPlus = theta;
    thetaPlus(i) = thetaPlus(i) + EPSILON;
    thetaMinus = theta;
    thetaMinus(i) = thetaMinus(i) - EPSILON;
    gradApprox(i) = (J(thetaPlus) - J(thetaMinus))
                    / (2*EPSILON);
end;
```

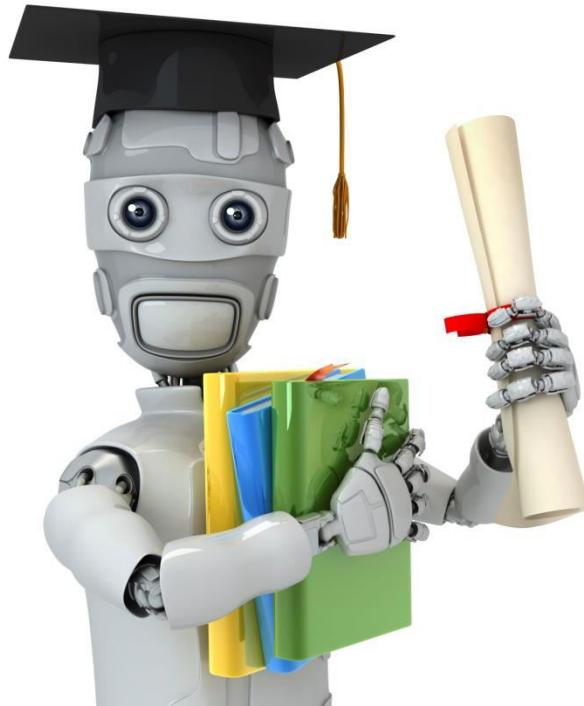
Check that **gradApprox** \approx DVec

Implementation Note:

- Implement backprop to compute **DVec** (unrolled $D^{(1)}, D^{(2)}, D^{(3)}$).
- Implement numerical gradient check to compute **gradApprox**.
- Make sure they give similar values.
- Turn off gradient checking. Using backprop code for learning.

Important:

- Be sure to disable your gradient checking code before training your classifier. If you run numerical gradient computation on every iteration of gradient descent (or in the inner loop of **costFunction (...)**) your code will be very slow.



Machine Learning

Neural Networks: Learning

Random initialization

Initial value of Θ

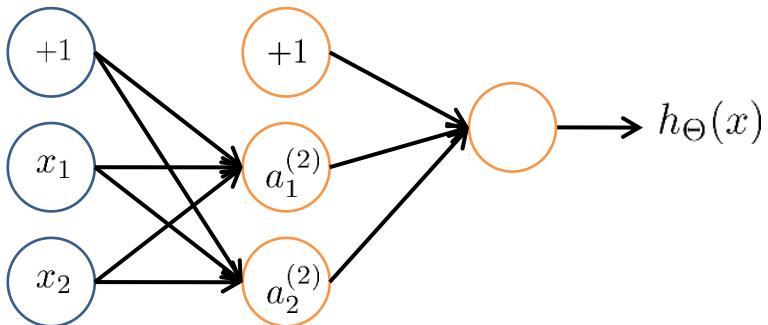
For gradient descent and advanced optimization method, need initial value for Θ .

```
optTheta = fminunc(@costFunction,  
                    initialTheta, options)
```

Consider gradient descent

Set `initialTheta = zeros(n,1)` ?

Zero initialization



$$\Theta_{ij}^{(l)} = 0 \text{ for all } i, j, l.$$

After each update, parameters corresponding to inputs going into each of two hidden units are identical.

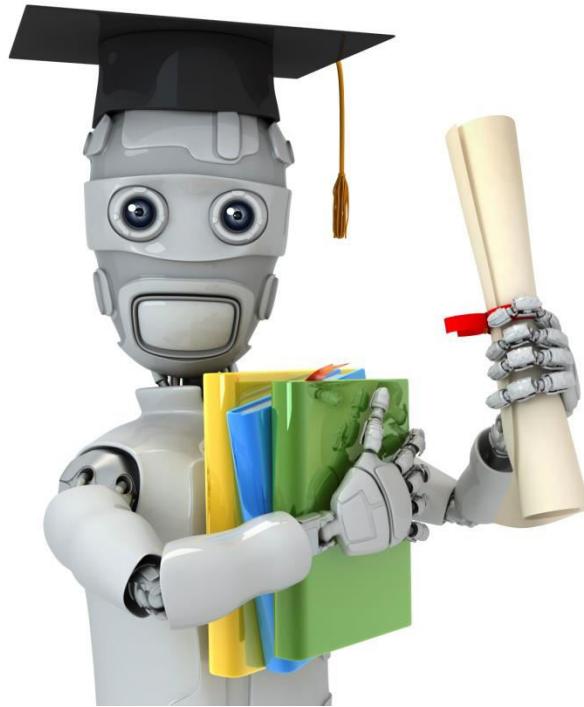
Random initialization: Symmetry breaking

Initialize each $\Theta_{ij}^{(l)}$ to a random value in $[-\epsilon, \epsilon]$
(i.e. $-\epsilon \leq \Theta_{ij}^{(l)} \leq \epsilon$)

E.g.

```
Theta1 = rand(10,11)*(2*INIT_EPSILON)
        - INIT_EPSILON;
```

```
Theta2 = rand(1,11)*(2*INIT_EPSILON)
        - INIT_EPSILON;
```



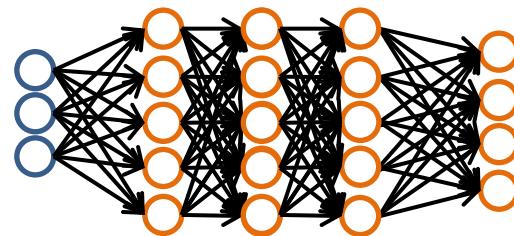
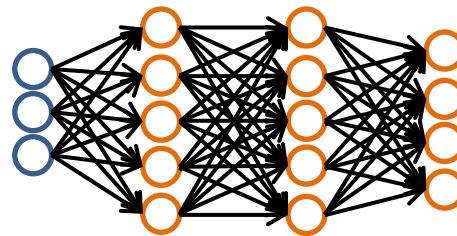
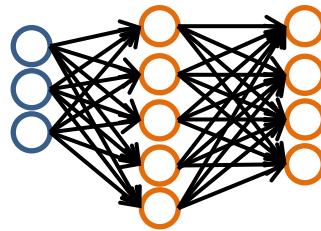
Machine Learning

Neural Networks: Learning

Putting it together

Training a neural network

Pick a network architecture (connectivity pattern between neurons)



No. of input units: Dimension of features $x^{(i)}$

No. output units: Number of classes

Reasonable default: 1 hidden layer, or if >1 hidden layer, have same no. of hidden units in every layer (usually the more the better)

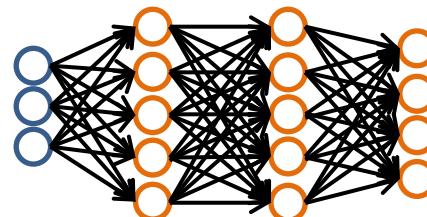
Training a neural network

1. Randomly initialize weights
2. Implement forward propagation to get $h_{\Theta}(x^{(i)})$ for any $x^{(i)}$
3. Implement code to compute cost function $J(\Theta)$
4. Implement backprop to compute partial derivatives $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$

for $i = 1:m$

 Perform forward propagation and backpropagation using
 example $(x^{(i)}, y^{(i)})$

 (Get activations $a^{(l)}$ and delta terms $\delta^{(l)}$ for $l = 2, \dots, L$).

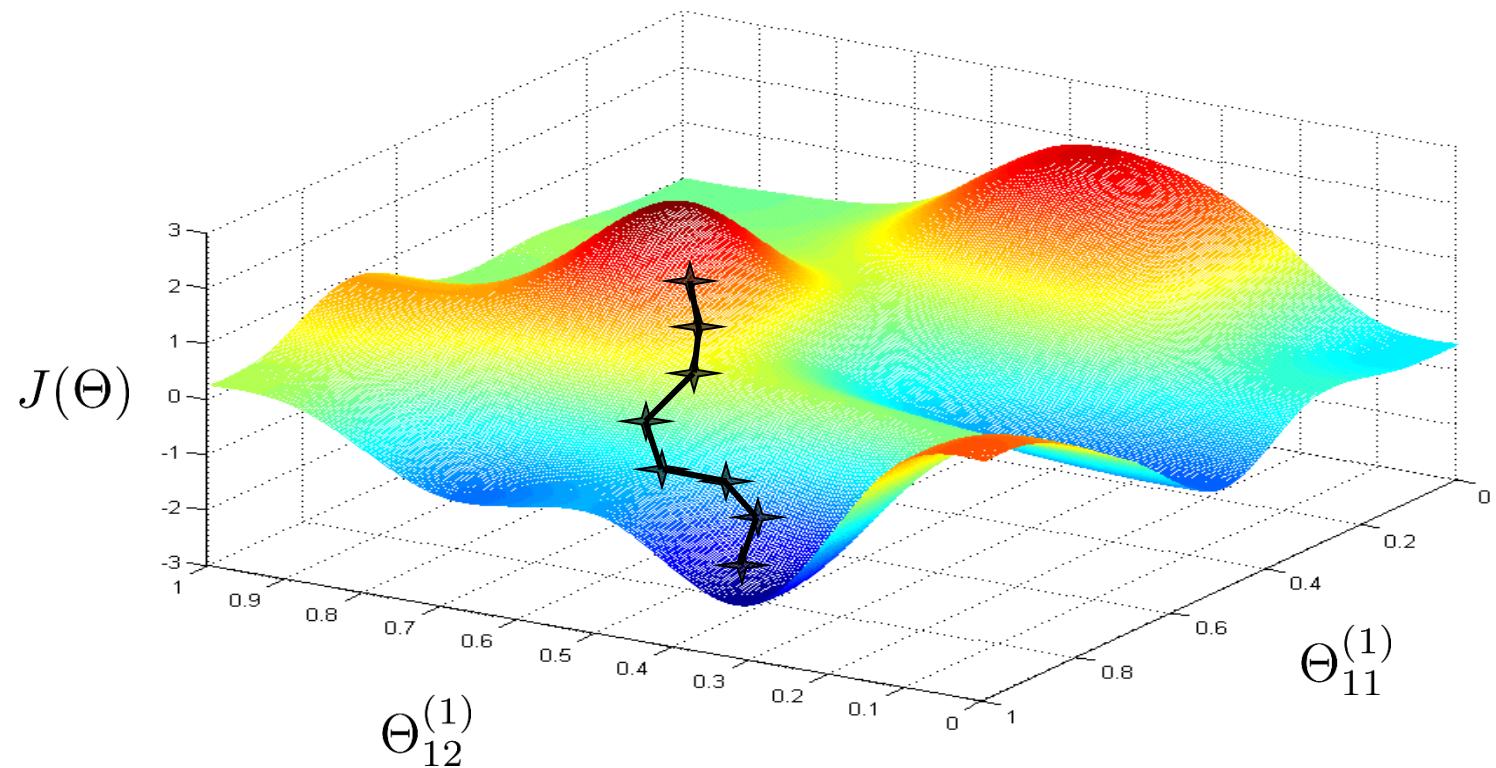


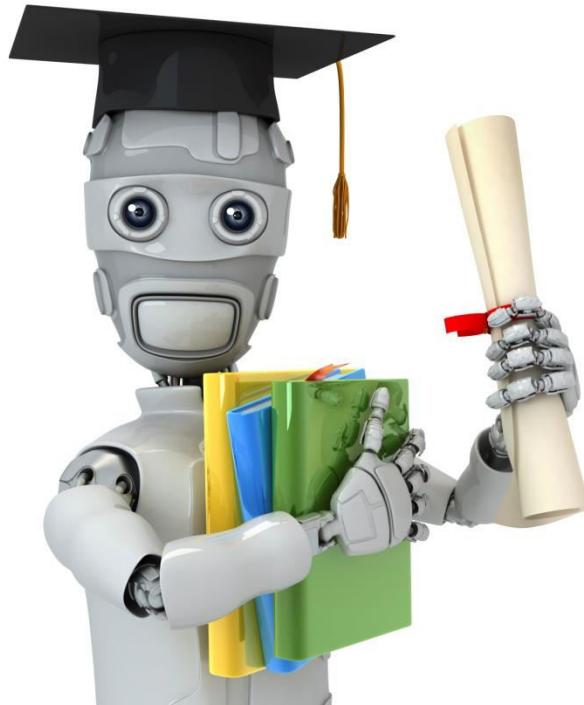
Training a neural network

5. Use gradient checking to compare $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$ computed using backpropagation vs. using numerical estimate of gradient of $J(\Theta)$.

Then disable gradient checking code.

6. Use gradient descent or advanced optimization method with backpropagation to try to minimize $J(\Theta)$ as a function of parameters Θ

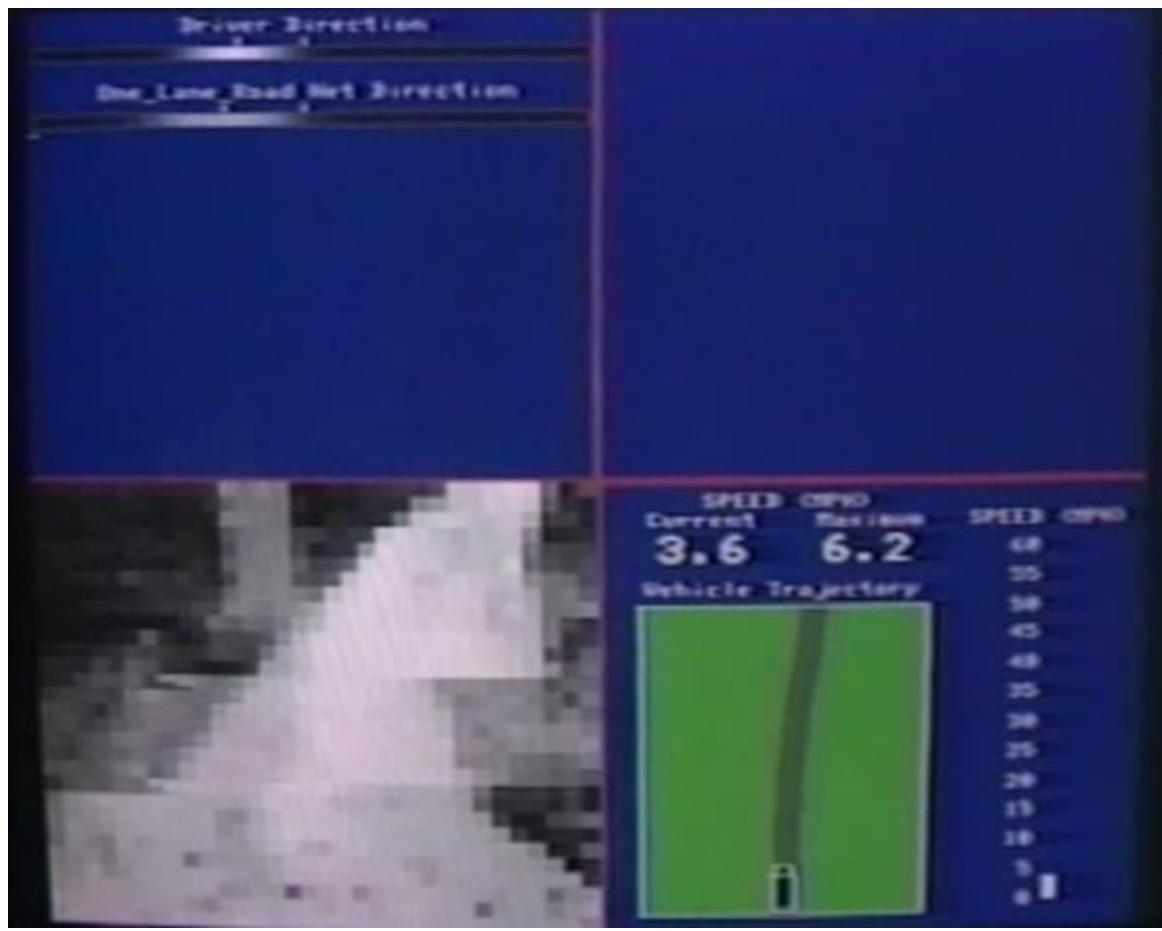




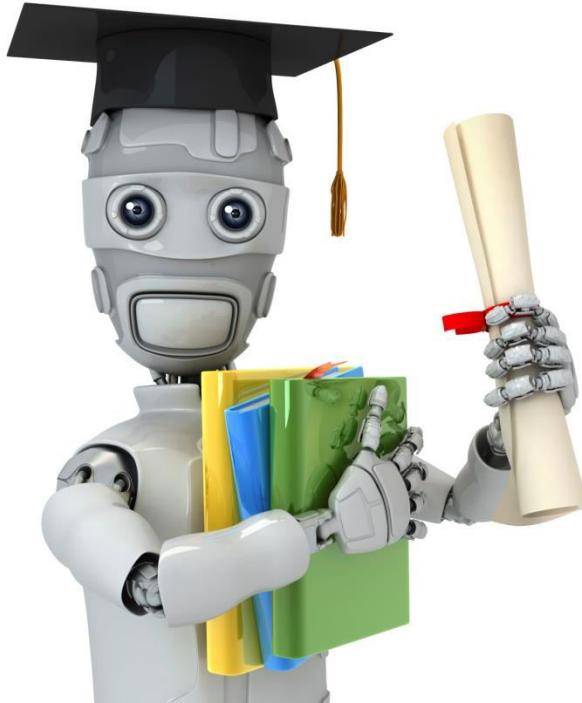
Machine Learning

Neural Networks: Learning

Backpropagation
example: Autonomous
driving (optional)



[Courtesy of Dean Pomerleau]



Machine Learning

Advice for applying
machine learning

Deciding what
to try next

Debugging a learning algorithm:

Suppose you have implemented regularized linear regression to predict housing prices.

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^m \theta_j^2 \right]$$

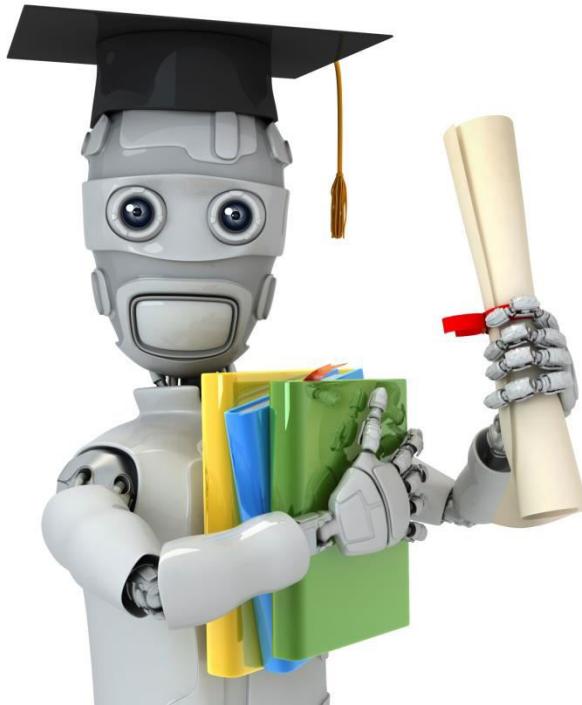
However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2 ,etc.)
- Try decreasing λ
- Try increasing λ

Machine learning diagnostic:

Diagnostic: A test that you can run to gain insight what is/isn't working with a learning algorithm, and gain guidance as to how best to improve its performance.

Diagnostics can take time to implement, but doing so can be a very good use of your time.



Machine Learning

Advice for applying
machine learning

Evaluating a
hypothesis

Evaluating your hypothesis



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

Fails to generalize to new examples not in training set.

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

⋮

x_{100}

Evaluating your hypothesis

Dataset:

Size	Price	
2104	400	$(x^{(1)}, y^{(1)})$
1600	330	$(x^{(2)}, y^{(2)})$
2400	369	\vdots
1416	232	\vdots
3000	540	$(x^{(m)}, y^{(m)})$
1985	300	
1534	315	
1427	199	\longrightarrow
1380	212	\longrightarrow
1494	243	$(x_{test}^{(1)}, y_{test}^{(1)})$
		$(x_{test}^{(2)}, y_{test}^{(2)})$
		\vdots
		$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Training/testing procedure for linear regression

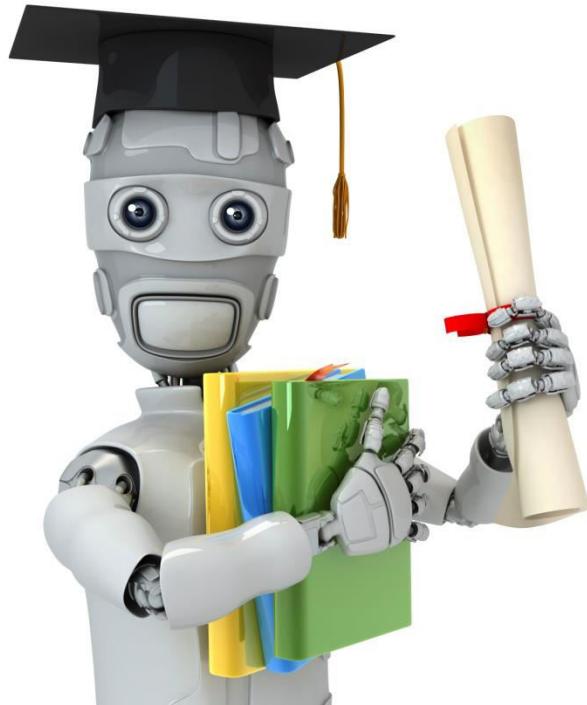
- Learn parameter θ from training data (minimizing training error $J(\theta)$)
- Compute test set error:

Training/testing procedure for logistic regression

- Learn parameter θ from training data
- Compute test set error:

$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log (1 - h_\theta(x_{test}^{(i)}))$$

- Misclassification error (0/1 misclassification error):



Machine Learning

Advice for applying machine learning

Model selection and
training/validation/test
sets

Overfitting example



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

Once parameters $\theta_0, \theta_1, \dots, \theta_4$ were fit to some set of data (training set), the error of the parameters as measured on that data (the training error $J(\theta)$) is likely to be lower than the actual generalization error.

Model selection

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_3 x^3$
- \vdots
10. $h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10}$

Choose $\theta_0 + \dots \theta_5 x^5$

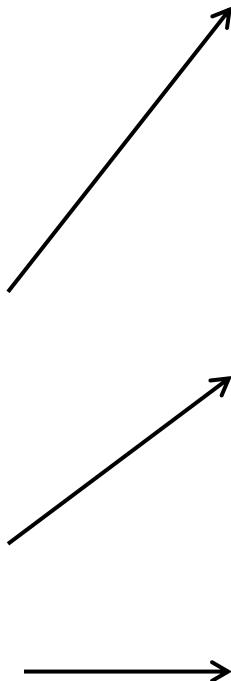
How well does the model generalize? Report test set error $J_{test}(\theta^{(5)})$.

Problem: $J_{test}(\theta^{(5)})$ is likely to be an optimistic estimate of generalization error. I.e. our extra parameter (d = degree of polynomial) is fit to test set.

Evaluating your hypothesis

Dataset:

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243



$$(x^{(1)}, y^{(1)})$$

$$(x^{(2)}, y^{(2)})$$

⋮

$$(x^{(m)}, y^{(m)})$$

$$(x_{cv}^{(1)}, y_{cv}^{(1)})$$

$$(x_{cv}^{(2)}, y_{cv}^{(2)})$$

⋮

$$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$$

$$(x_{test}^{(1)}, y_{test}^{(1)})$$

$$(x_{test}^{(2)}, y_{test}^{(2)})$$

⋮

$$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$$

Train/validation/test error

Training error:

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

Cross Validation error:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

Test error:

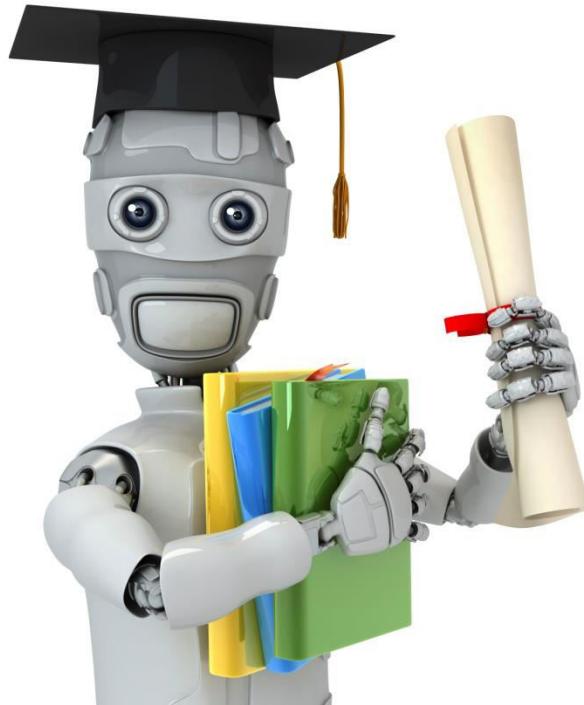
$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

Model selection

1. $h_{\theta}(x) = \theta_0 + \theta_1 x$
2. $h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_3 x^3$
- ⋮
10. $h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10}$

Pick $\theta_0 + \theta_1 x_1 + \dots + \theta_4 x^4$

Estimate generalization error for test set $J_{test}(\theta^{(4)})$

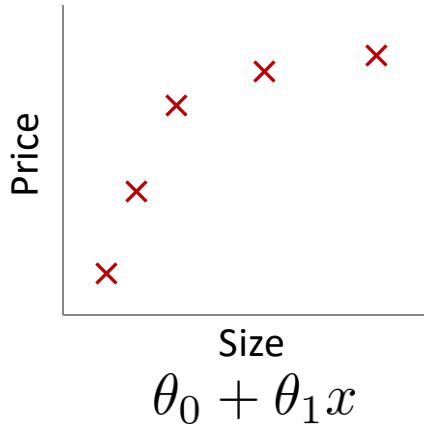


Machine Learning

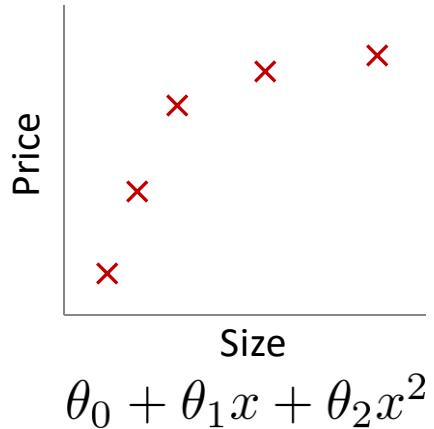
Advice for applying machine learning

Diagnosing bias vs. variance

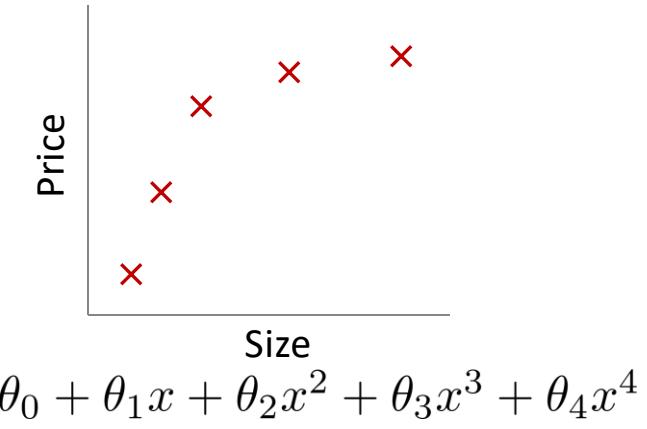
Bias/variance



High bias
(underfit)



“Just right”

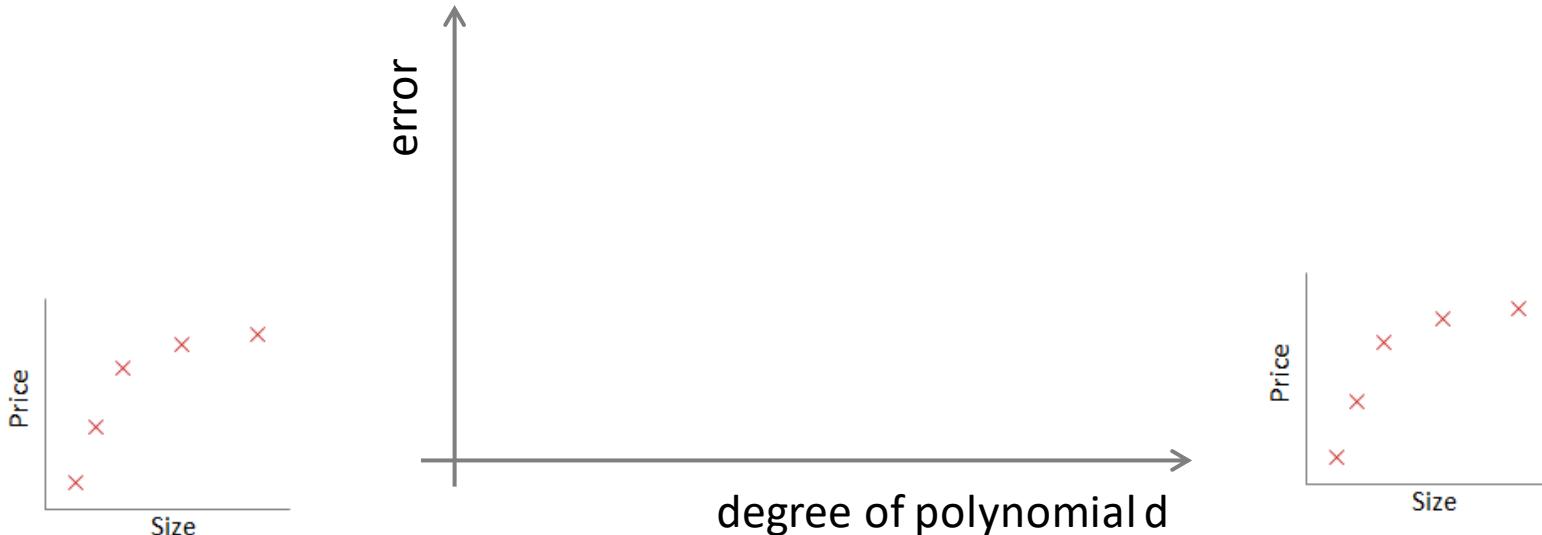


High variance
(overfit)

Bias/variance

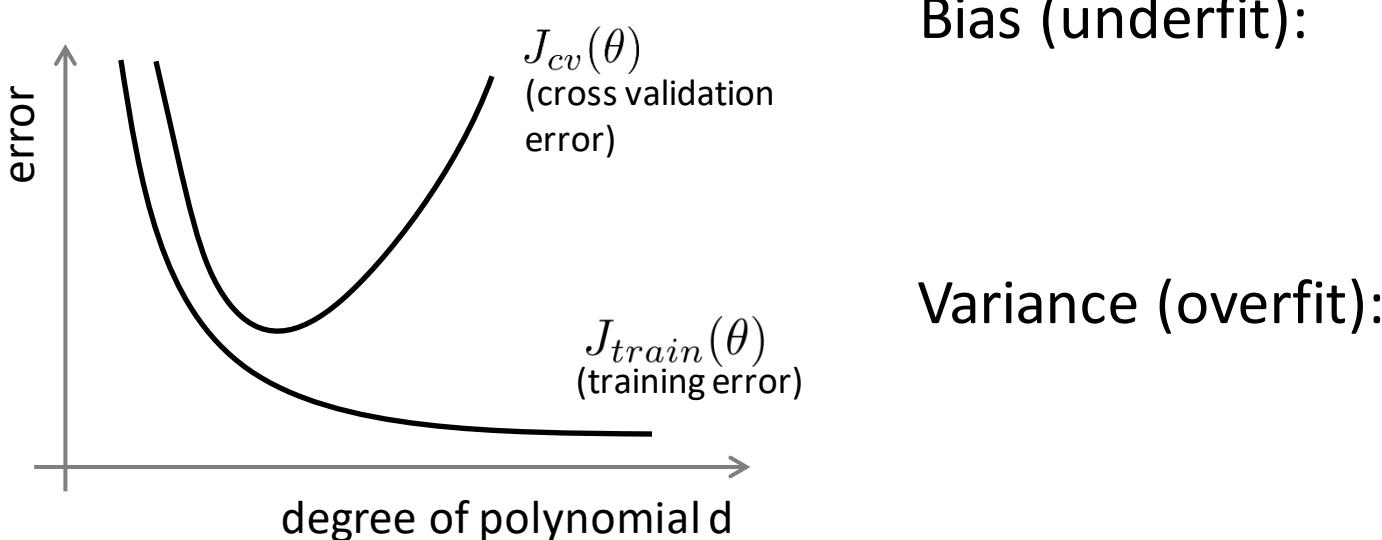
Training error: $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$

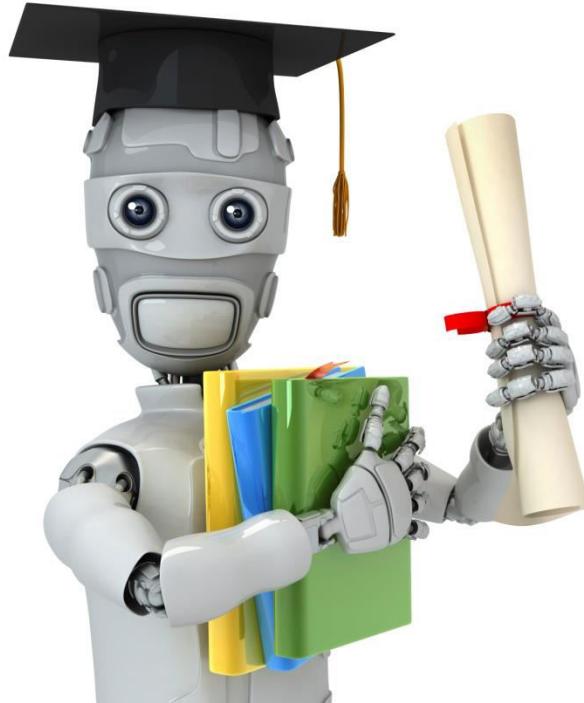
Cross validation error: $J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$



Diagnosing bias vs. variance

Suppose your learning algorithm is performing less well than you were hoping. ($J_{cv}(\theta)$ or $J_{test}(\theta)$ is high.) Is it a bias problem or a variance problem?





Machine Learning

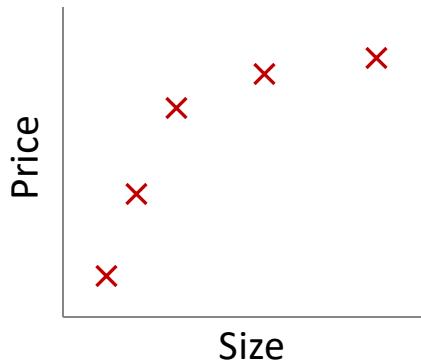
Advice for applying machine learning

Regularization and bias/variance

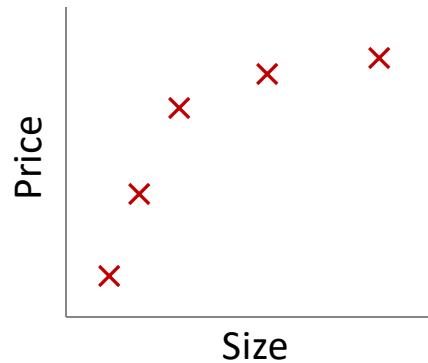
Linear regression with regularization

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

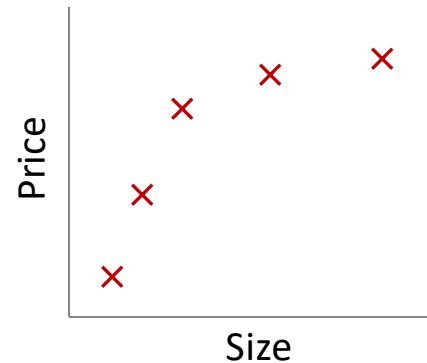
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^m \theta_j^2$$



High bias (underfit)



"Just right"



High variance (overfit)

$\lambda = 10000$. $\theta_1 \approx 0, \theta_2 \approx 0, \dots$

$$h_{\theta}(x) \approx \theta_0$$

Choosing the regularization parameter λ

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

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

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

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_{\theta}(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_{\theta}(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

Choosing the regularization parameter λ

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

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

1. Try $\lambda = 0$
2. Try $\lambda = 0.01$
3. Try $\lambda = 0.02$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$
- ⋮
12. Try $\lambda = 10$

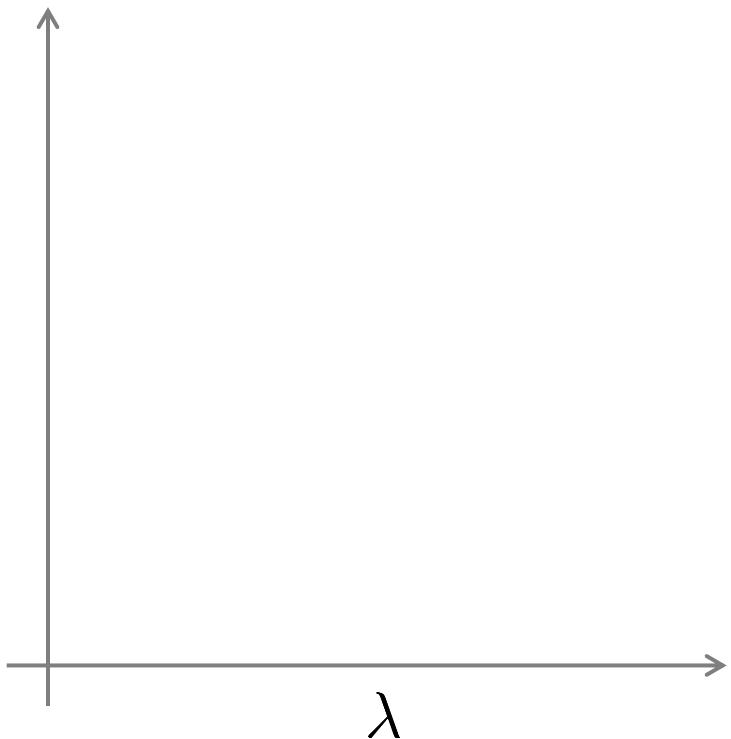
Pick (say) $\theta^{(5)}$. Test error:

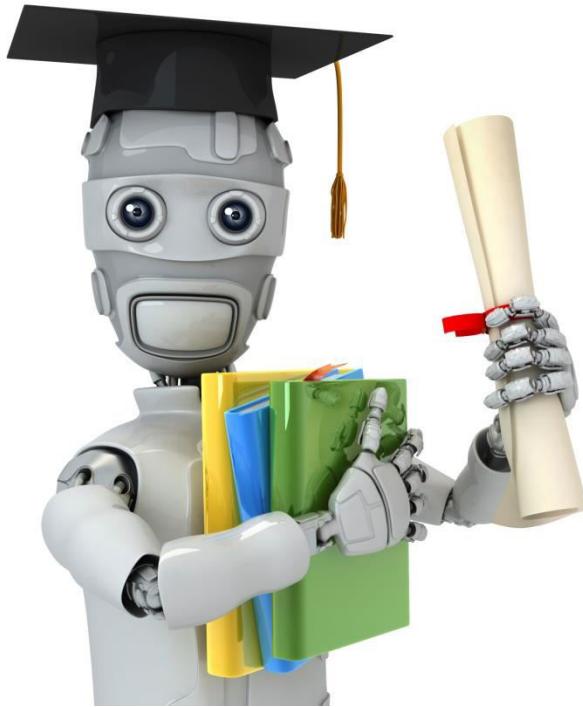
Bias/variance as a function of the regularization parameter λ

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

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

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$





Machine Learning

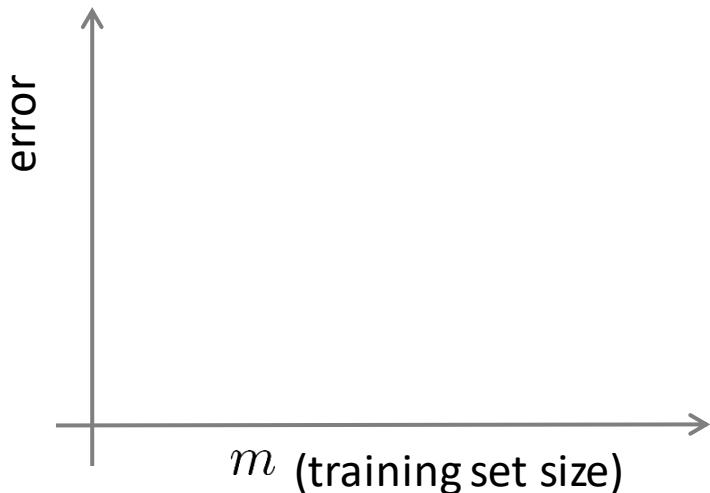
Advice for applying machine learning

Learning curves

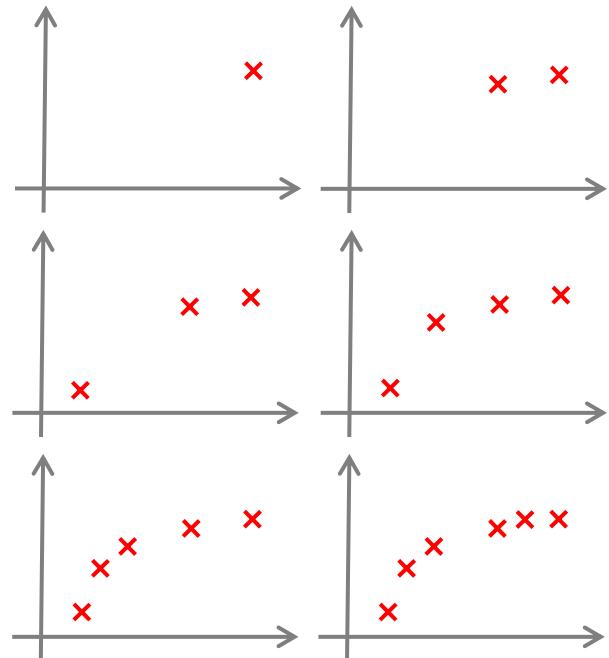
Learning curves

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

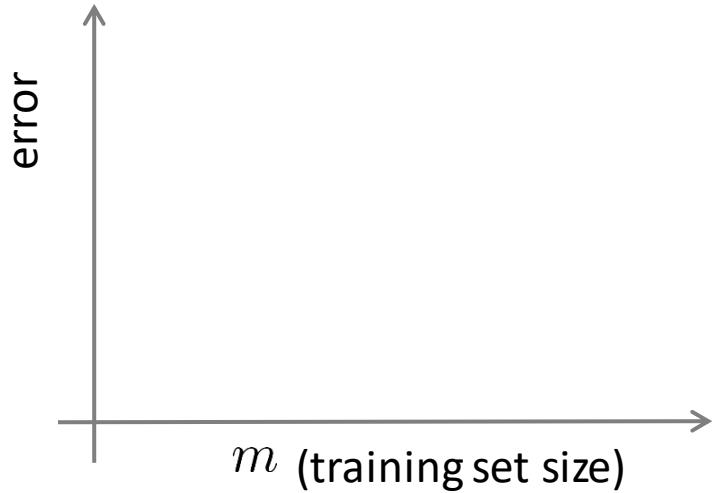
$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$



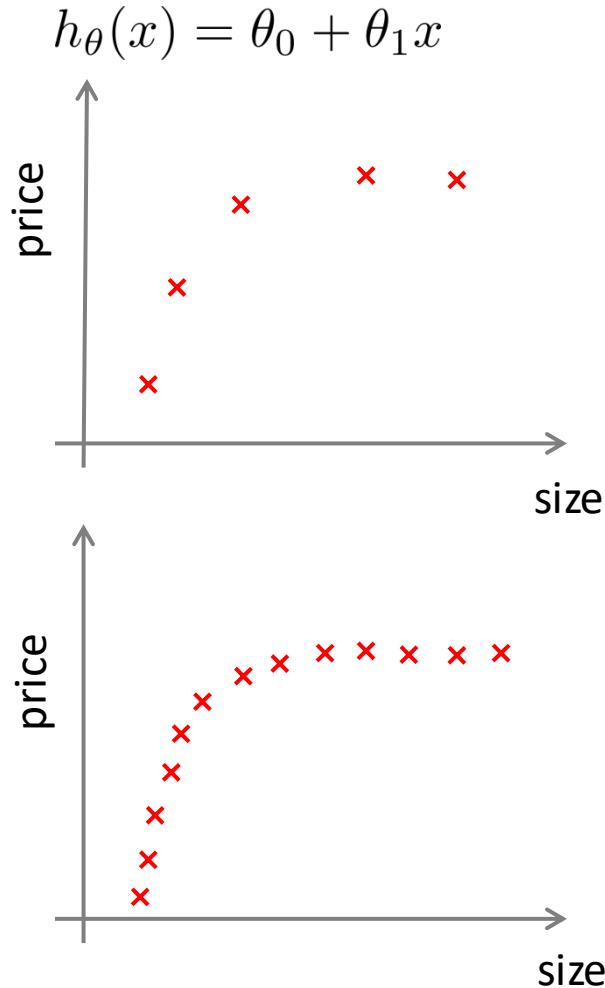
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$



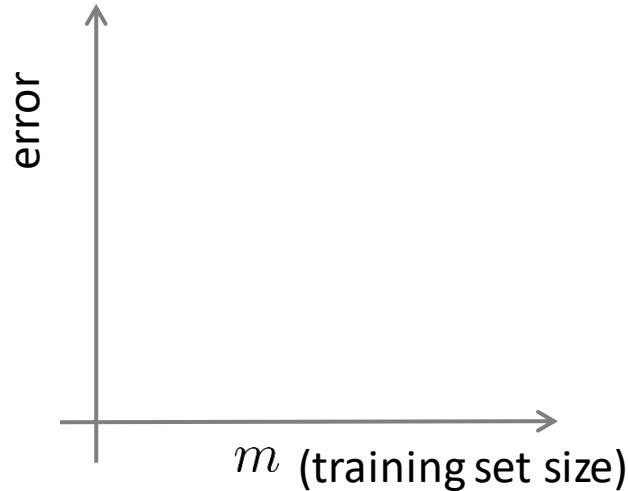
High bias



If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.



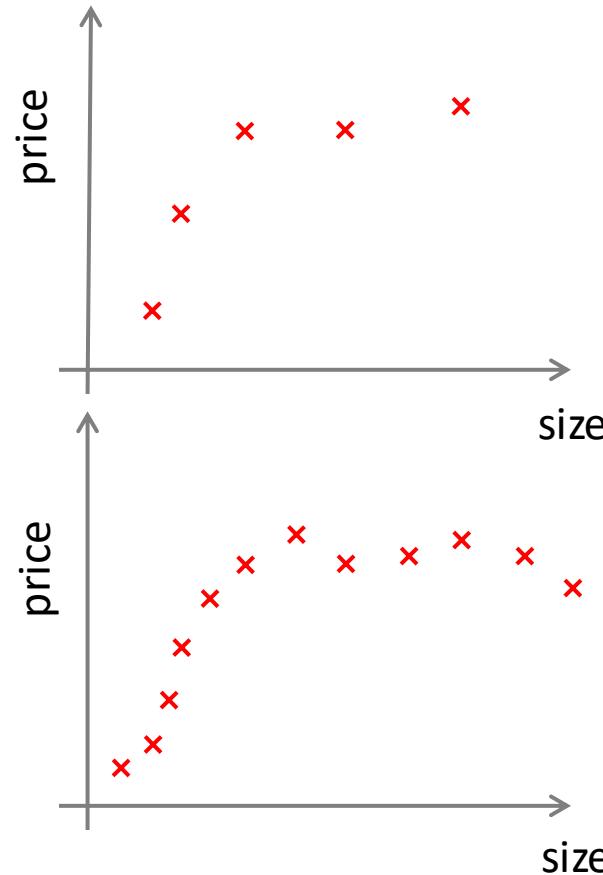
High variance

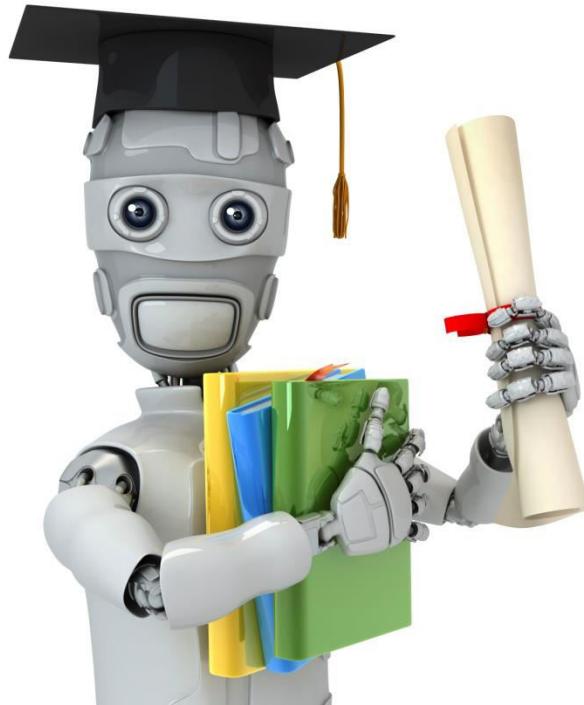


If a learning algorithm is suffering from high variance, getting more training data is likely to help.

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \cdots + \theta_{100} x^{100}$$

(and small λ)





Machine Learning

Advice for applying machine learning

Deciding what to try next (revisited)

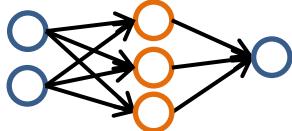
Debugging a learning algorithm:

Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2 ,etc)
- Try decreasing λ
- Try increasing λ

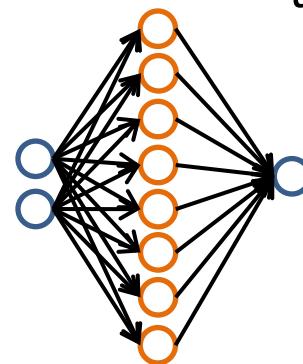
Neural networks and overfitting

“Small” neural network
(fewer parameters; more
prone to underfitting)



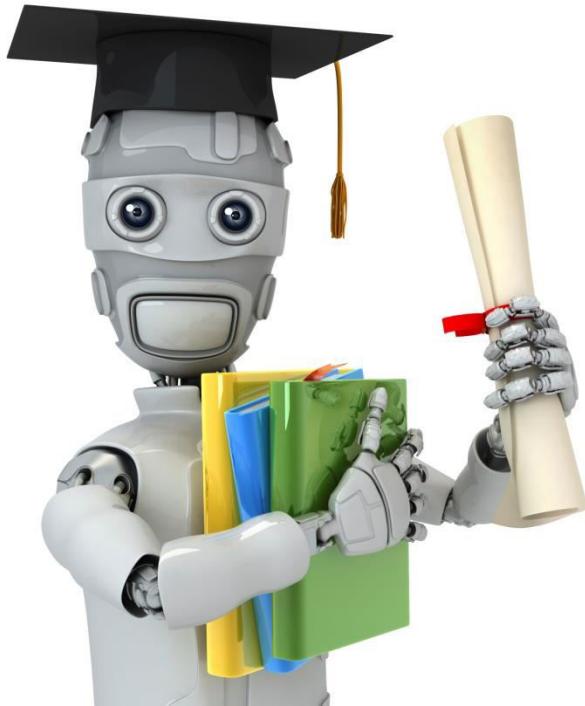
Computationally cheaper

“Large” neural network
(more parameters; more prone
to overfitting)



Computationally more expensive.

Use regularization (λ) to address overfitting.



Machine Learning

Machine learning system design

Prioritizing what to
work on: Spam
classification example

Building a spam classifier

From: cheapsales@buystufffromme.com
To: ang@cs.stanford.edu
Subject: Buy now!

Deal of the week! Buy now!
Rolex w4tchs - \$100
Med1cine (any kind) - \$50
Also low cost M0rgages
available.

From: Alfred Ng
To: ang@cs.stanford.edu
Subject: Christmas dates?

Hey Andrew,
Was talking to Mom about plans
for Xmas. When do you get off
work. Meet Dec 22?
Alf

Building a spam classifier

Supervised learning. x = features of email. y = spam (1) or not spam (0).

Features x : Choose 100 words indicative of spam/not spam.

From: `cheapsales@buystufffromme.com`
To: `ang@cs.stanford.edu`
Subject: Buy now!

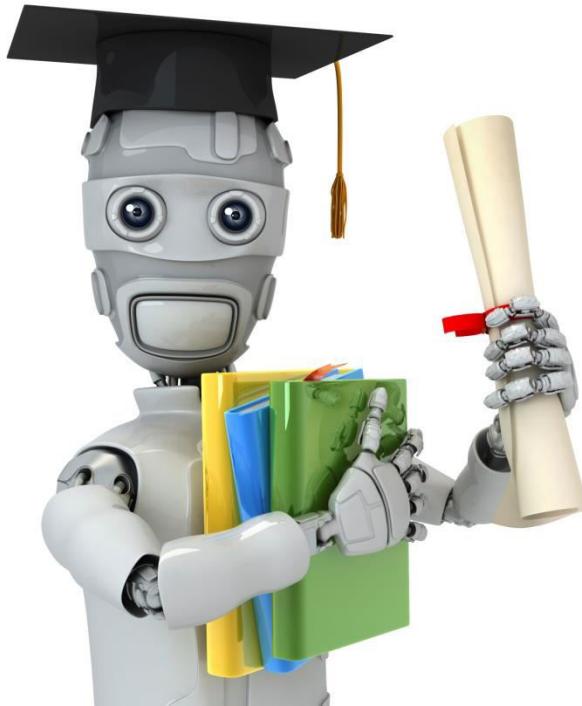
Deal of the week! Buy now!

Note: In practice, take most frequently occurring n words (10,000 to 50,000) in training set, rather than manually pick 100 words.

Building a spam classifier

How to spend your time to make it have low error?

- Collect lots of data
 - E.g. “honeypot” project.
- Develop sophisticated features based on email routing information (from email header).
- Develop sophisticated features for message body, e.g. should “discount” and “discounts” be treated as the same word? How about “deal” and “Dealer”? Features about punctuation?
- Develop sophisticated algorithm to detect misspellings (e.g. m0rtgage, med1cine, w4tches.)



Machine Learning

Machine learning
system design

Error analysis

Recommended approach

- Start with a simple algorithm that you can implement quickly. Implement it and test it on your cross-validation data.
- Plot learning curves to decide if more data, more features, etc. are likely to help.
- Error analysis: Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.

Error Analysis

$m_{CV} = 500$ examples in cross validation set

Algorithm misclassifies 100 emails.

Manually examine the 100 errors, and categorize them based on:

- (i) What type of email it is
- (ii) What cues (features) you think would have helped the algorithm classify them correctly.

Pharma:

Replica/fake:

Steal passwords:

Other:

Deliberate misspellings:

(m0rgage, med1cine, etc.)

Unusual email routing:

Unusual (spamming) punctuation:

The importance of numerical evaluation

Should discount/discounts/discounted/discharging be treated as the same word?

Can use “stemming” software (E.g. “Porter stemmer”)
universe/university.

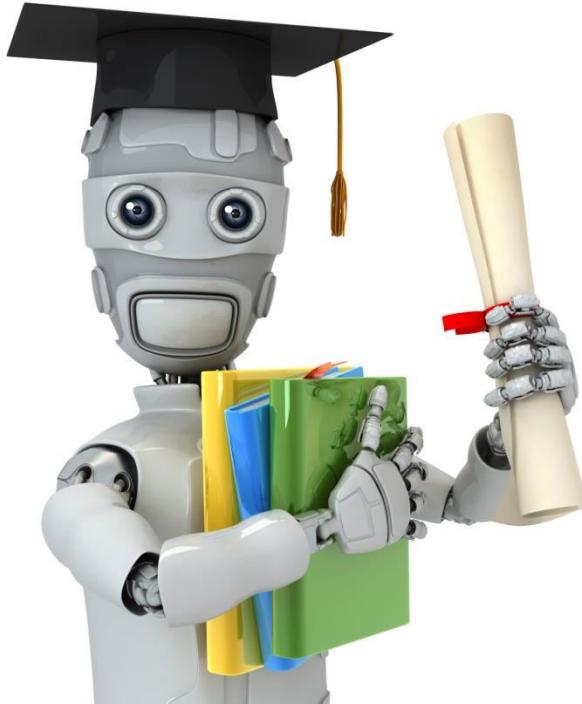
Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try it and see if it works.

Need numerical evaluation (e.g., cross validation error) of algorithm's performance with and without stemming.

Without stemming:

With stemming:

Distinguish upper vs. lower case (Mom/mom):



Machine Learning

Machine learning system design

Error metrics for skewed classes

Cancer classification example

Train logistic regression model $h_{\theta}(x)$. ($y = 1$ if cancer, $y = 0$ otherwise)

Find that you got 1% error on test set.
(99% correct diagnoses)

Only 0.50% of patients have cancer.

```
function y = predictCancer(x)
    y = 0; %ignore x!
return
```

Precision/Recall

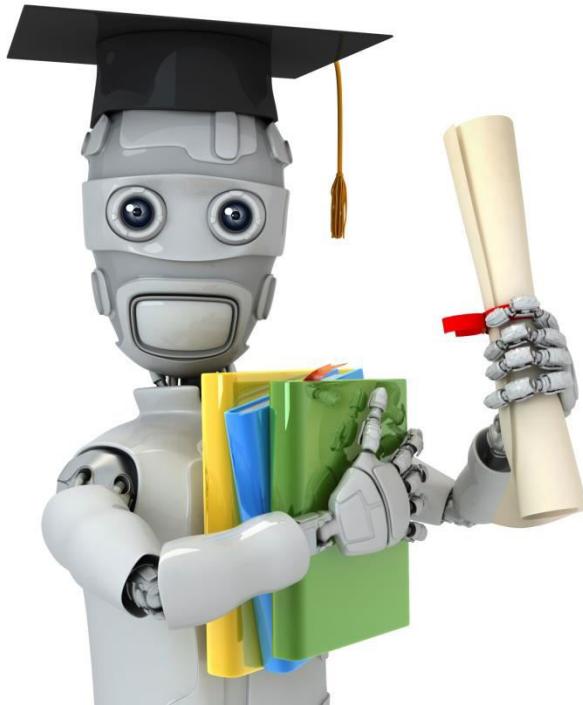
$y = 1$ in presence of rare class that we want to detect

Precision

(Of all patients where we predicted $y = 1$, what fraction actually has cancer?)

Recall

(Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?)



Machine Learning

Machine learning system design

Trading off precision
and recall

Trading off precision and recall

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

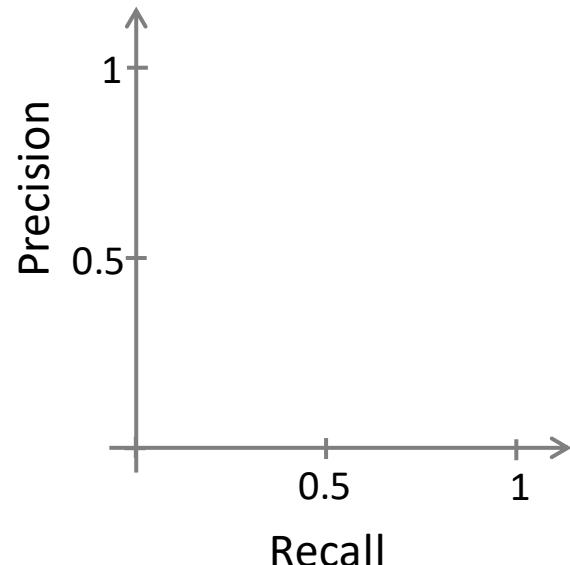
Predict 1 if $h_\theta(x) \geq 0.5$

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

Suppose we want to predict $y = 1$ (cancer)
only if very confident.

Suppose we want to avoid missing too many
cases of cancer (avoid false negatives).

$$\text{precision} = \frac{\text{true positives}}{\text{no. of predicted positive}}$$
$$\text{recall} = \frac{\text{true positives}}{\text{no. of actual positive}}$$



More generally: Predict 1 if $h_\theta(x) \geq \text{threshold}$.

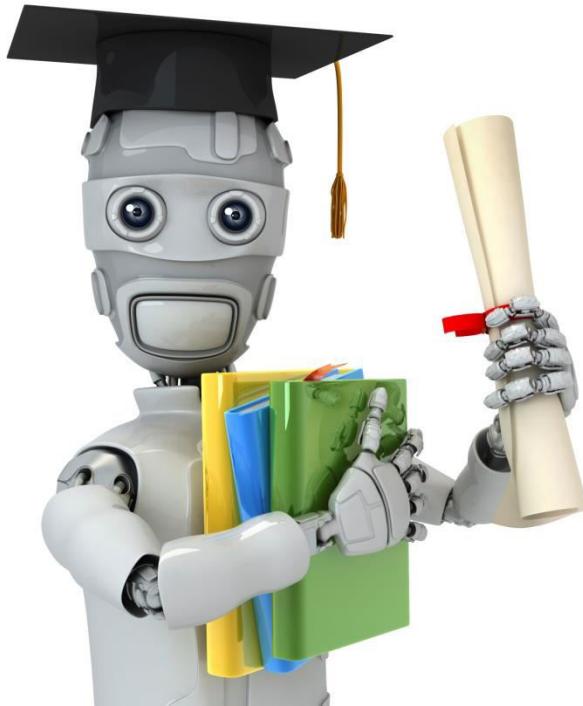
F_1 Score (F score)

How to compare precision/recall numbers?

	Precision(P)	Recall (R)
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1.0

Average: $\frac{P+R}{2}$

F_1 Score: $2 \frac{PR}{P+R}$



Machine Learning

Machine learning system design

Data for machine learning

Designing a high accuracy learning system

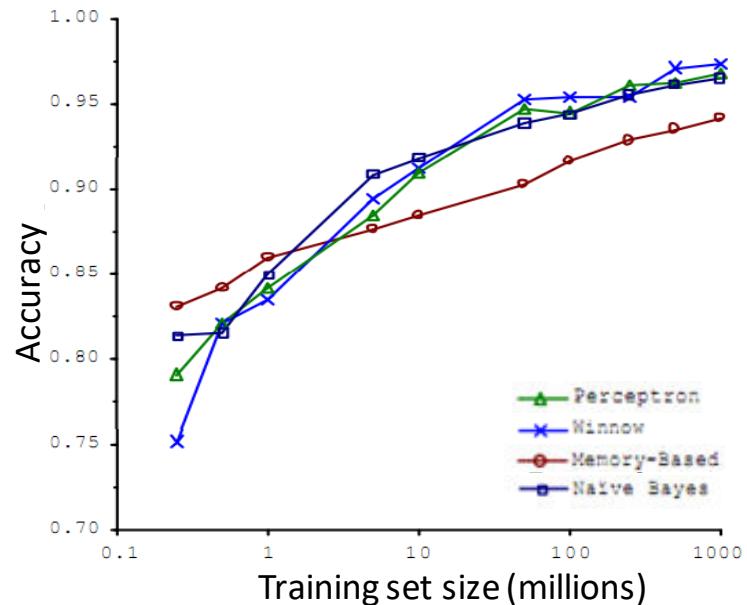
E.g. Classify between confusable words.

{to, two, too}, {then, than}

For breakfast I ate _____ eggs.

Algorithms

- Perceptron (Logistic regression)
- Winnow
- Memory-based
- Naïve Bayes



“It’s not who has the best algorithm that wins.

It’s who has the most data.”

Large data rationale

Assume feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately.

Example: For breakfast I ate _____ eggs.

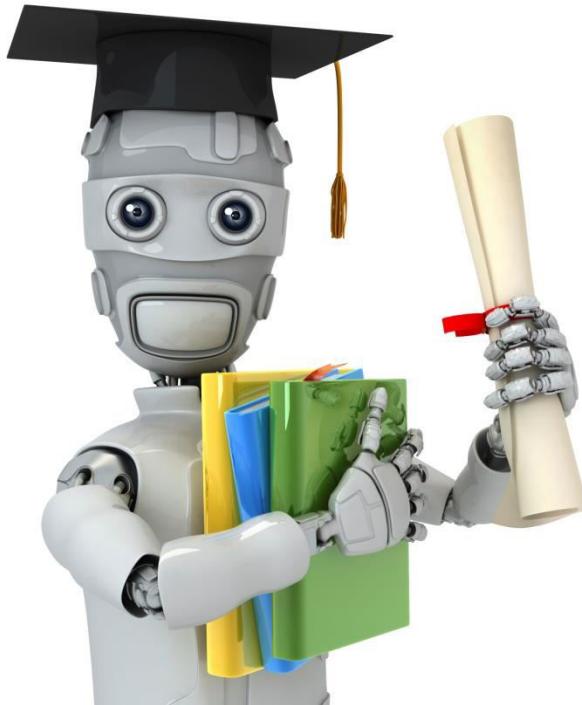
Counterexample: Predict housing price from only size (feet²) and no other features.

Useful test: Given the input x , can a human expert confidently predict y ?

Large data rationale

Use a learning algorithm with many parameters (e.g. logistic regression/linear regression with many features; neural network with many hidden units).

Use a very large training set (unlikely to overfit)



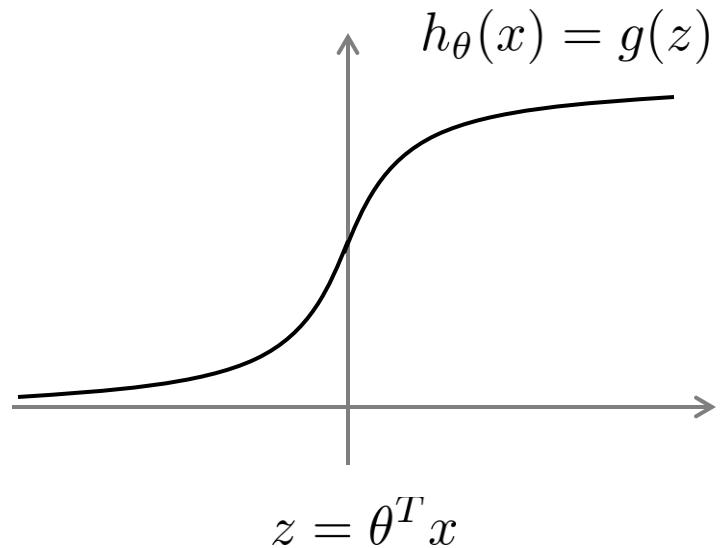
Machine Learning

Support Vector Machines

Optimization objective

Alternative view of logistic regression

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



If $y = 1$, we want $h_{\theta}(x) \approx 1$, $\theta^T x \gg 0$

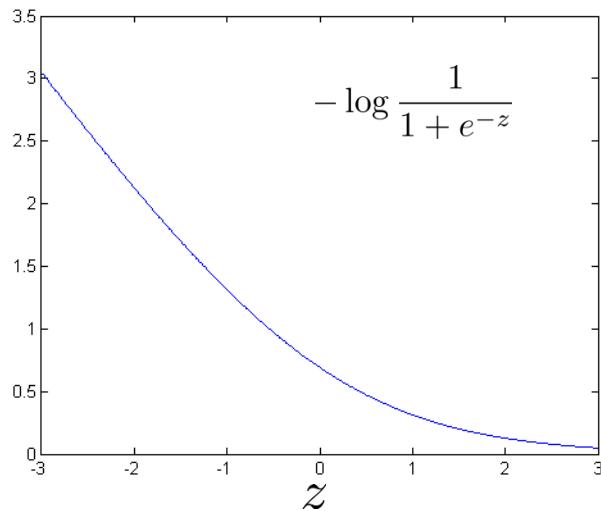
If $y = 0$, we want $h_{\theta}(x) \approx 0$, $\theta^T x \ll 0$

Alternative view of logistic regression

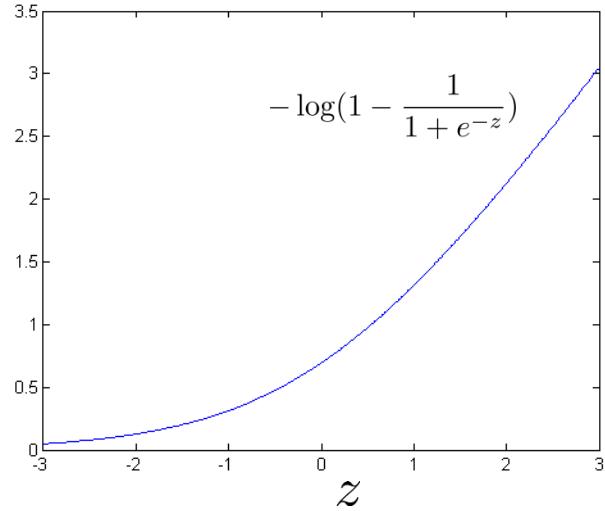
Cost of example: $-(y \log h_\theta(x) + (1 - y) \log(1 - h_\theta(x)))$

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

If $y = 1$ (want $\theta^T x \gg 0$):



If $y = 0$ (want $\theta^T x \ll 0$):



Support vector machine

Logistic regression:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \left(-\log h_{\theta}(x^{(i)}) \right) + (1 - y^{(i)}) \left((-\log(1 - h_{\theta}(x^{(i)}))) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

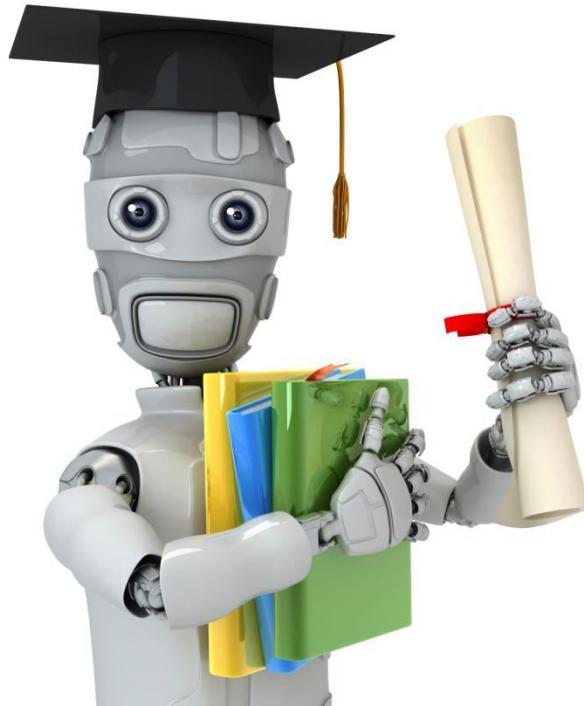
Support vector machine:

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^n \theta_j^2$$

SVM hypothesis

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

Hypothesis:



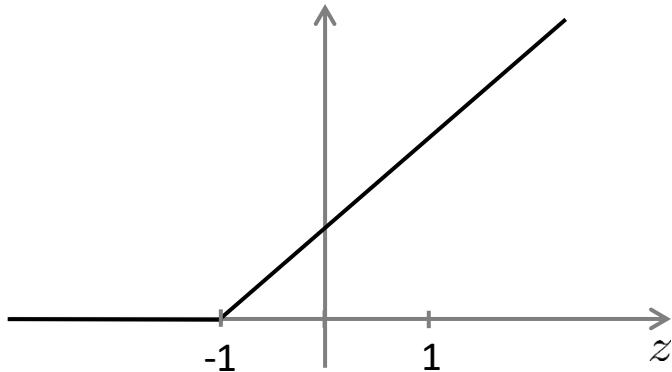
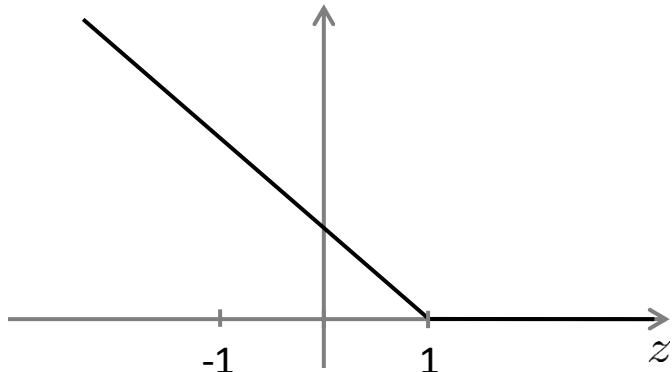
Machine Learning

Support Vector Machines

Large Margin Intuition

Support Vector Machine

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$



If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)

If $y = 0$, we want $\theta^T x \leq -1$ (not just < 0)

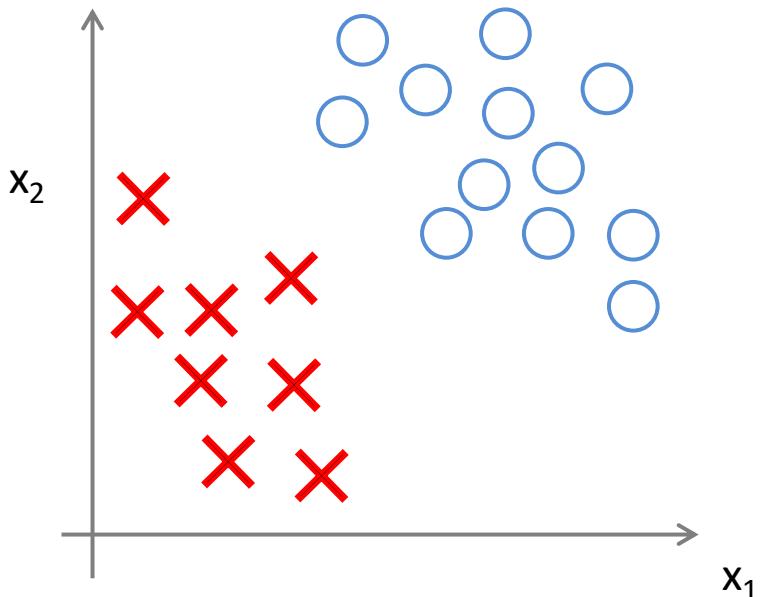
SVM Decision Boundary

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^n \theta_j^2$$

Whenever $y^{(i)} = 1$:

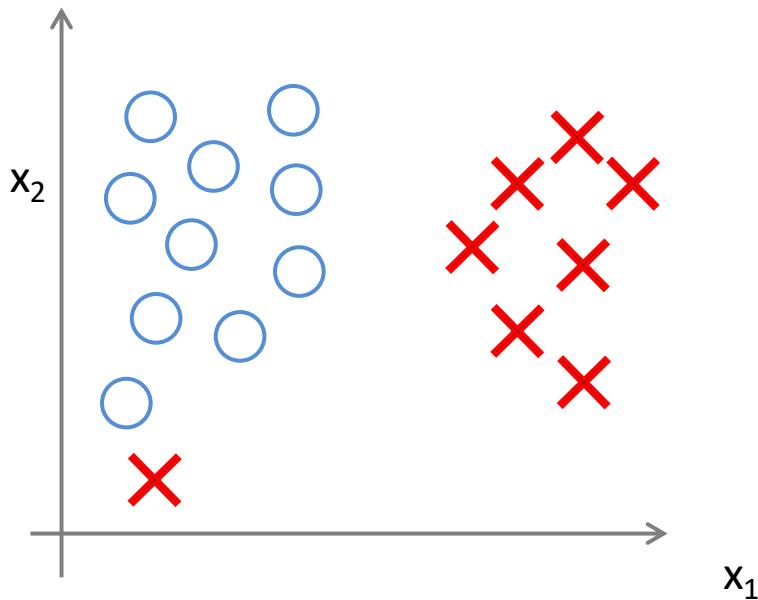
Whenever $y^{(i)} = 0$:

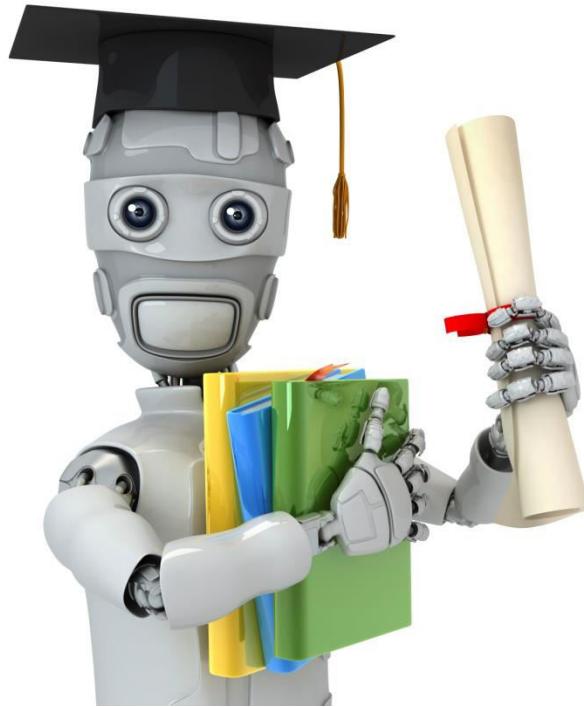
SVM Decision Boundary: Linearly separable case



Large margin classifier

Large margin classifier in presence of outliers





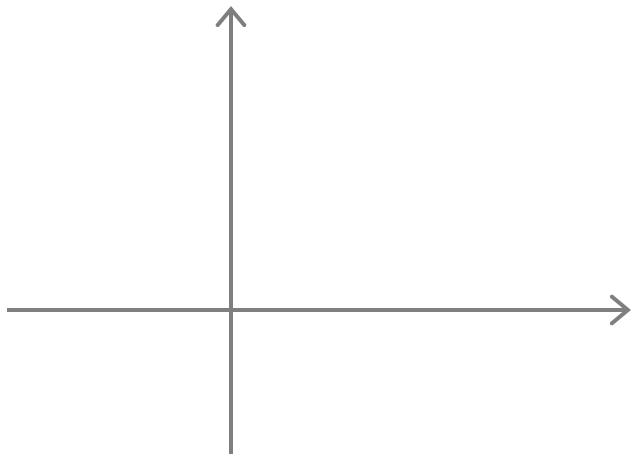
Machine Learning

Support Vector Machines

The mathematics
behind large margin
classification (optional)

Vector Inner Product

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

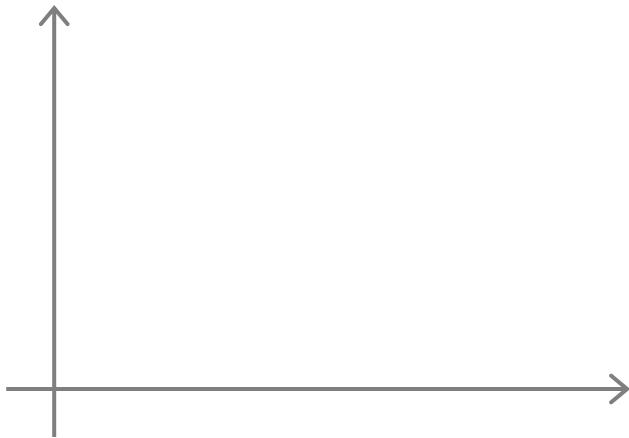


SVM Decision Boundary

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

$$\text{s.t. } \theta^T x^{(i)} \geq 1 \quad \text{if } y^{(i)} = 1$$

$$\theta^T x^{(i)} \leq -1 \quad \text{if } y^{(i)} = 0$$



SVM Decision Boundary

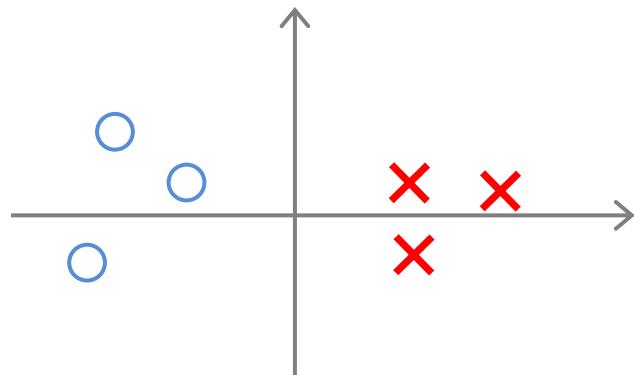
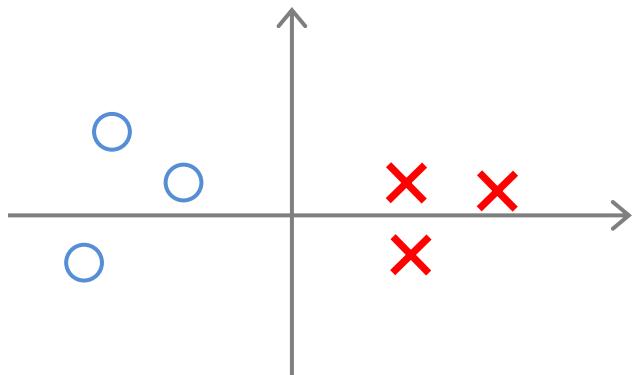
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

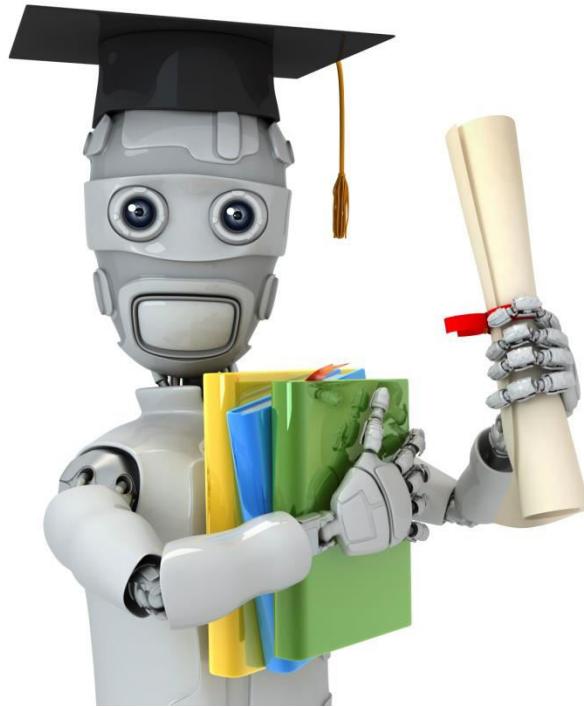
$$\text{s.t. } p^{(i)} \cdot \|\theta\| \geq 1 \quad \text{if } y^{(i)} = 1$$

$$p^{(i)} \cdot \|\theta\| \leq -1 \quad \text{if } y^{(i)} = -1$$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ .

Simplification: $\theta_0 = 0$



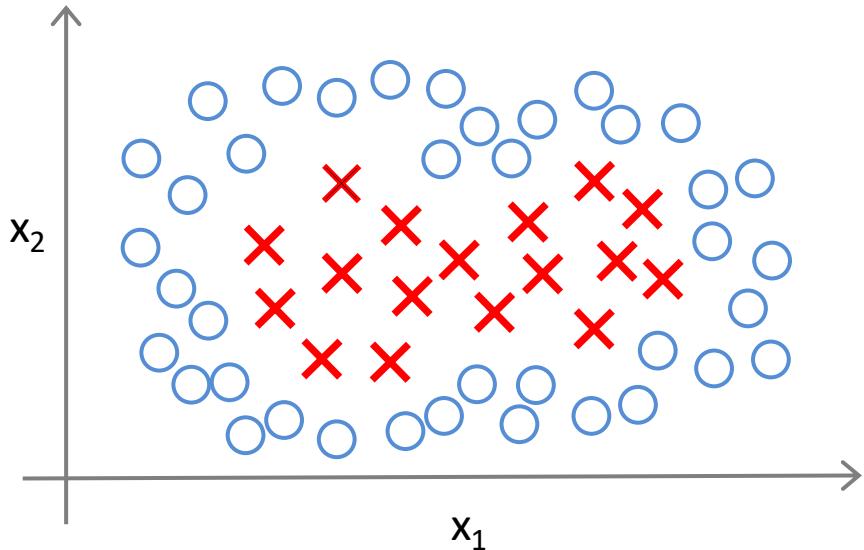


Machine Learning

Support Vector Machines

Kernels I

Non-linear Decision Boundary

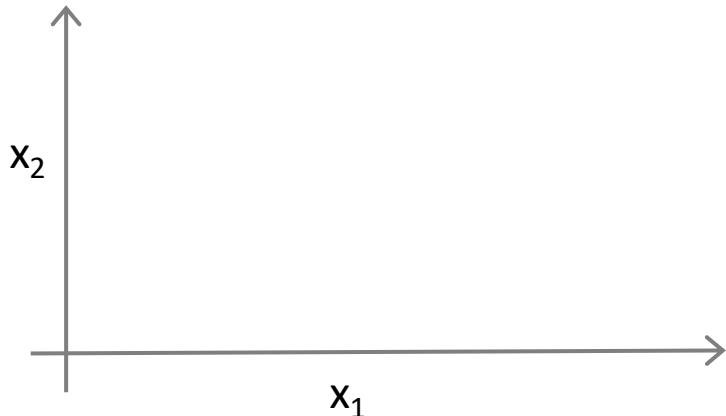


Predict $y = 1$ if

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \dots \geq 0$$

Is there a different / better choice of the features f_1, f_2, f_3, \dots ?

Kernel



Given x , compute new feature depending
on proximity to landmarks $l^{(1)}, l^{(2)}, l^{(3)}$

Kernels and Similarity

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

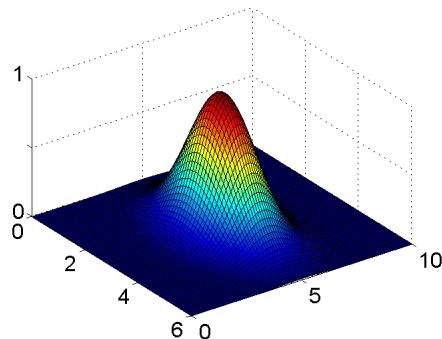
If $x \approx l^{(1)}$:

If x if far from $l^{(1)}$:

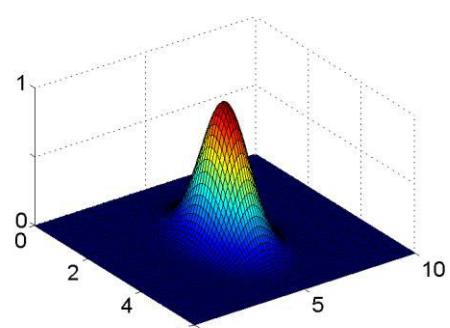
Example:

$$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2}\right)$$

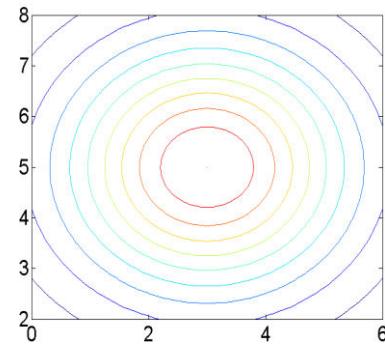
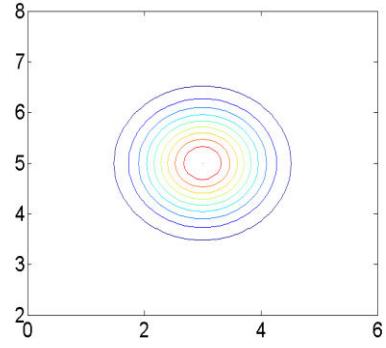
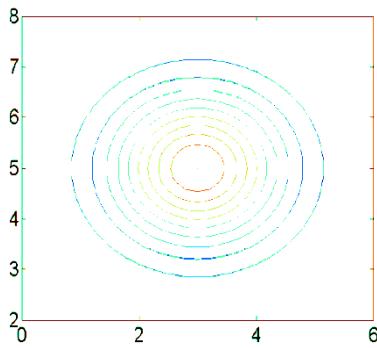
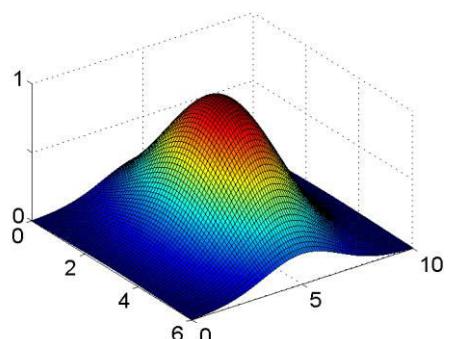
$$\sigma^2 = 1$$

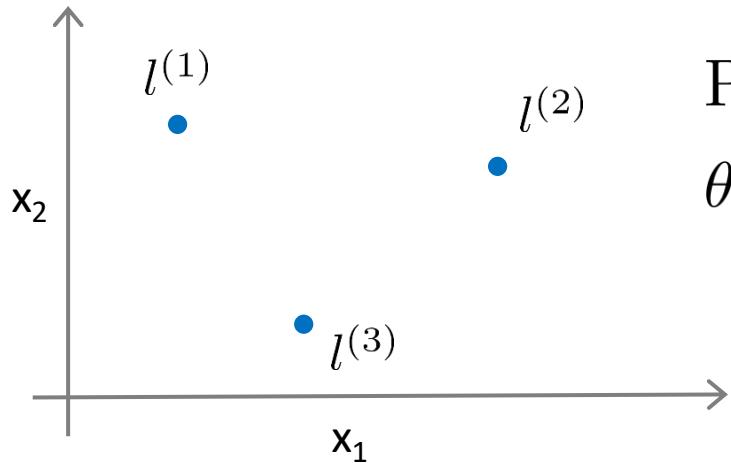


$$\sigma^2 = 0.5$$

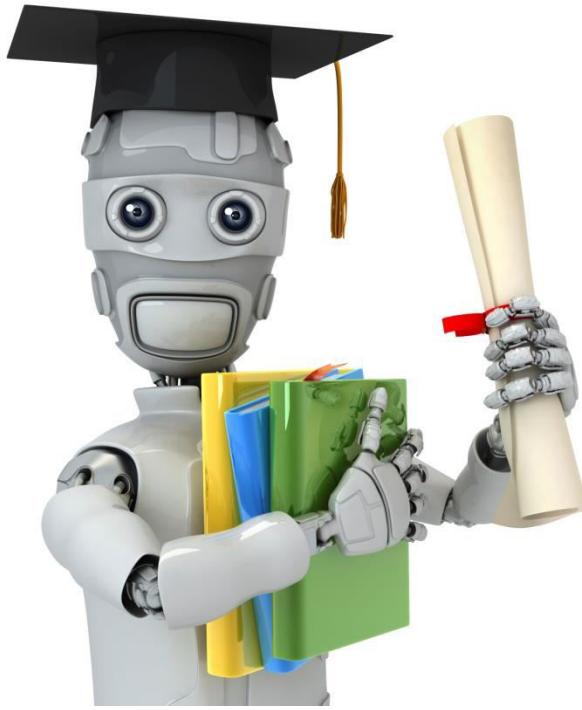


$$\sigma^2 = 3$$





Predict “1” when
 $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

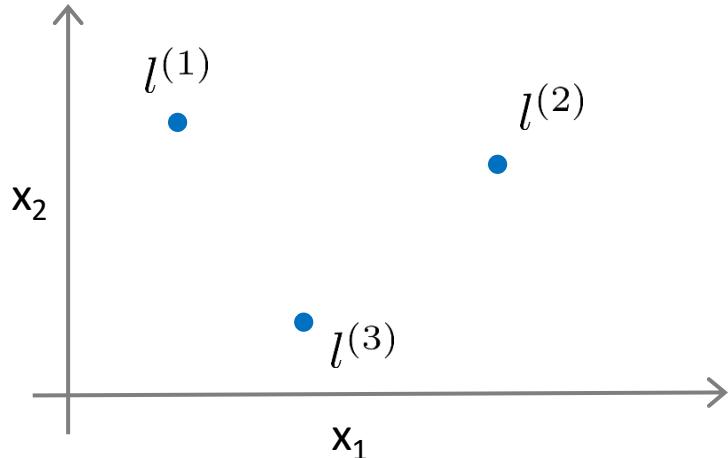


Machine Learning

Support Vector Machines

Kernels II

Choosing the landmarks

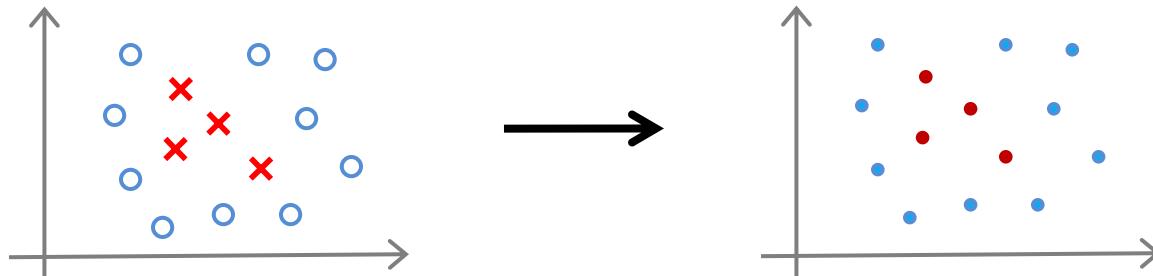


Given x :

$$\begin{aligned}f_i &= \text{similarity}(x, l^{(i)}) \\&= \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)\end{aligned}$$

Predict $y = 1$ if $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

Where to get $l^{(1)}, l^{(2)}, l^{(3)}, \dots$?



SVM with Kernels

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$,
choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$.

Given example x :

$$\begin{aligned}f_1 &= \text{similarity}(x, l^{(1)}) \\f_2 &= \text{similarity}(x, l^{(2)})\end{aligned}$$

...

For training example $(x^{(i)}, y^{(i)})$:

SVM with Kernels

Hypothesis: Given x , compute features $f \in \mathbb{R}^{m+1}$

Predict “y=1” if $\theta^T f \geq 0$

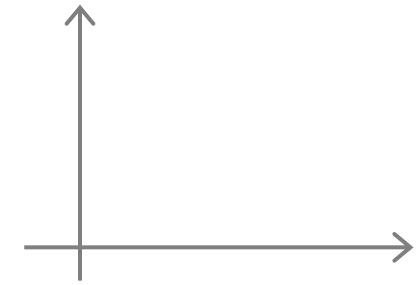
Training:

$$\min_{\theta} C \sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

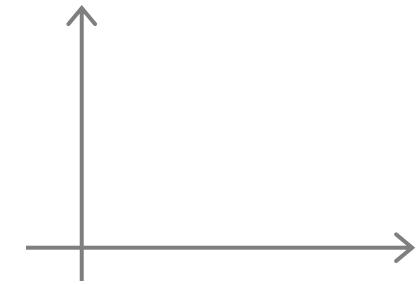
SVM parameters:

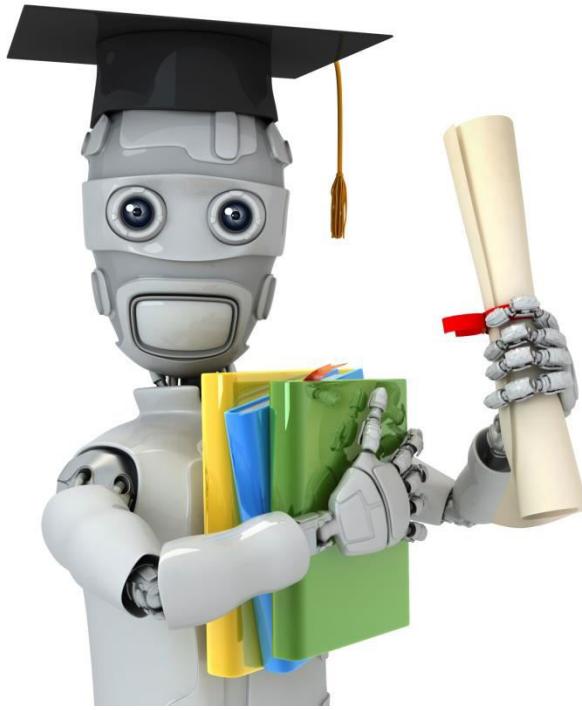
C ($= \frac{1}{\lambda}$). Large C : Lower bias, high variance.
Small C : Higher bias, low variance.

σ^2 Large σ^2 : Features f_i vary more smoothly.
Higher bias, lower variance.



Small σ^2 : Features f_i vary less smoothly.
Lower bias, higher variance.





Machine Learning

Support Vector Machines

Using an SVM

Use SVM software package (e.g. liblinear, libsvm, ...) to solve for parameters θ .

Need to specify:

Choice of parameter C.

Choice of kernel (similarity function):

E.g. No kernel (“linear kernel”)

Predict “y = 1” if $\theta^T x \geq 0$

Gaussian kernel:

$$f_i = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right), \text{ where } l^{(i)} = x^{(i)}.$$

Need to choose σ^2 .

Kernel (similarity) functions:

```
function f = kernel(x1,x2)
```

$$f = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}\right)$$

```
return
```

Note: Do perform feature scaling before using the Gaussian kernel.

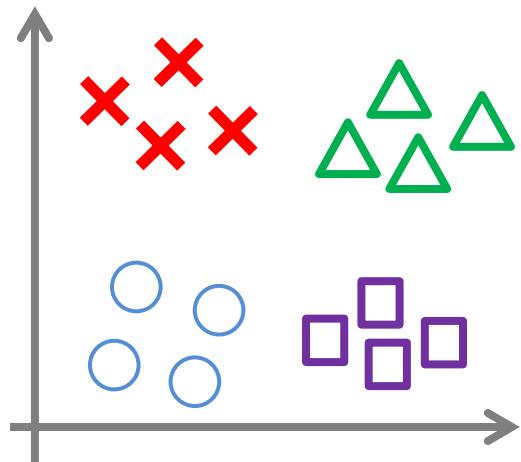
Other choices of kernel

Note: Not all similarity functions $\text{similarity}(x, l)$ make valid kernels.
(Need to satisfy technical condition called “Mercer’s Theorem” to make sure SVM packages’ optimizations run correctly, and do not diverge).

Many off-the-shelf kernels available:

- Polynomial kernel:
- More esoteric: String kernel, chi-square kernel, histogram intersection kernel, ...

Multi-class classification



$$y \in \{1, 2, 3, \dots, K\}$$

Many SVM packages already have built-in multi-class classification functionality.

Otherwise, use one-vs.-all method. (Train K SVMs, one to distinguish $y = i$ from the rest, for $i = 1, 2, \dots, K$), get $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)}$. Pick class i with largest $(\theta^{(i)})^T x$

Logistic regression vs. SVMs

n = number of features ($x \in \mathbb{R}^{n+1}$), m = number of training examples

If n is large (relative to m):

Use logistic regression, or SVM without a kernel (“linear kernel”)

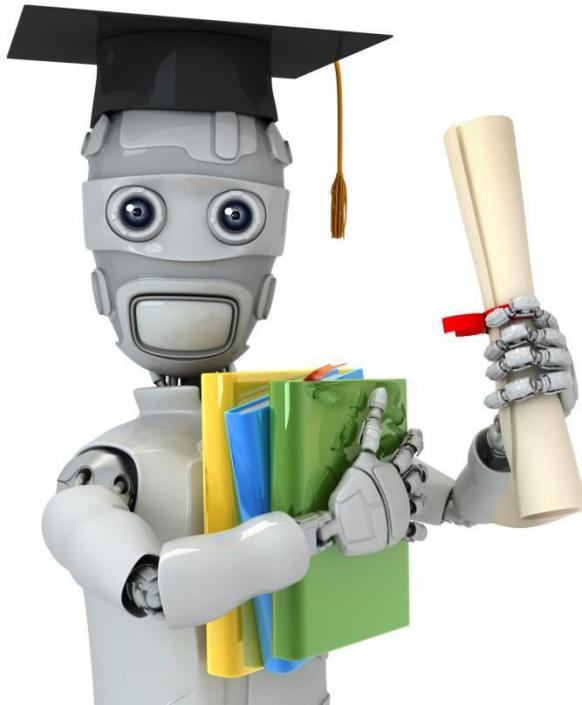
If n is small, m is intermediate:

Use SVM with Gaussian kernel

If n is small, m is large:

Create/add more features, then use logistic regression or SVM without a kernel

Neural network likely to work well for most of these settings, but may be slower to train.

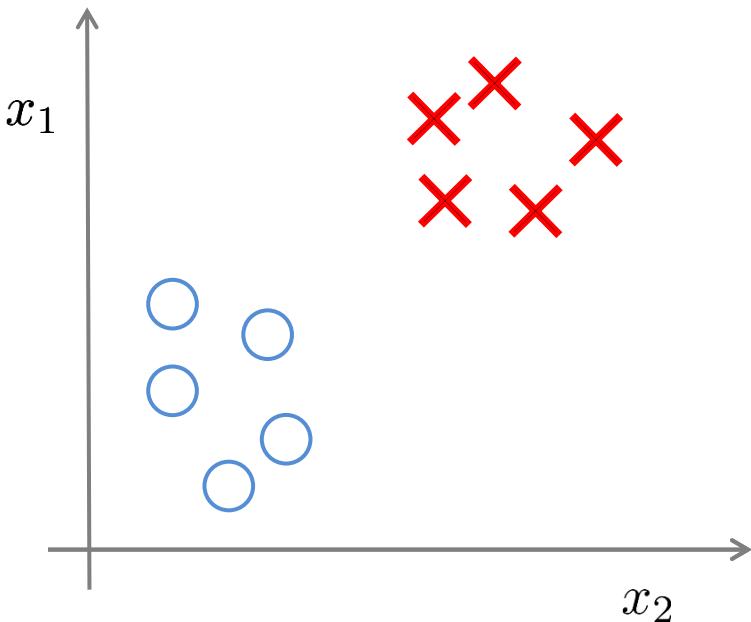


Machine Learning

Clustering

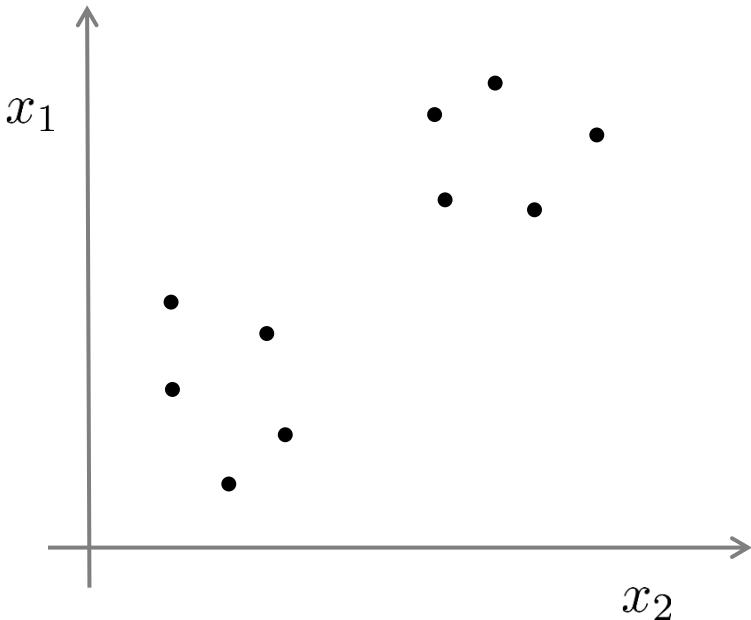
Unsupervised learning introduction

Supervised learning



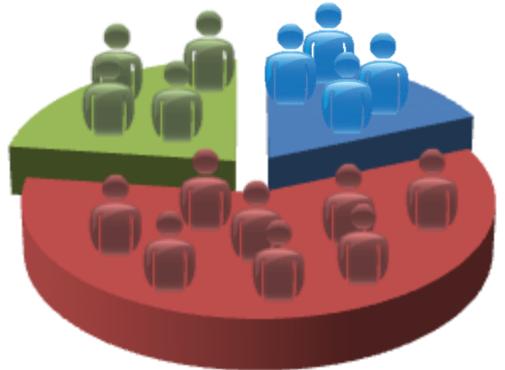
Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots, (x^{(m)}, y^{(m)})\}$

Unsupervised learning

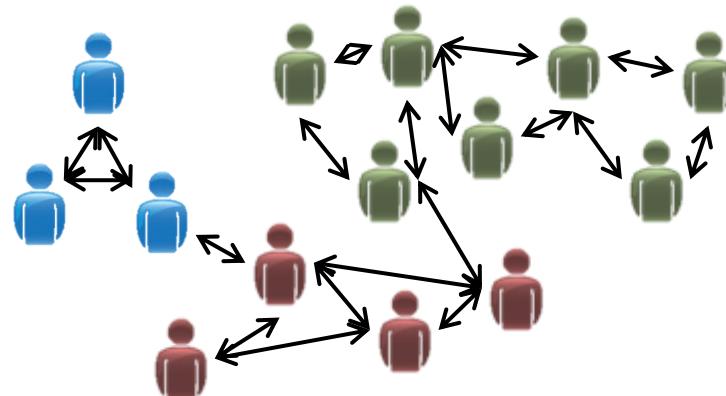


Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(m)}\}$

Applications of clustering



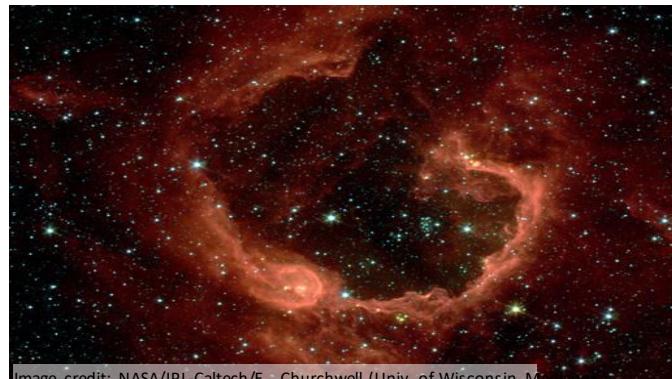
Market segmentation



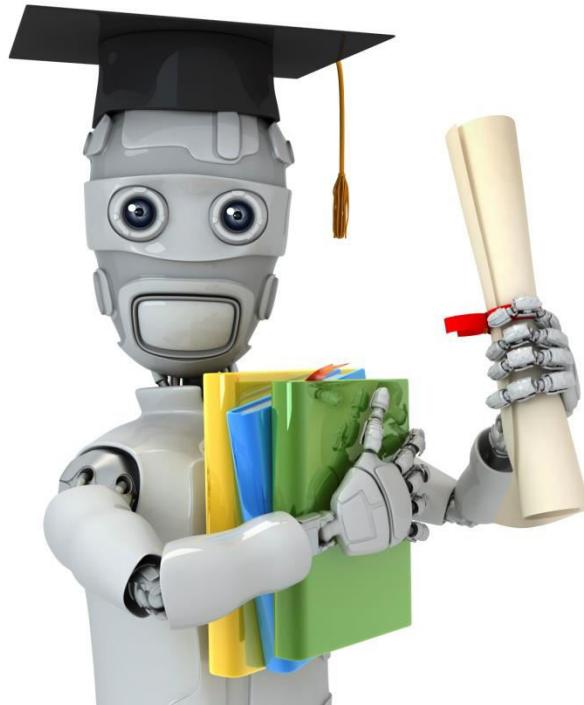
Social network analysis



Organize computing clusters



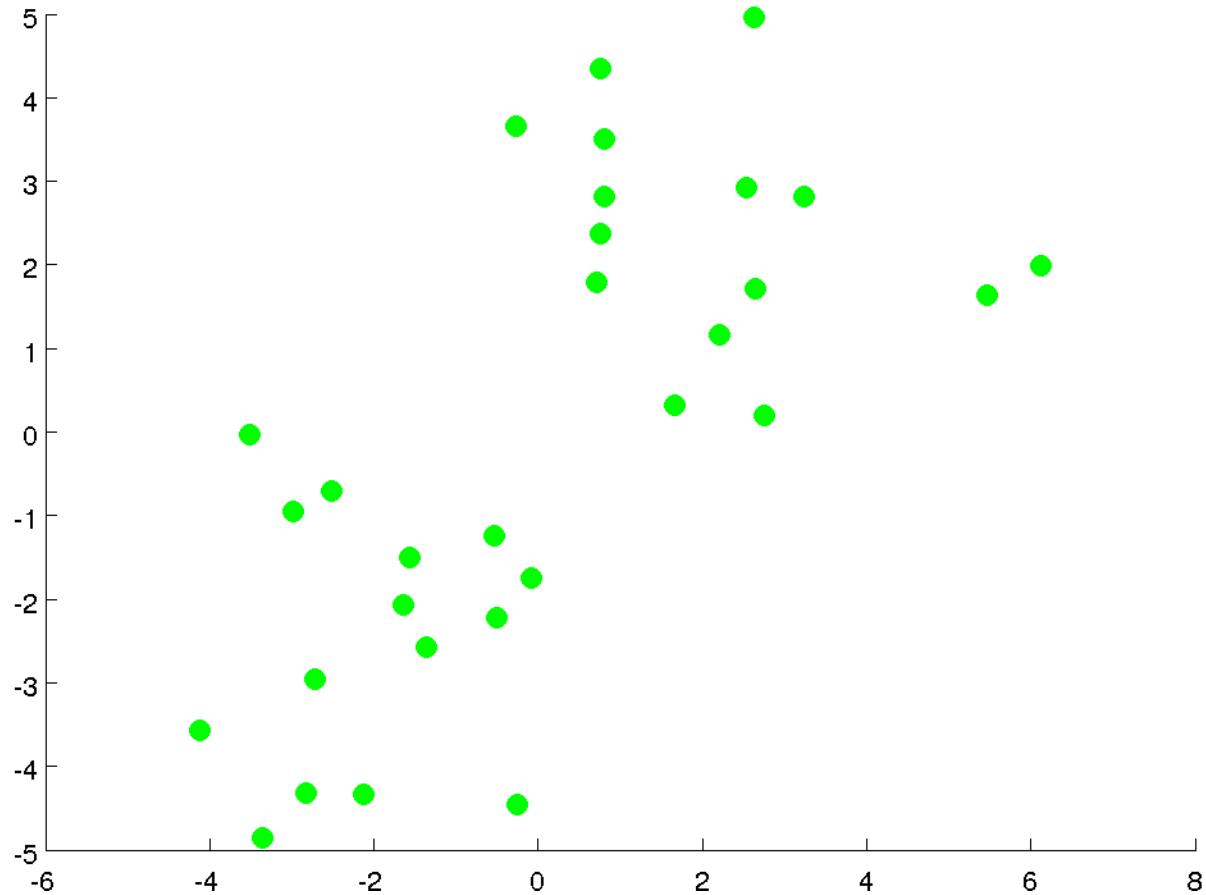
Astronomical data analysis

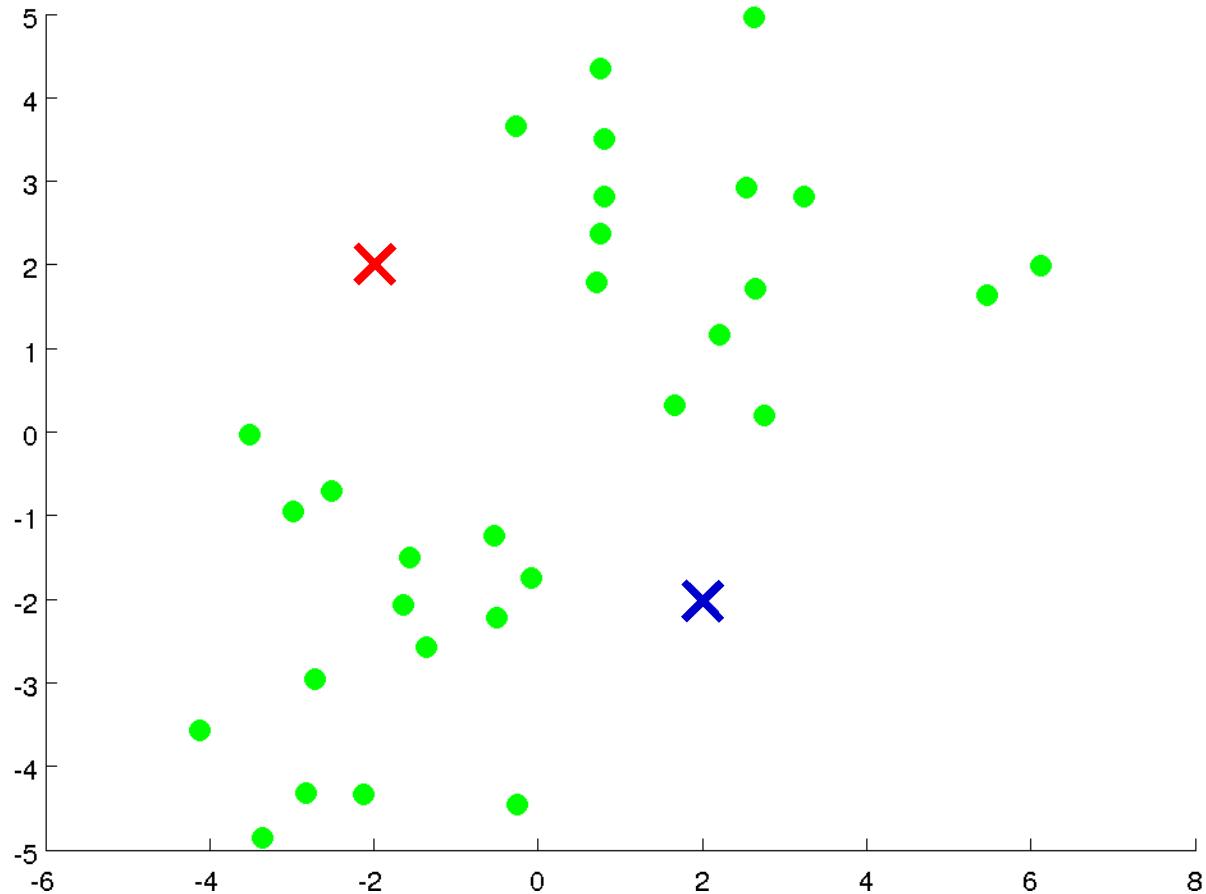


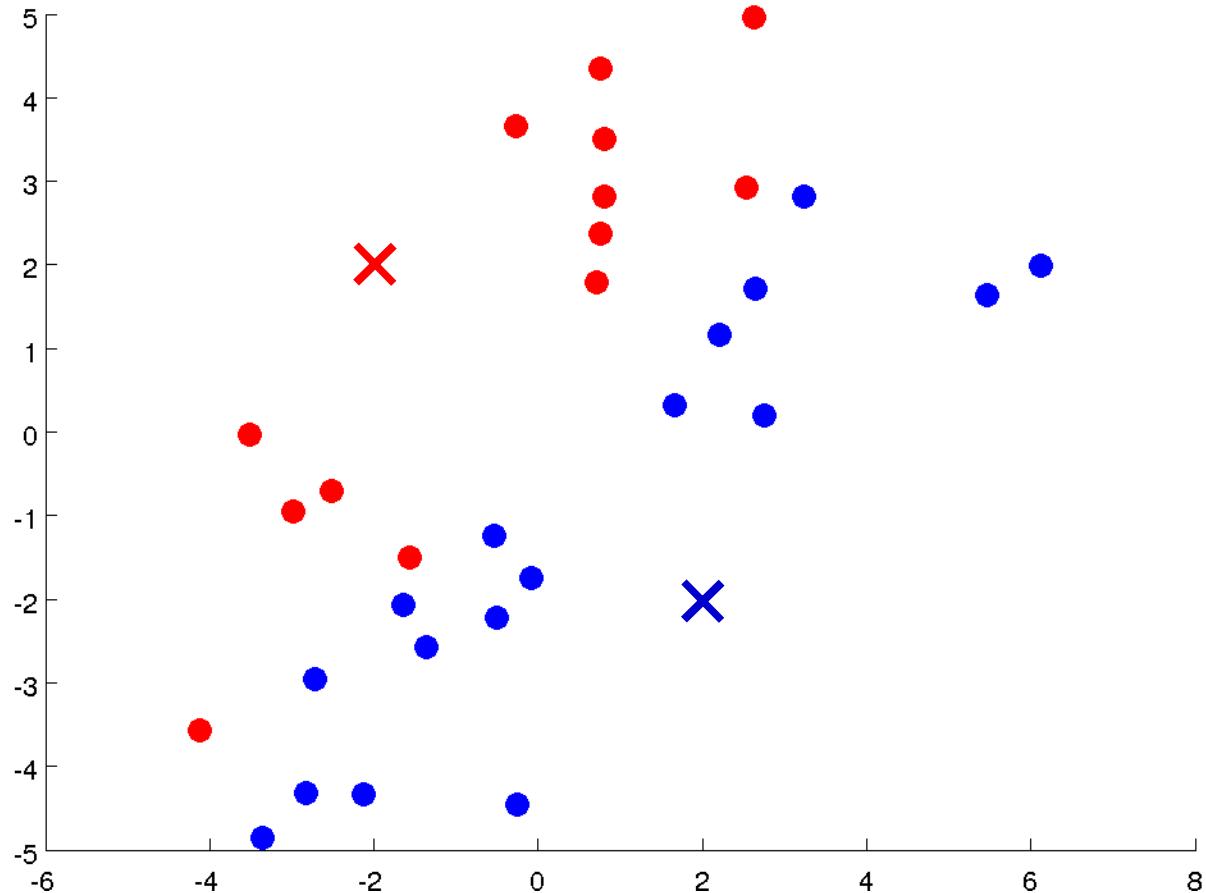
Machine Learning

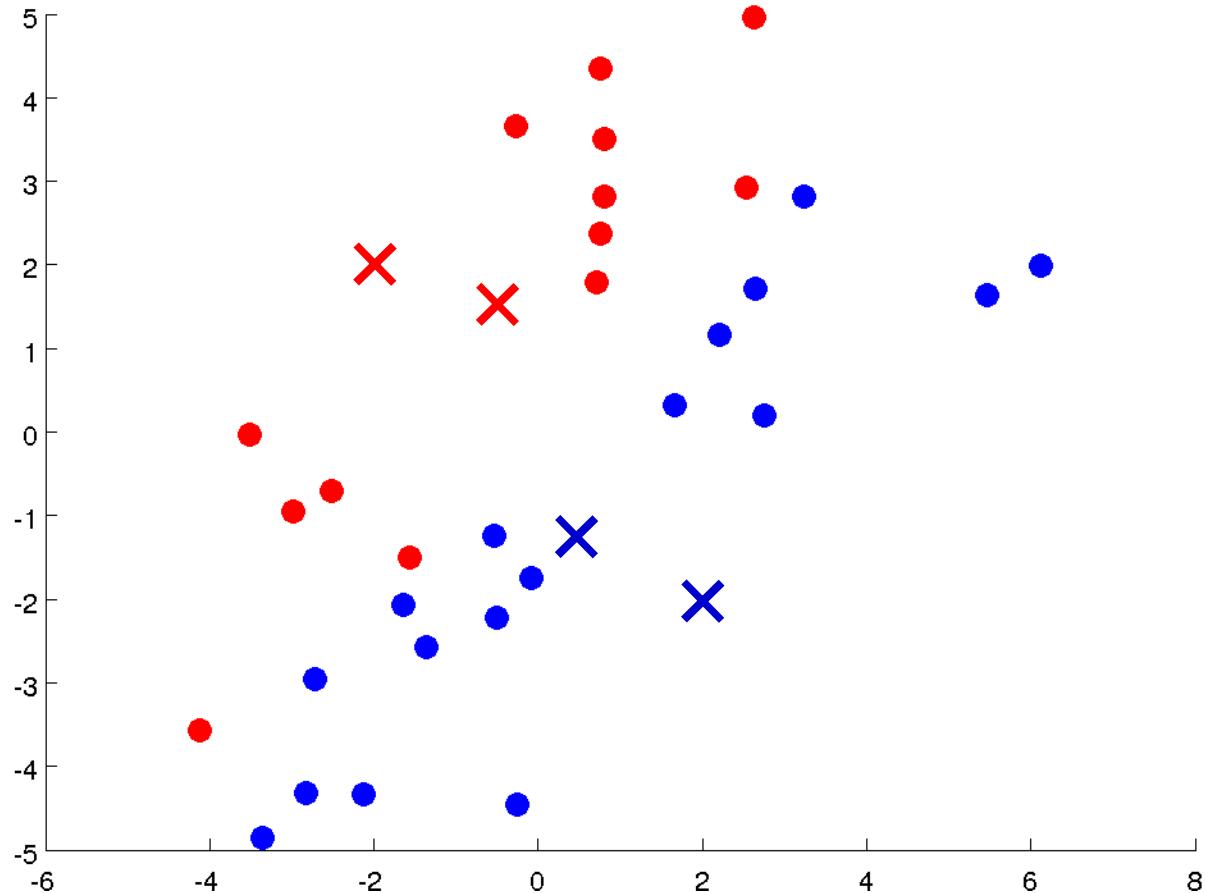
Clustering

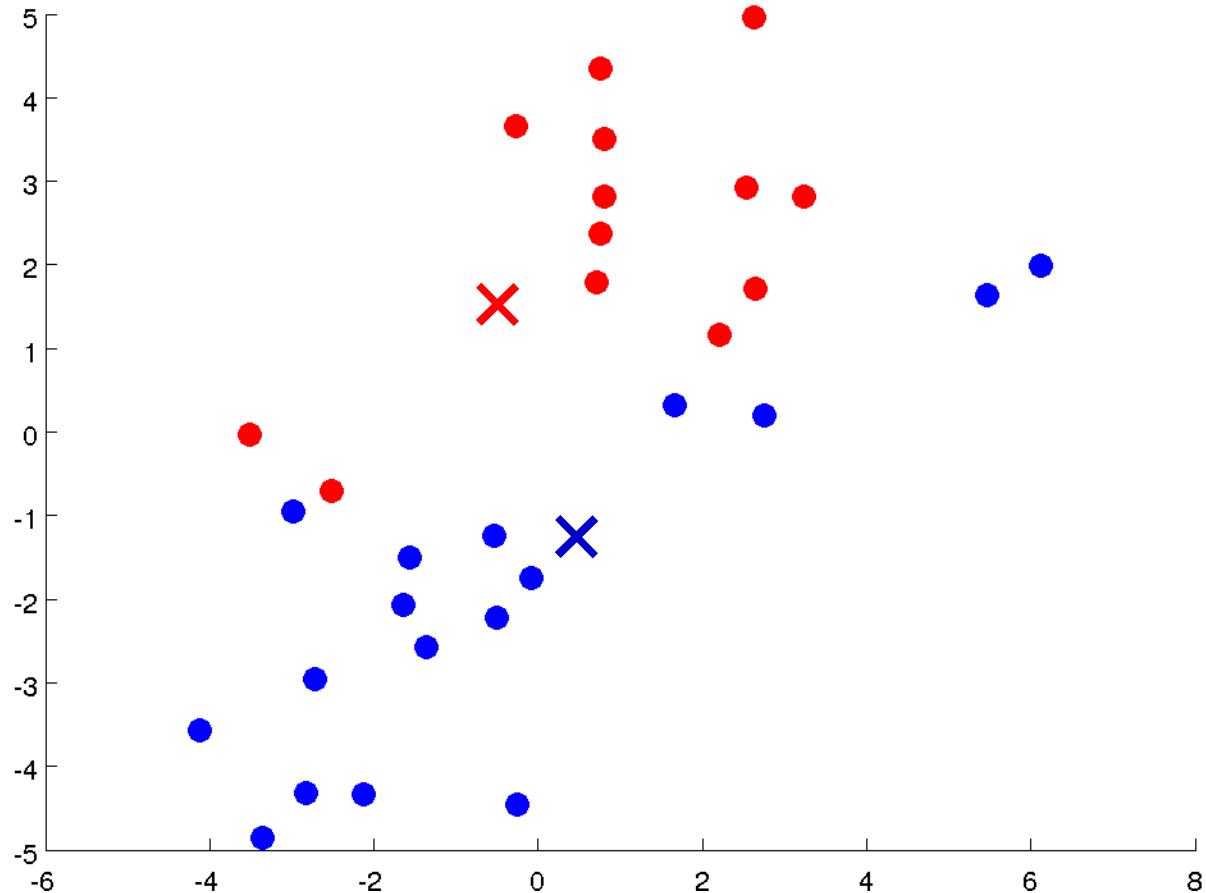
K-means algorithm

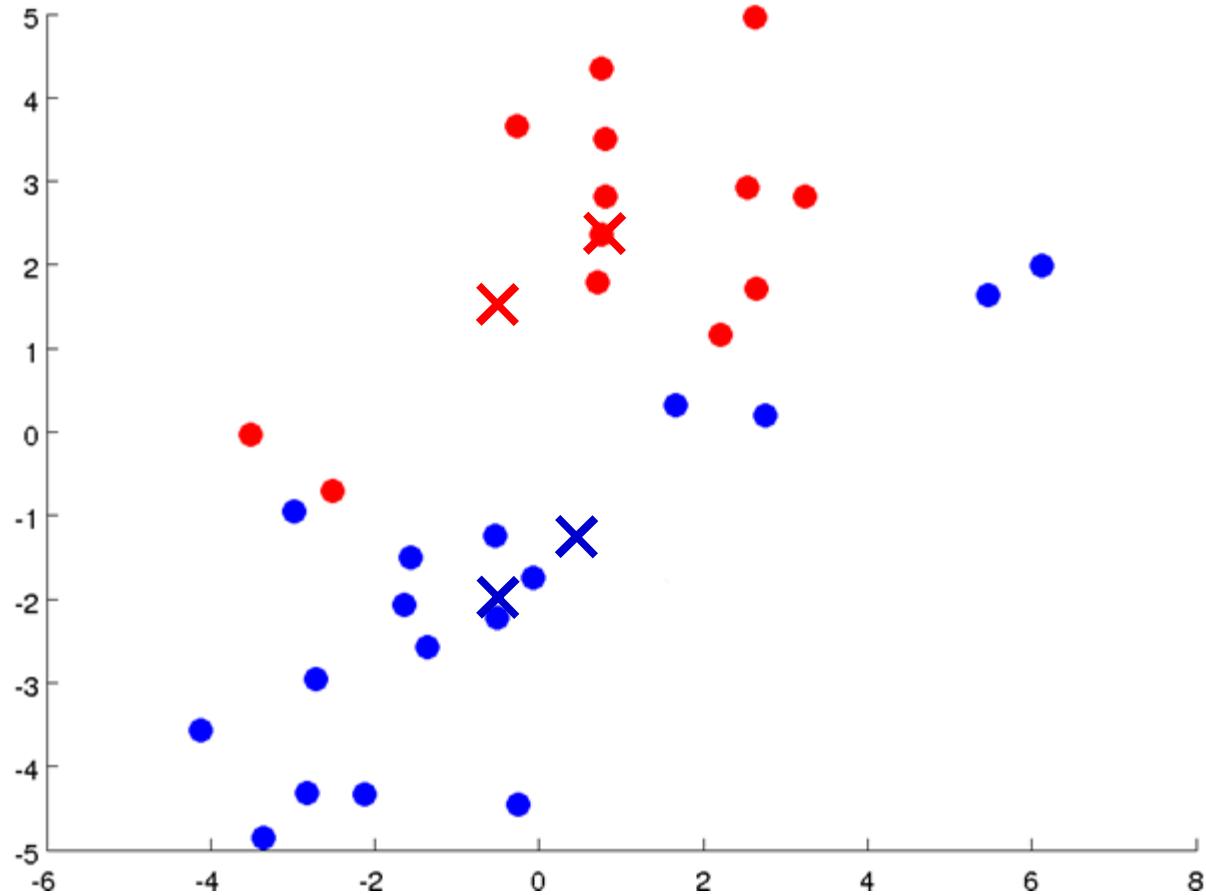


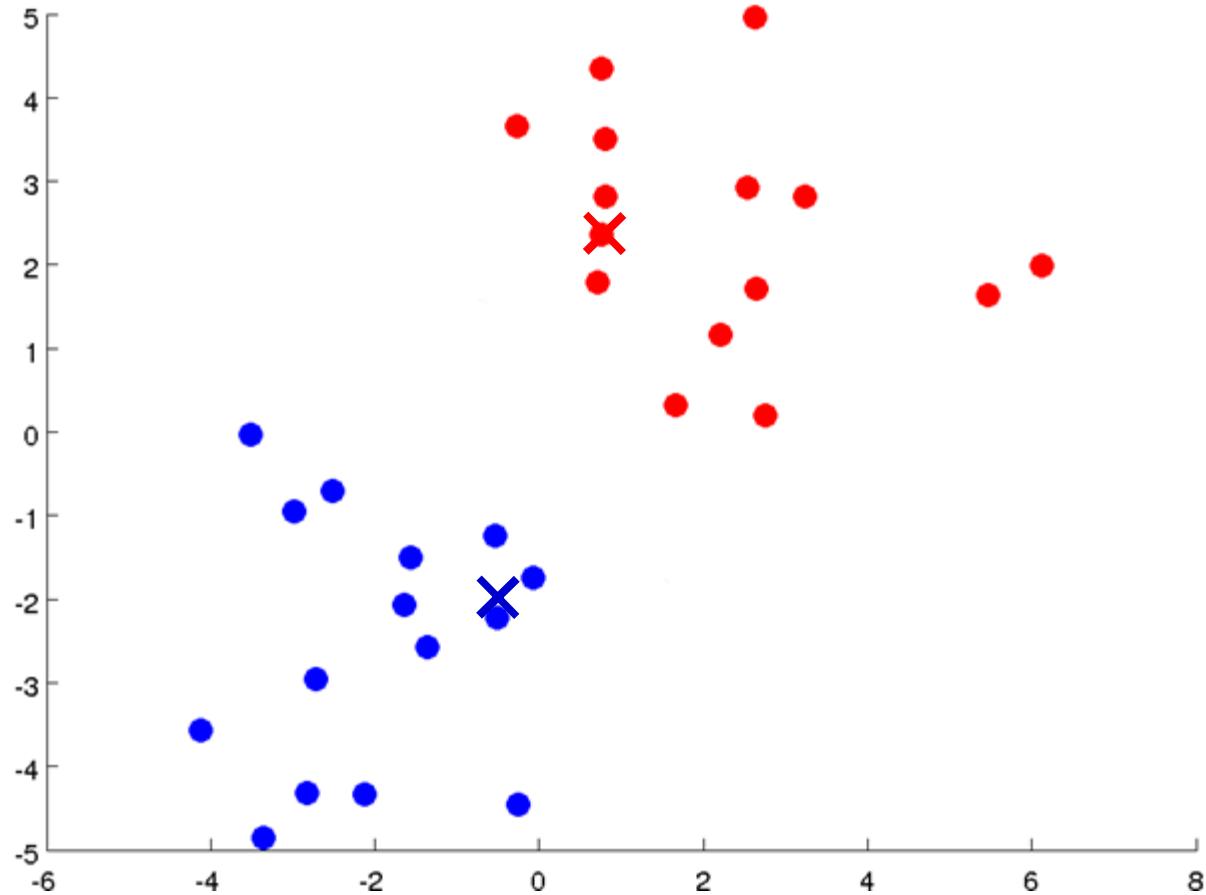


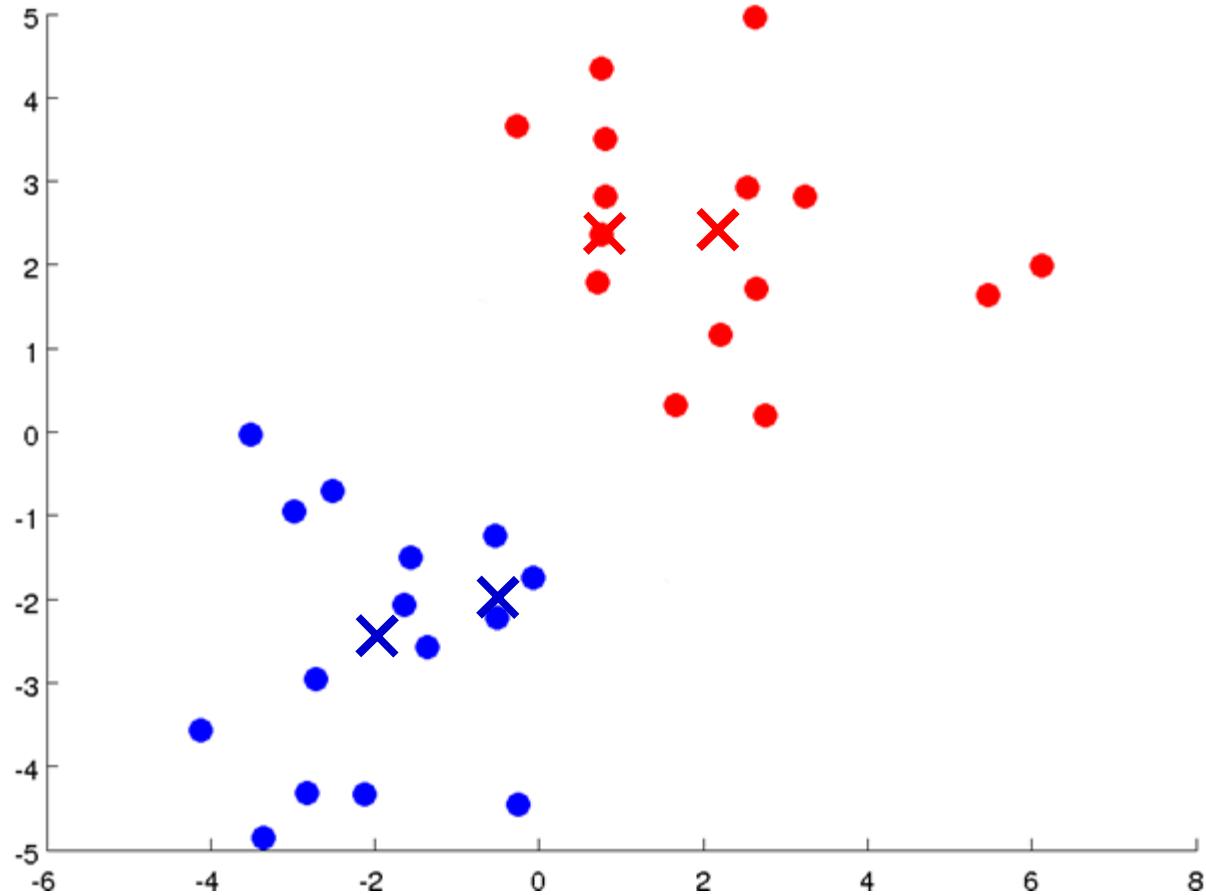


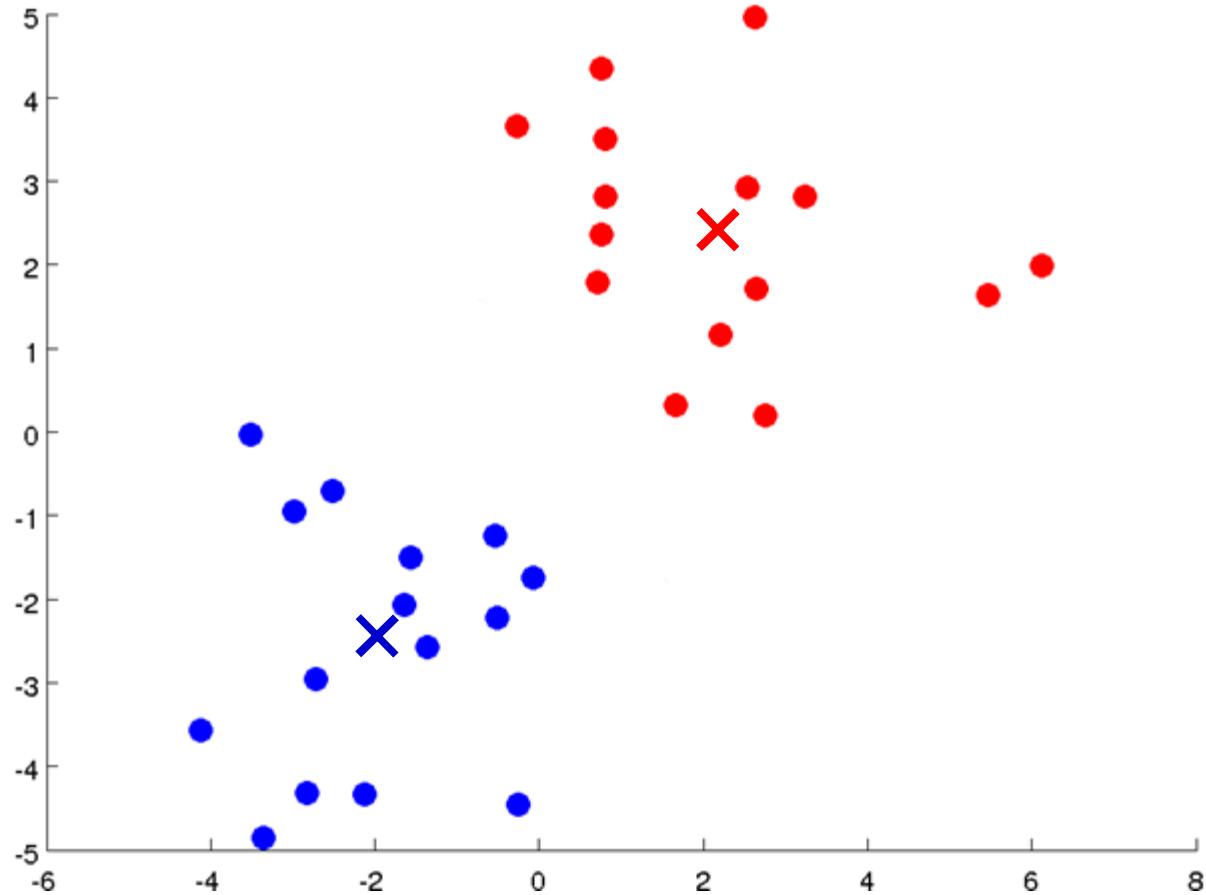












K-means algorithm

Input:

- K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

 for $i = 1$ to m

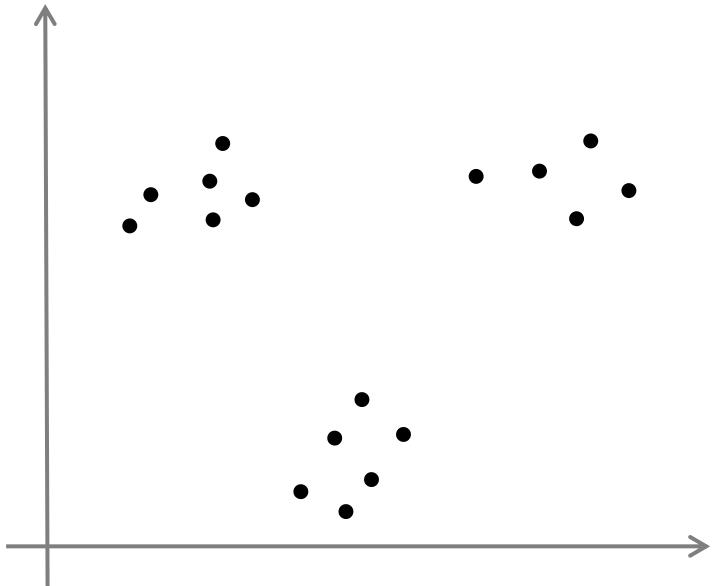
$c^{(i)} :=$ index (from 1 to K) of cluster centroid
 closest to $x^{(i)}$

 for $k = 1$ to K

$\mu_k :=$ average (mean) of points assigned to cluster k

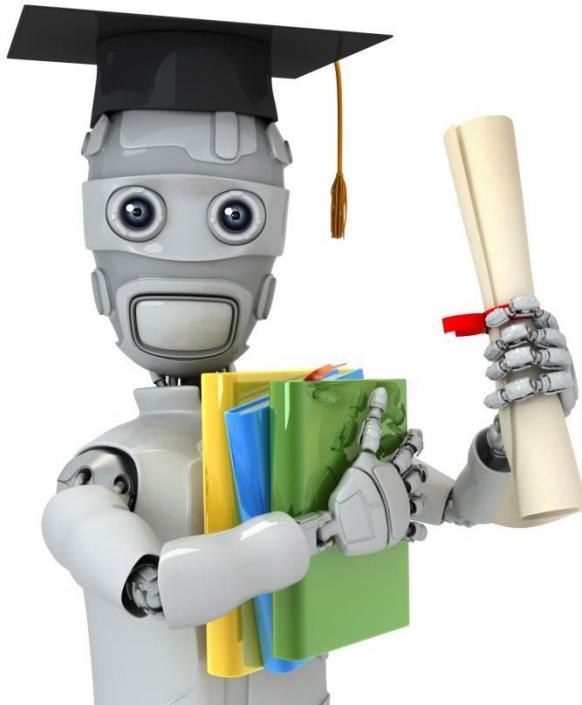
}

K-means for non-separated clusters



T-shirt sizing





Machine Learning

Clustering Optimization objective

K-means optimization objective

$c^{(i)}$ = index of cluster ($1, 2, \dots, K$) to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Optimization objective:

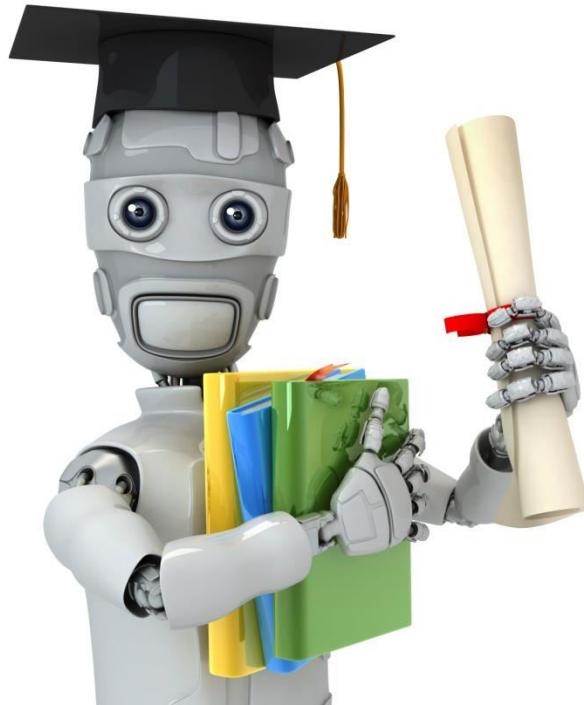
$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

$$\min_{\substack{c^{(1)}, \dots, c^{(m)}, \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

```
Repeat {  
    for  $i = 1$  to  $m$   
         $c^{(i)} :=$  index (from 1 to  $K$ ) of cluster centroid  
        closest to  $x^{(i)}$   
    for  $k = 1$  to  $K$   
         $\mu_k :=$  average (mean) of points assigned to cluster  $k$   
}
```



Machine Learning

Clustering

Random initialization

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

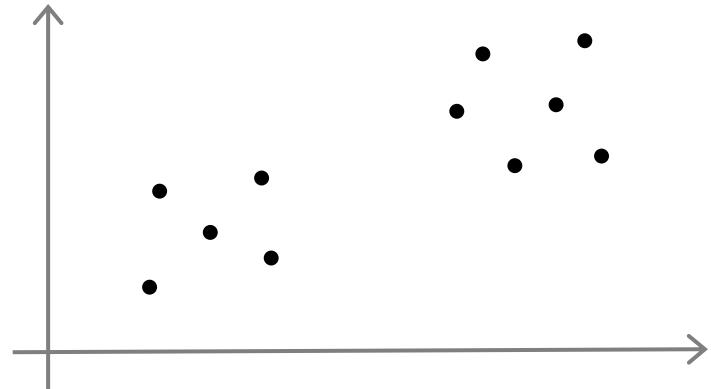
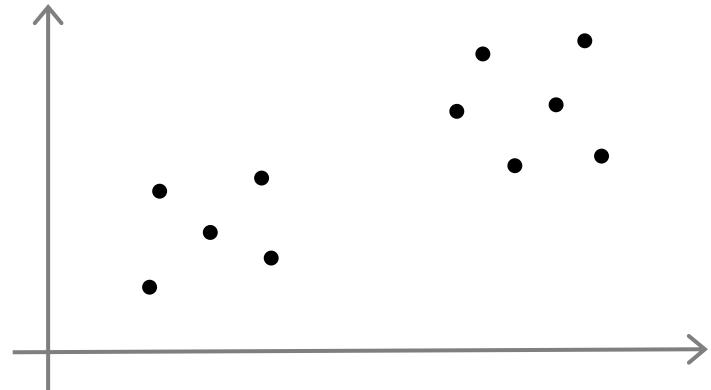
```
Repeat {  
    for  $i = 1$  to  $m$   
         $c^{(i)} :=$  index (from 1 to  $K$ ) of cluster centroid  
        closest to  $x^{(i)}$   
    for  $k = 1$  to  $K$   
         $\mu_k :=$  average (mean) of points assigned to cluster  $k$   
}
```

Random initialization

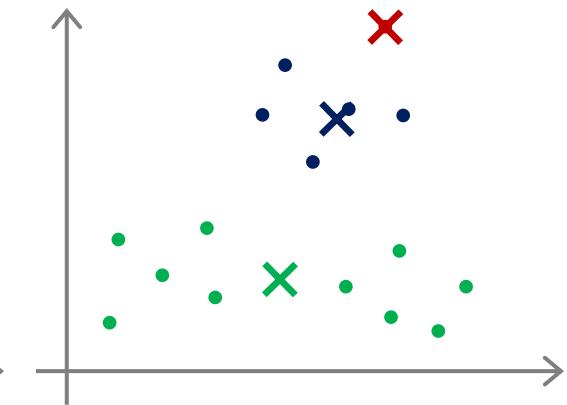
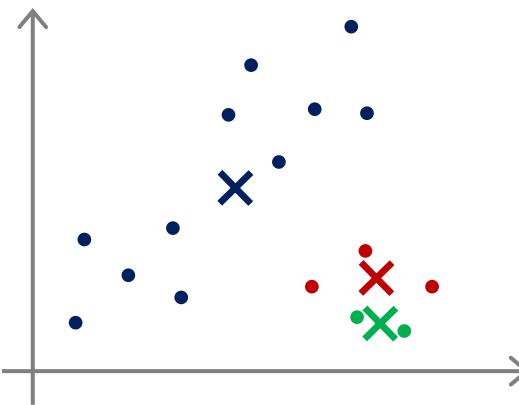
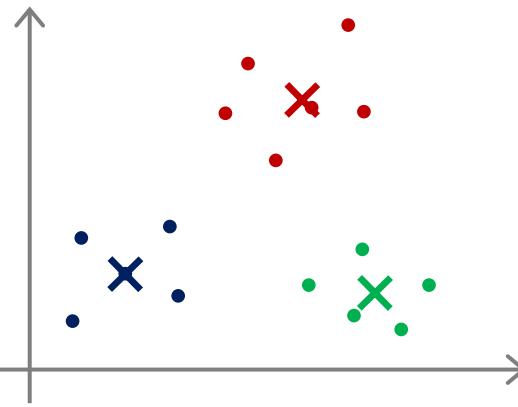
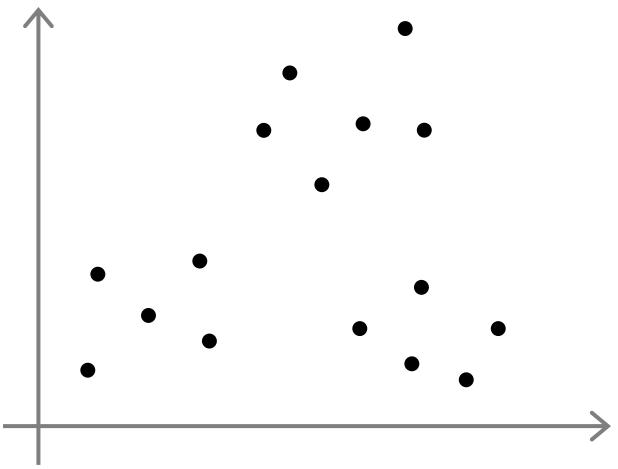
Should have $K < m$

Randomly pick K training examples.

Set μ_1, \dots, μ_K equal to these K examples.



Local optima



Random initialization

For i = 1 to 100 {

 Randomly initialize K-means.

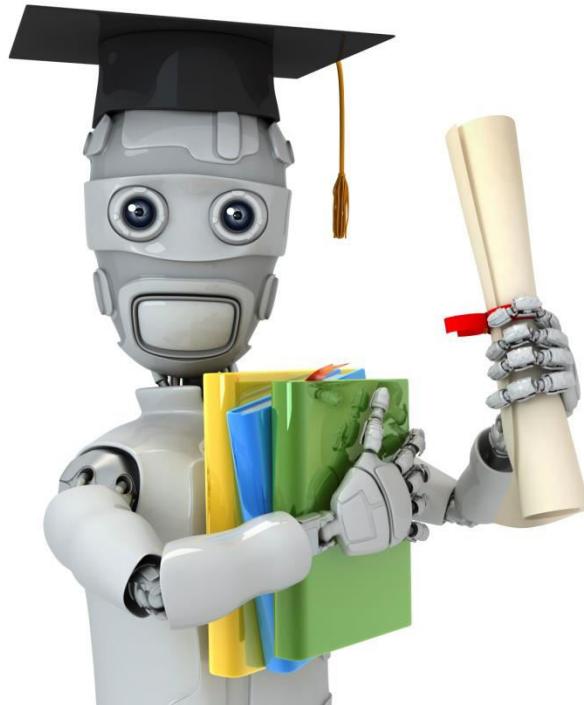
 Run K-means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$.

 Compute cost function (distortion)

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

}

Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

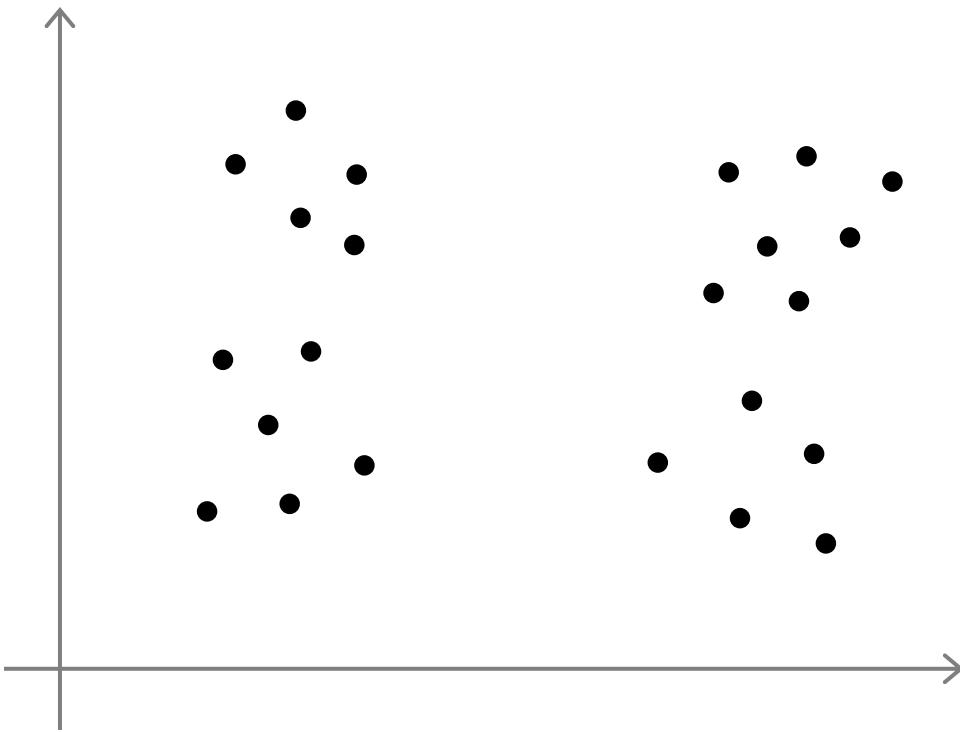


Machine Learning

Clustering

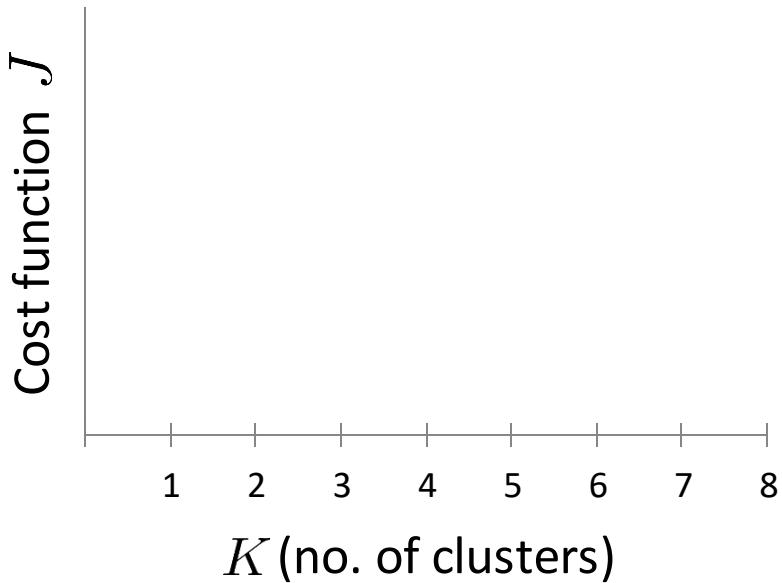
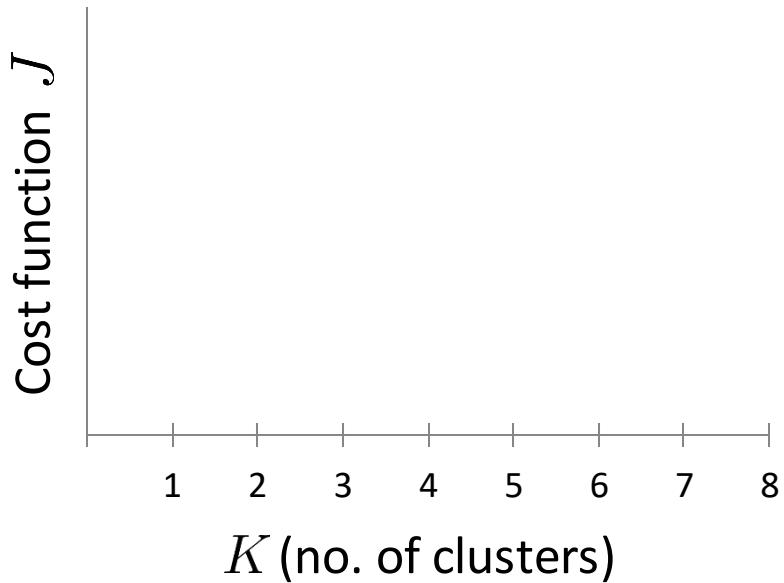
Choosing the number of clusters

What is the right value of K?



Choosing the value of K

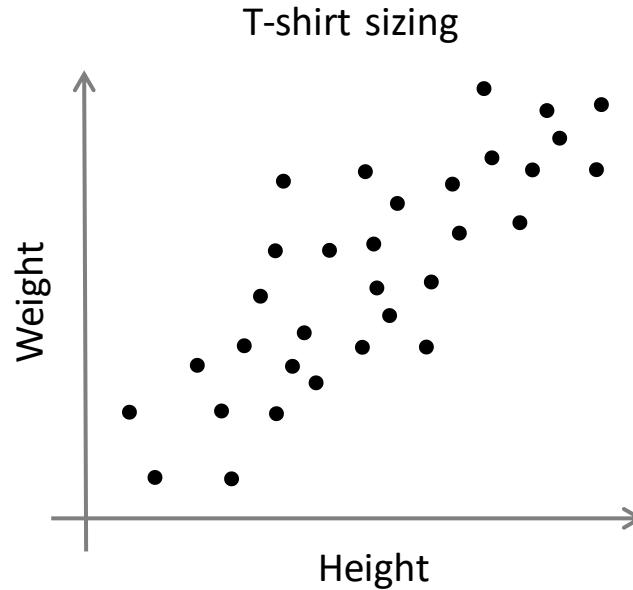
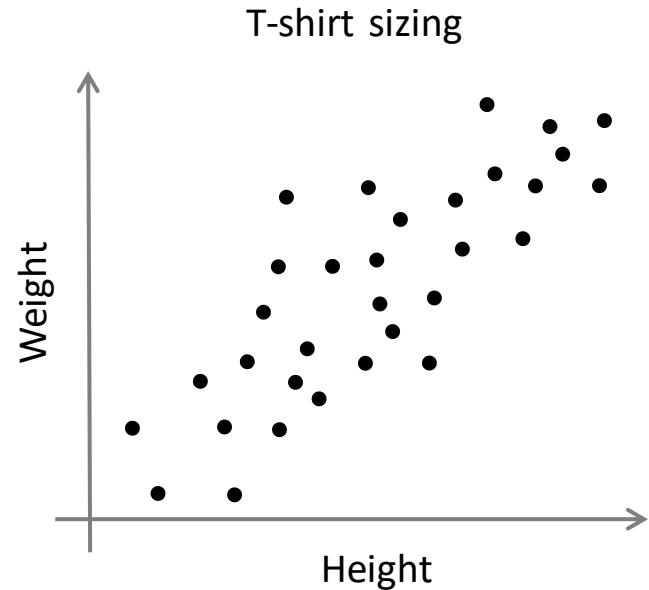
Elbow method:

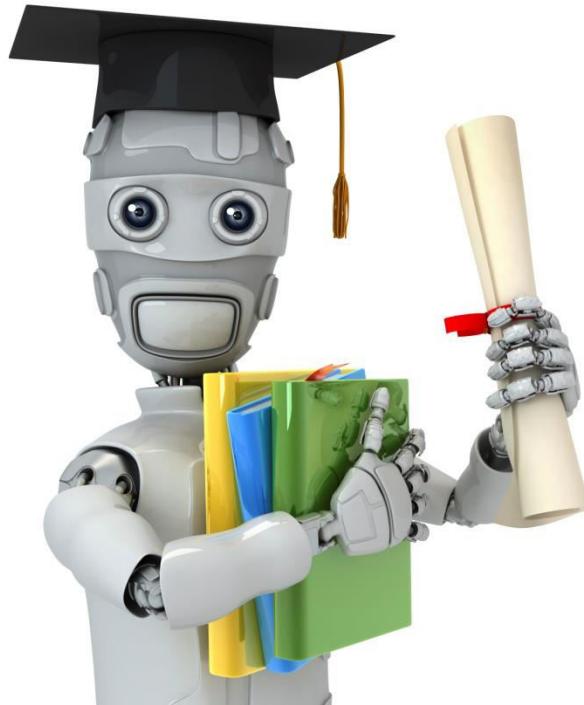


Choosing the value of K

Sometimes, you're running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

E.g.



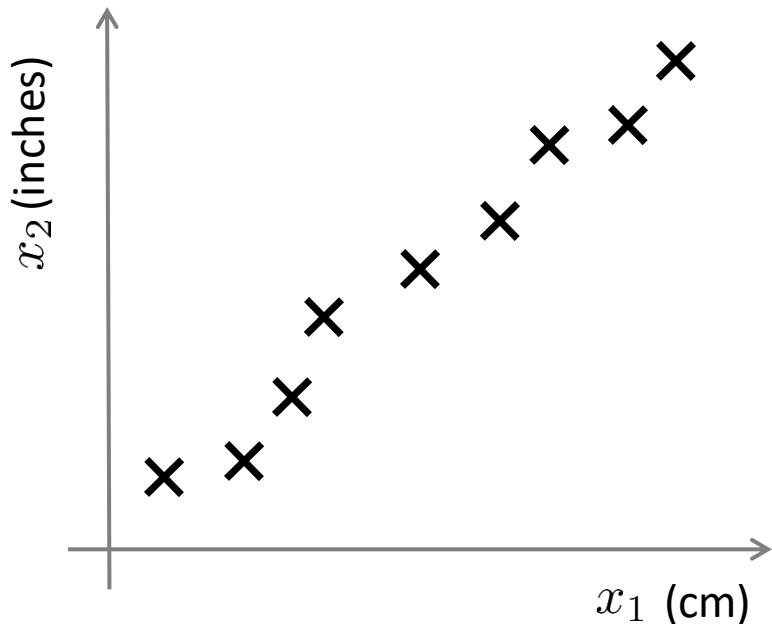


Machine Learning

Dimensionality Reduction

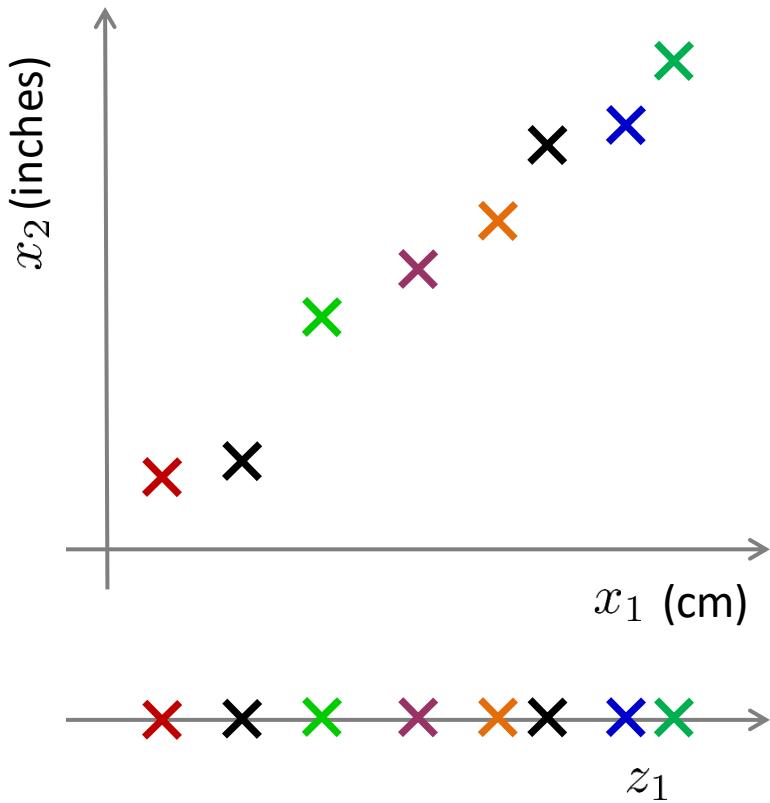
Motivation I:
Data Compression

Data Compression



Reduce data from
2D to 1D

Data Compression



Reduce data from
2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$

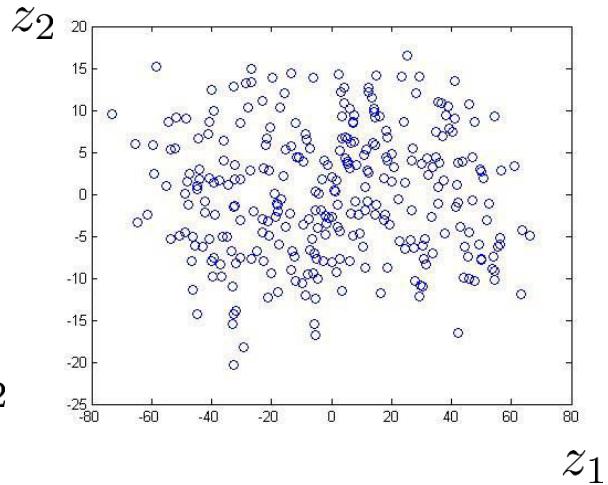
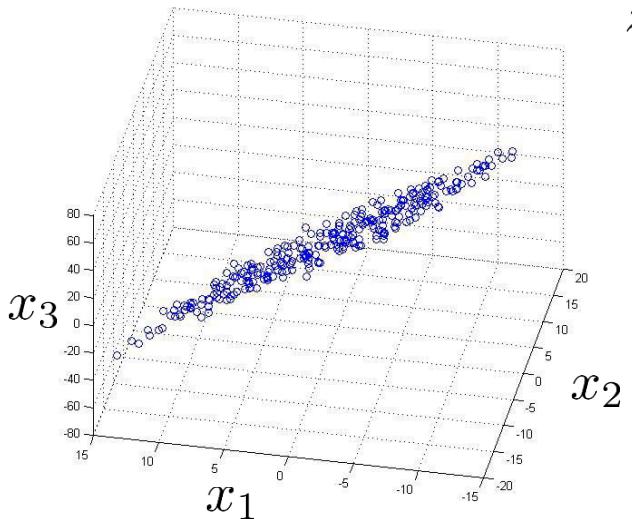
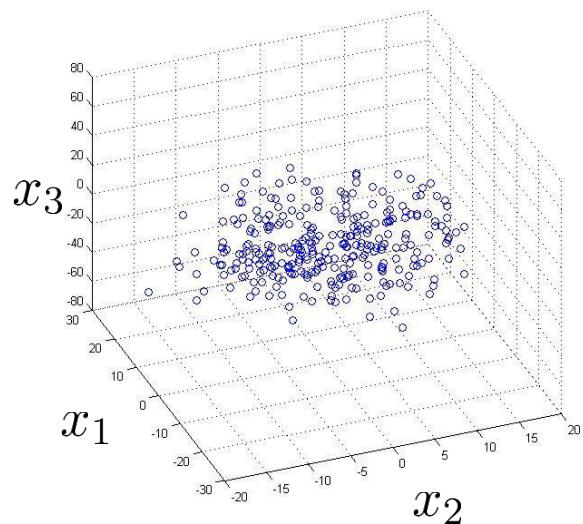
$$x^{(2)} \rightarrow z^{(2)}$$

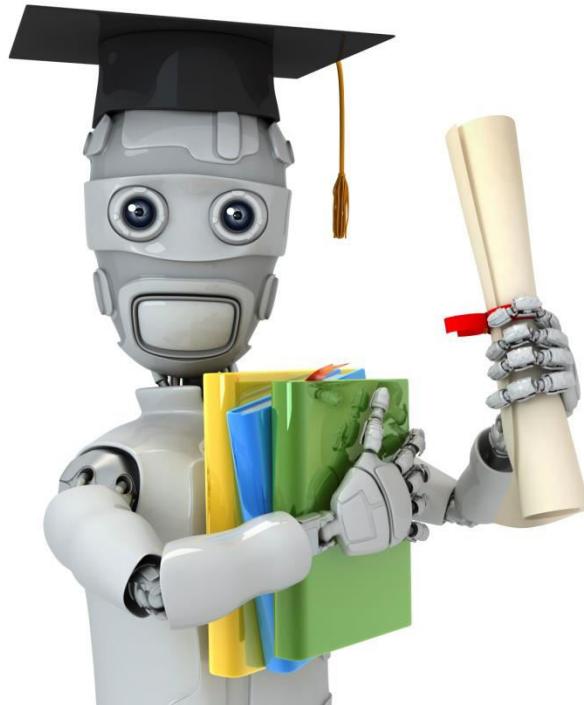
⋮

$$x^{(m)} \rightarrow z^{(m)}$$

Data Compression

Reduce data from 3D to 2D





Machine Learning

Dimensionality Reduction

Motivation II: Data Visualization

Data Visualization

Country	GDP (trillions of US\$)	Per capita GDP (thousands of intl. \$)	Human Develop- ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US\$)	...
Canada	1.577	39.17	0.908	80.7	32.6	67.293	...
China	5.878	7.54	0.687	73	46.9	10.22	...
India	1.632	3.41	0.547	64.7	36.8	0.735	...
Russia	1.48	19.84	0.755	65.5	39.9	0.72	...
Singapore	0.223	56.69	0.866	80	42.5	67.1	...
USA	14.527	46.86	0.91	78.3	40.8	84.3	...
...

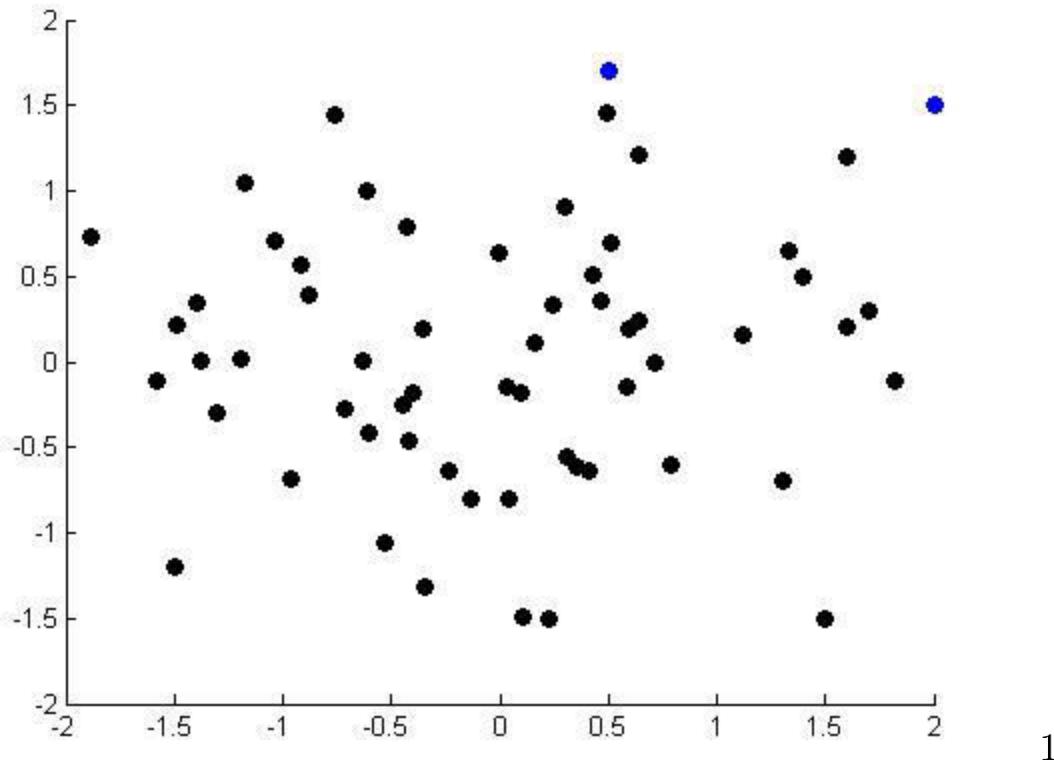
[resources from en.wikipedia.org]

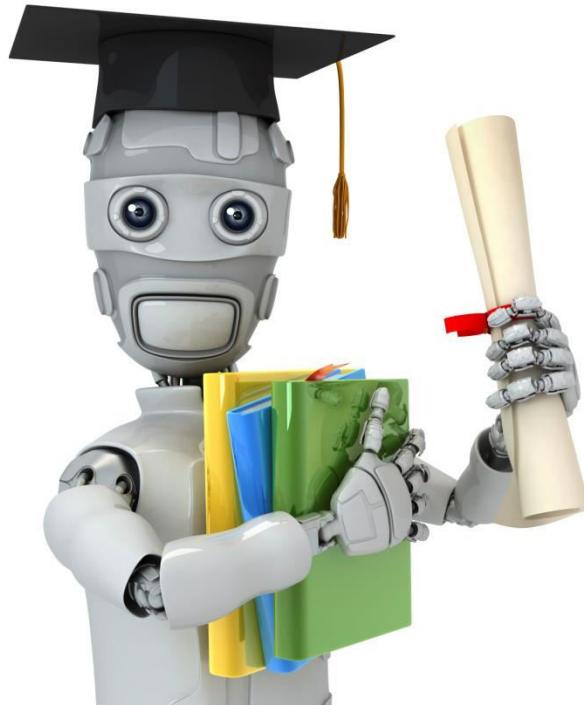
Andrew Ng

Data Visualization

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
...

Data Visualization



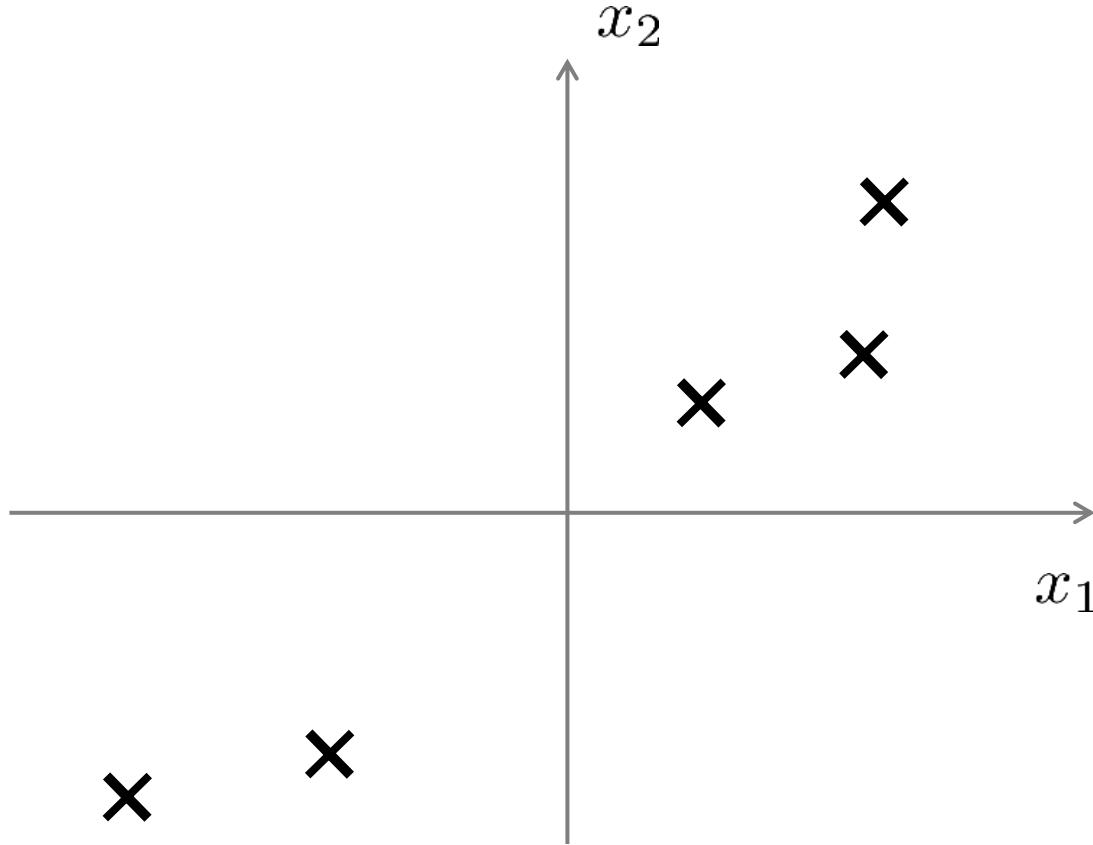


Machine Learning

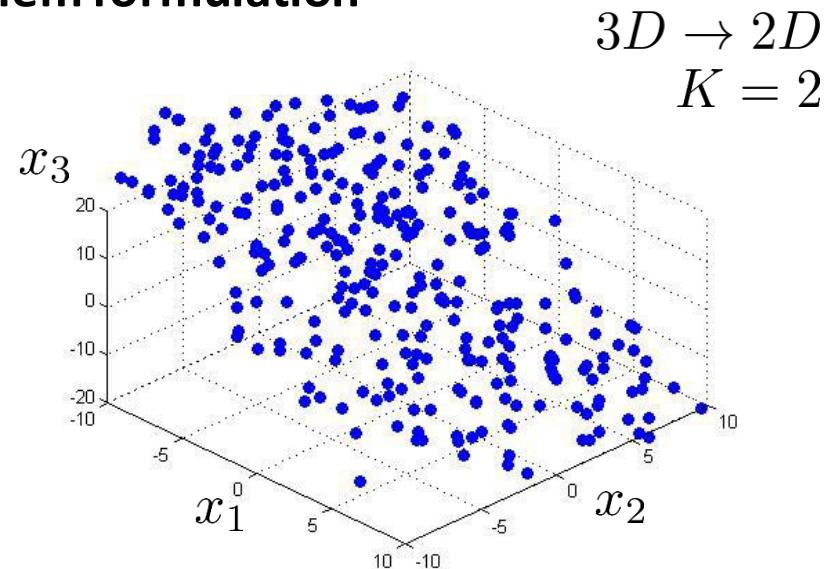
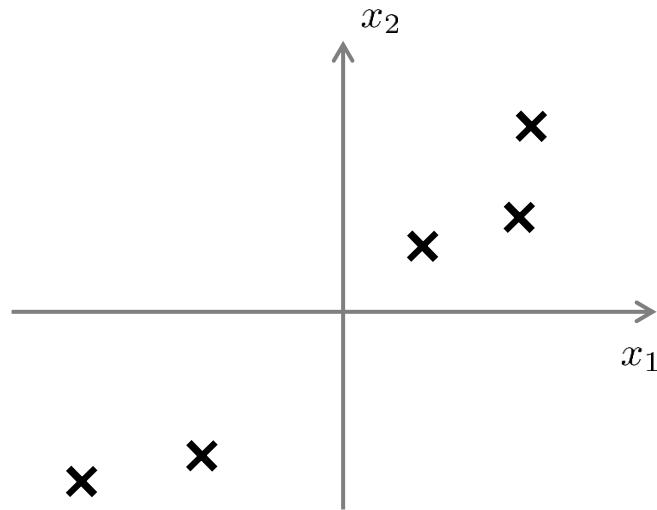
Dimensionality Reduction

Principal Component
Analysis problem
formulation

Principal Component Analysis (PCA) problem formulation



Principal Component Analysis (PCA) problem formulation

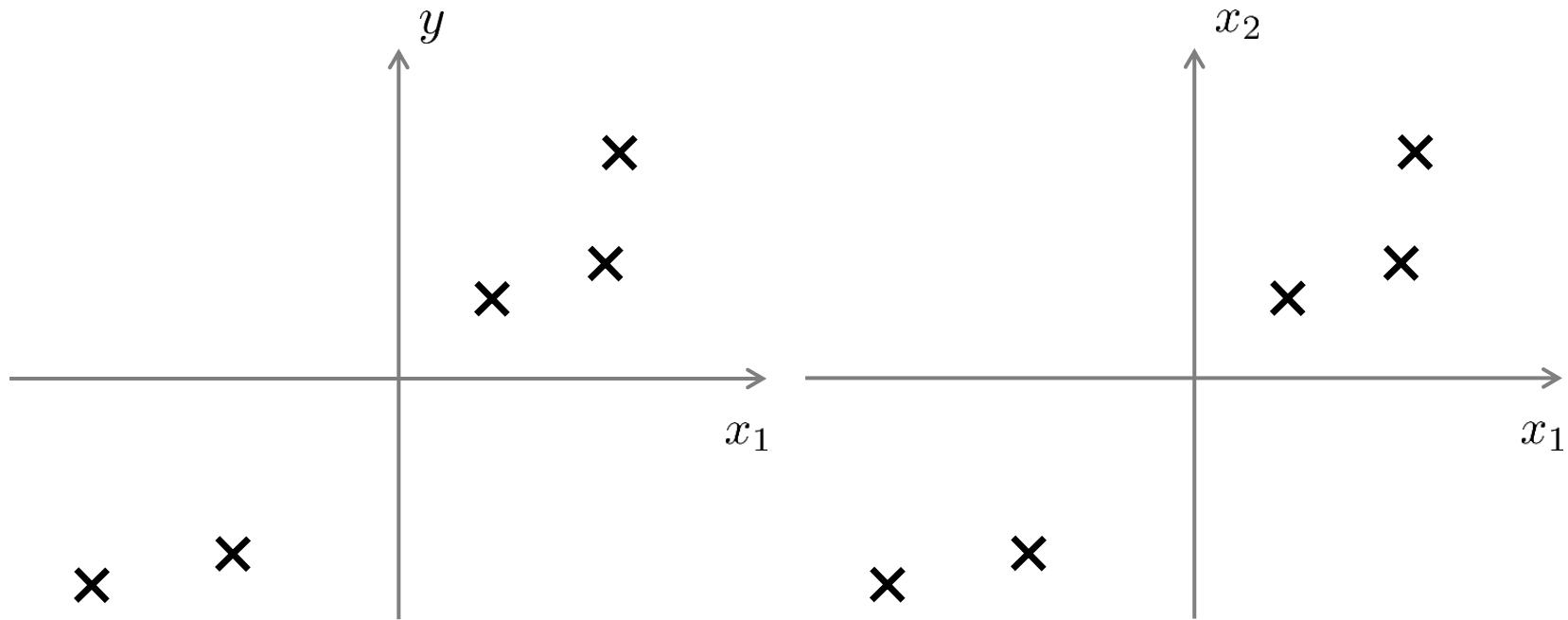


$$3D \rightarrow 2D \\ K = 2$$

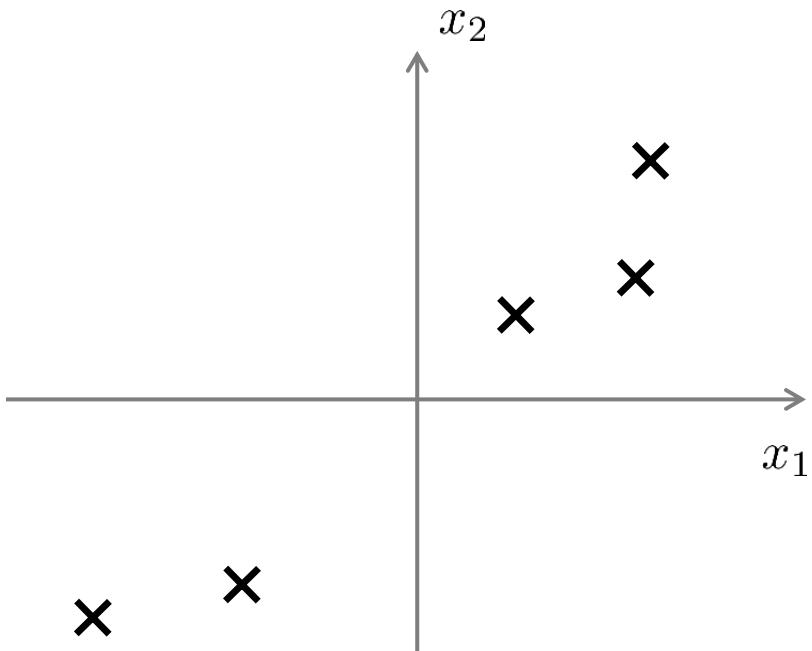
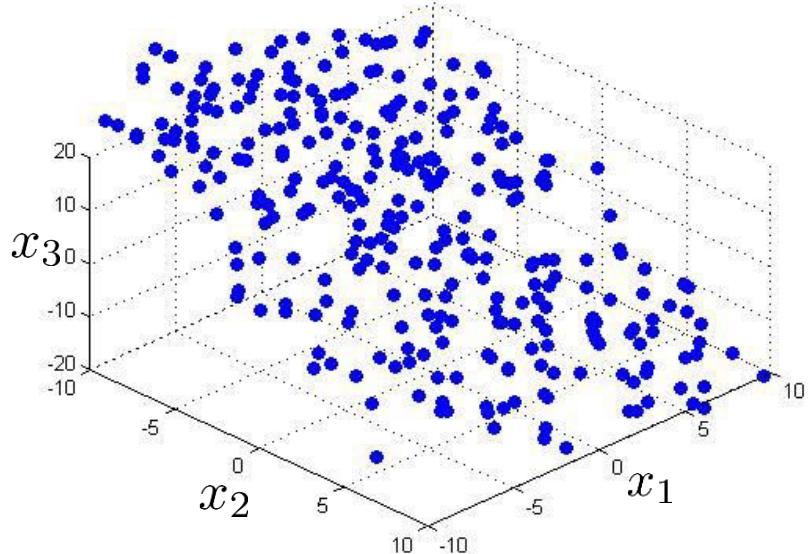
Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

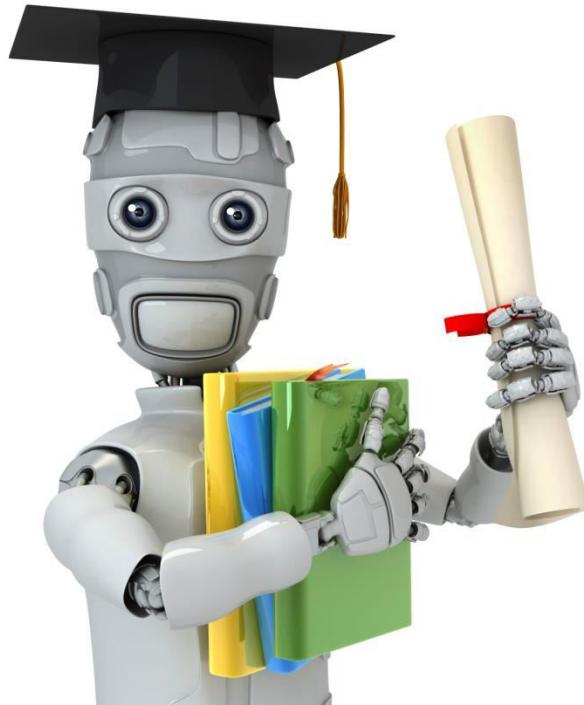
Reduce from n -dimension to k -dimension: Find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

PCA is not linear regression



PCA is not linear regression





Machine Learning

Dimensionality Reduction

Principal Component
Analysis algorithm

Data preprocessing

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

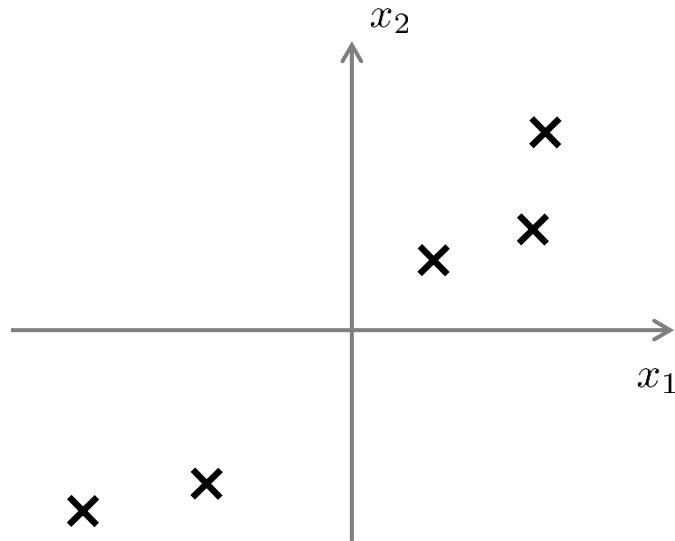
Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

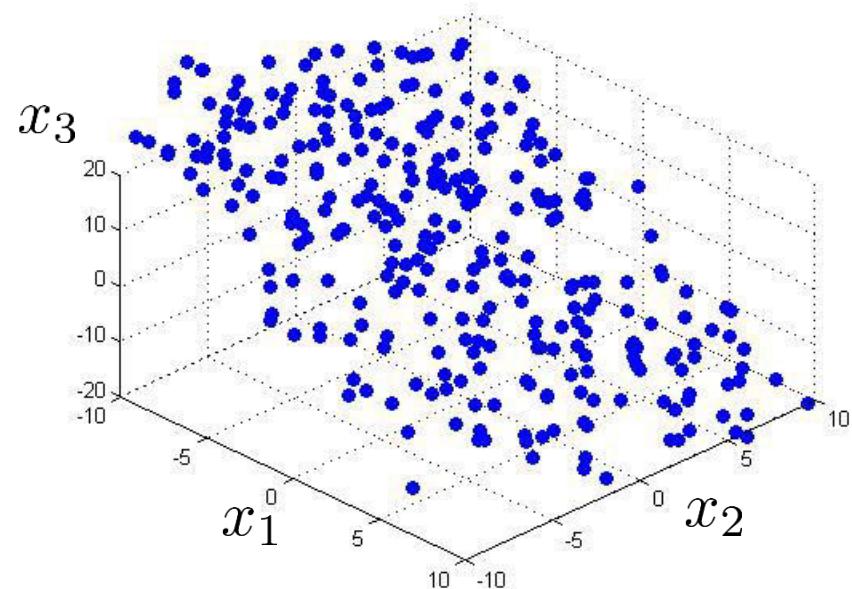
Replace each $x_j^{(i)}$ with $x_j^{(i)} - \mu_j$.

If different features on different scales (e.g., x_1 = size of house, x_2 = number of bedrooms), scale features to have comparable range of values.

Principal Component Analysis (PCA) algorithm



Reduce data from 2D to 1D



Reduce data from 3D to 2D

Principal Component Analysis (PCA) algorithm

Reduce data from n -dimensions to k -dimensions

Compute “covariance matrix”:

$$\Sigma = \frac{1}{m} \sum_{i=1}^n (x^{(i)})(x^{(i)})^T$$

Compute “eigenvectors” of matrix Σ :

`[U, S, V] = svd(Sigma);`

Principal Component Analysis (PCA) algorithm

From $[U, S, V] = \text{svd}(\Sigma)$, we get:

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Principal Component Analysis (PCA) algorithm summary

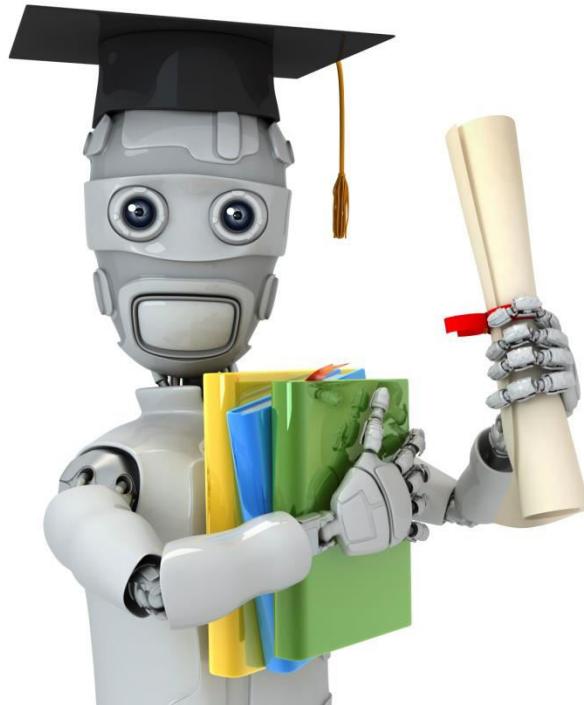
After mean normalization (ensure every feature has zero mean) and optionally feature scaling:

$$\text{Sigma} = \frac{1}{m} \sum_{i=1}^m (x^{(i)})(x^{(i)})^T$$

[U, S, V] = svd(Sigma);

Ureduce = U(:, 1:k);

z = Ureduce' * x;

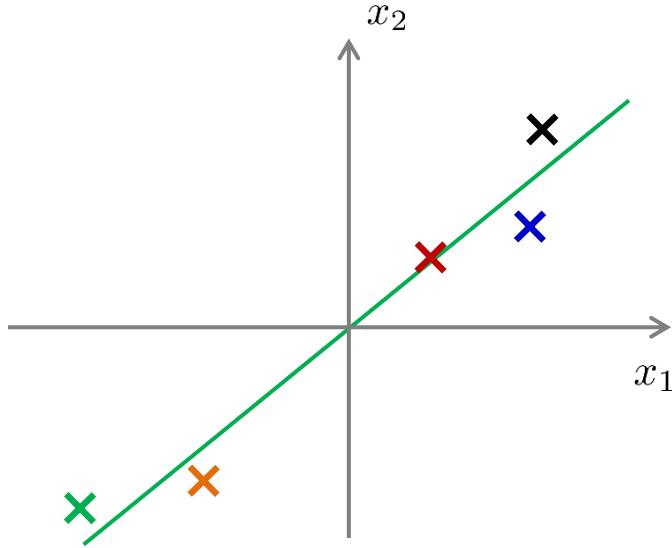


Dimensionality Reduction

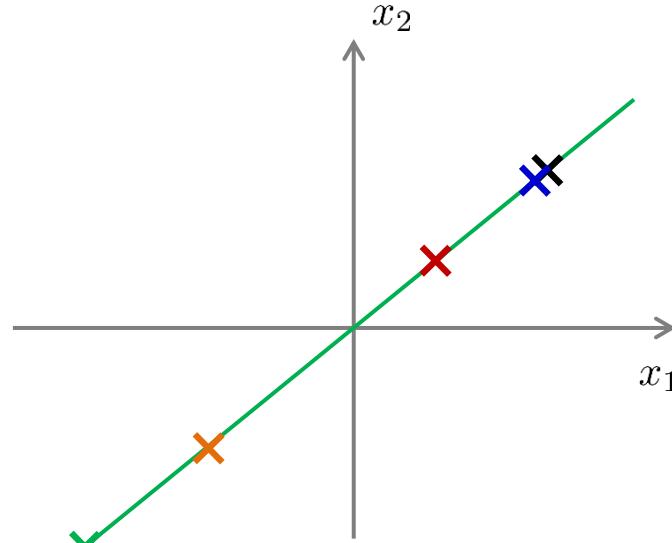
Reconstruction from
compressed
representation

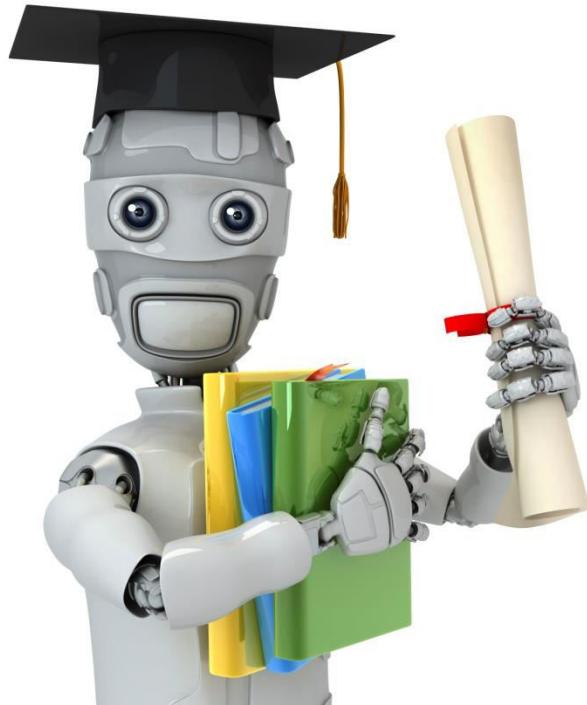
Machine Learning

Reconstruction from compressed representation



$$z = U_{reduce}^T x$$





Machine Learning

Dimensionality Reduction

Choosing the number of principal components

Choosing k (number of principal components)

Average squared projection error:

Total variation in the data:

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01 \quad (1\%)$$

“99% of variance is retained”

Choosing k (number of principal components)

Algorithm:

Try PCA with $k = 1$

Compute $U_{reduce}, z^{(1)}, z^{(2)}, \dots, z^{(m)}, x_{approx}^{(1)}, \dots, x_{approx}^{(m)}$

Check if

$$\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01?$$

$[U, S, V] = svd(Sigma)$

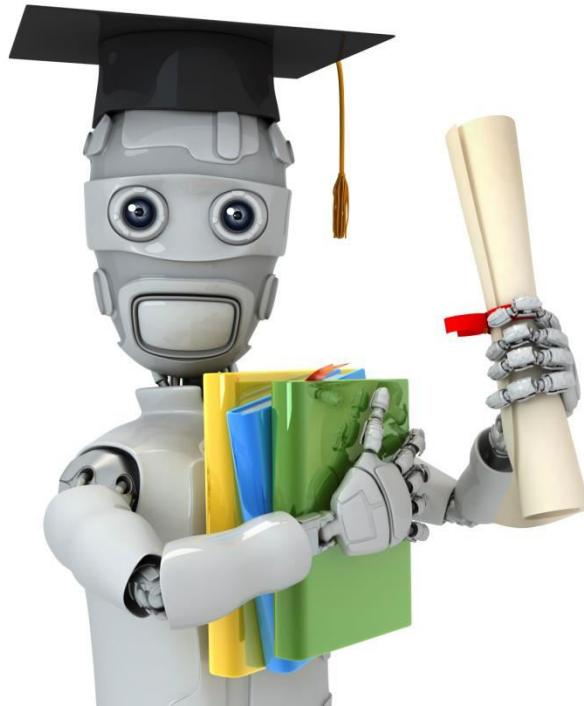
Choosing k (number of principal components)

$[U, S, V] = \text{svd}(Sigma)$

Pick smallest value of k for which

$$\frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^m S_{ii}} \geq 0.99$$

(99% of variance retained)



Machine Learning

Dimensionality Reduction

Advice for applying PCA

Supervised learning speedup

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$$

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^{10000}$

$\downarrow PCA$

$$z^{(1)}, z^{(2)}, \dots, z^{(m)} \in \mathbb{R}^{1000}$$

New training set:

$$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})$$

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

Application of PCA

- Compression
 - Reduce memory/disk needed to store data
 - Speed up learning algorithm
- Visualization

Bad use of PCA: To prevent overfitting

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$.

Thus, fewer features, less likely to overfit.

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

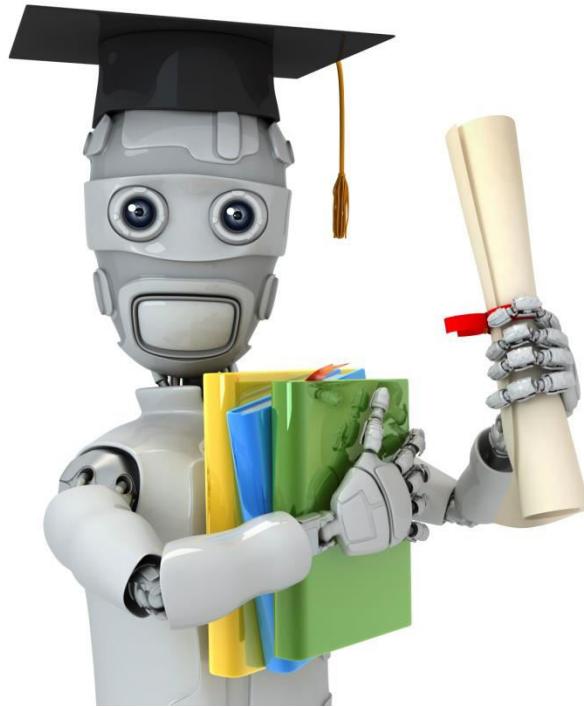
PCA is sometimes used where it shouldn't be

Design of ML system:

- Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
- Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
- Train logistic regression on $\{(z^{(1)}, y^{(1)}), \dots, (z^{(m)}, y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \dots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

How about doing the whole thing without using PCA?

Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.



Machine Learning

Anomaly detection

Problem motivation

Anomaly detection example

Aircraft engine features:

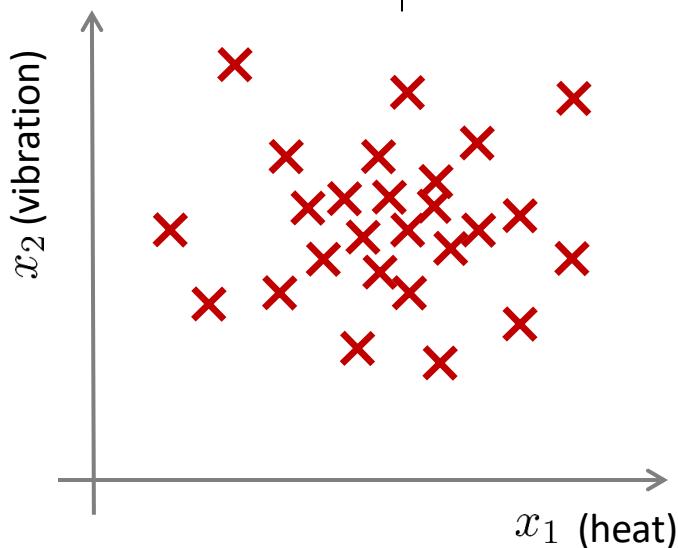
x_1 = heat generated

x_2 = vibration intensity

...

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

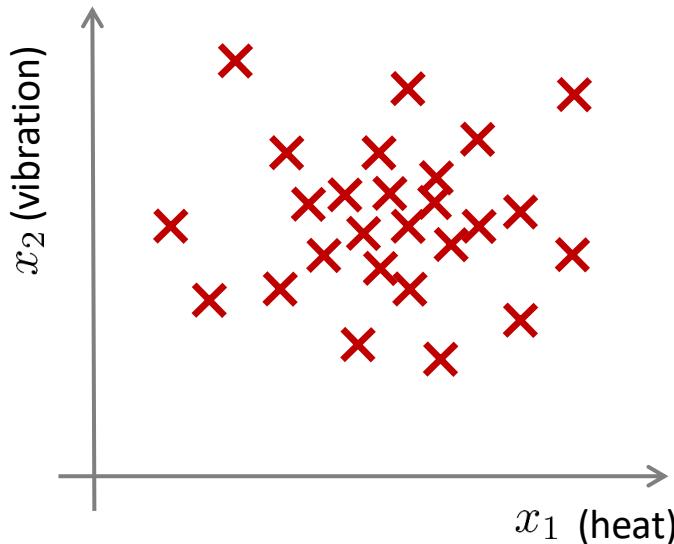
New engine: x_{test}



Density estimation

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

Is x_{test} anomalous?



Anomaly detection example

Fraud detection:

$x^{(i)}$ = features of user i 's activities

Model $p(x)$ from data.

Identify unusual users by checking which have $p(x) < \varepsilon$

Manufacturing

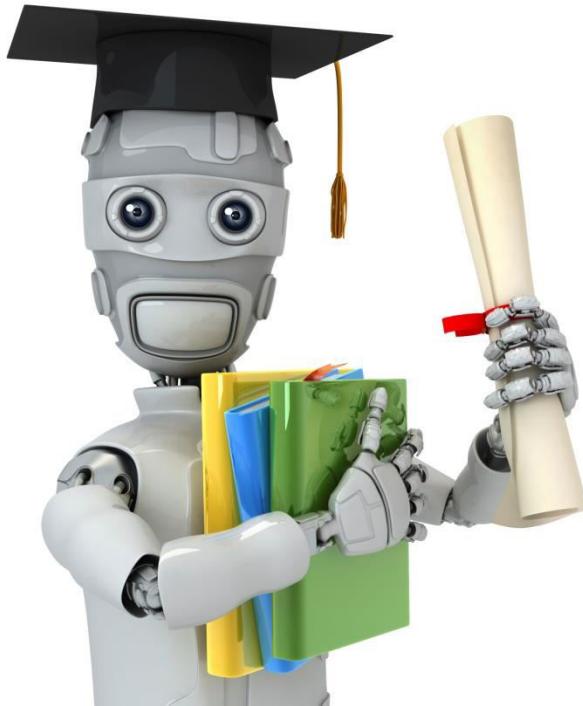
Monitoring computers in a data center.

$x^{(i)}$ = features of machine i

x_1 = memory use, x_2 = number of disk accesses/sec,

x_3 = CPU load, x_4 = CPU load/network traffic.

...



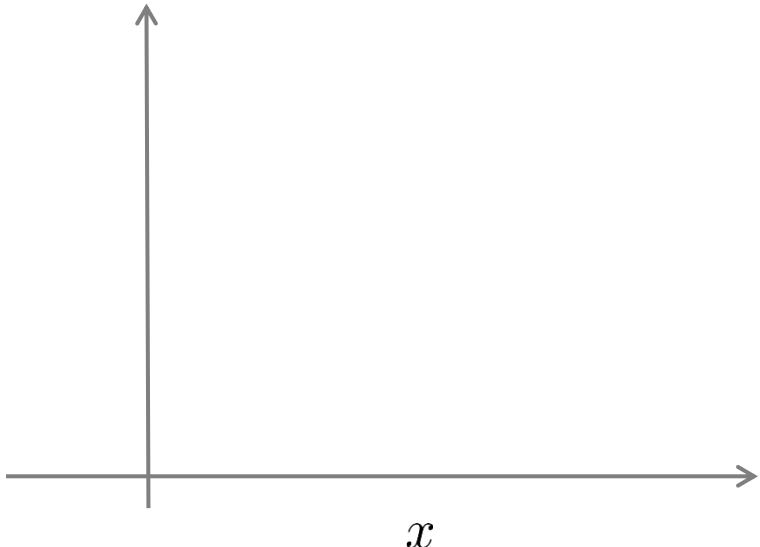
Machine Learning

Anomaly detection

Gaussian distribution

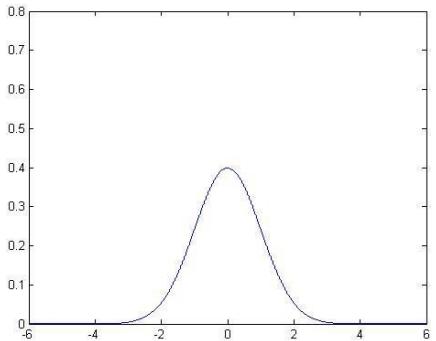
Gaussian (Normal) distribution

Say $x \in \mathbb{R}$. If x is a distributed Gaussian with mean μ , variance σ^2 .

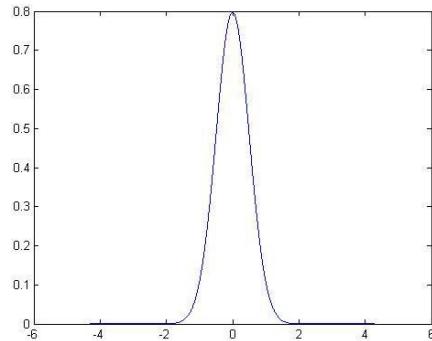


Gaussian distribution example

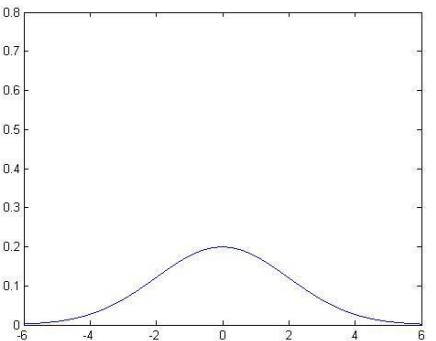
$$\mu = 0, \sigma = 1$$



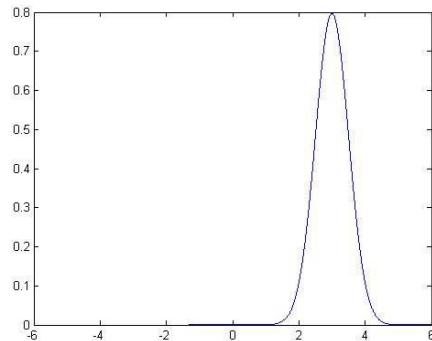
$$\mu = 0, \sigma = 0.5$$



$$\mu = 0, \sigma = 2$$



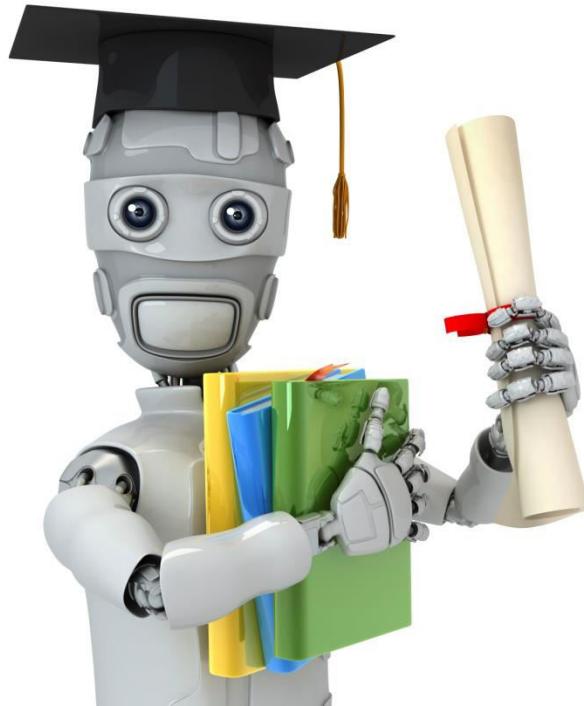
$$\mu = 3, \sigma = 0.5$$



Parameter estimation

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}$





Machine Learning

Anomaly detection

Algorithm

Density estimation

Training set: $\{x^{(1)}, \dots, x^{(m)}\}$

Each example is $x \in \mathbb{R}^n$

Anomaly detection algorithm

1. Choose features x_i that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

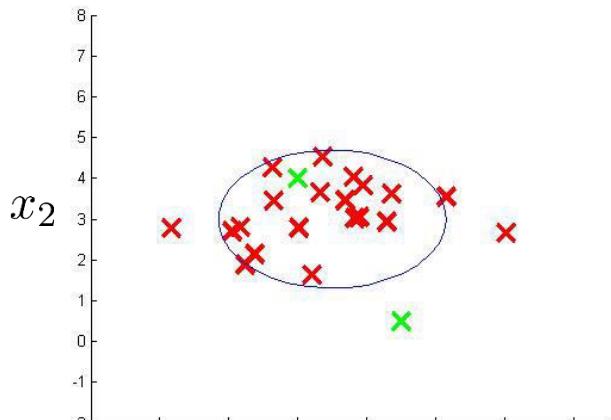
$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

3. Given new example x , compute $p(x)$:

$$p(x) = \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^n \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

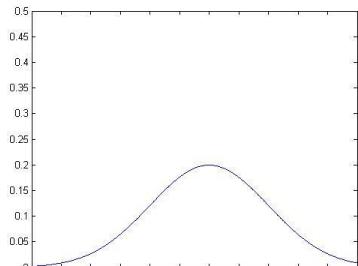
Anomaly if $p(x) < \varepsilon$

Anomaly detection example

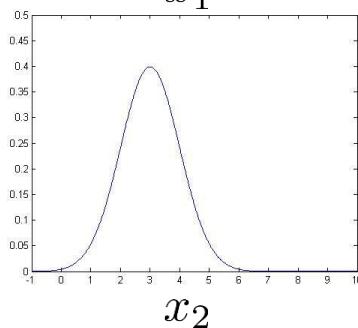


$$\mu_1 = 5, \sigma_1 = 2$$

$$\mu_2 = 3, \sigma_2 = 1$$



$$p(x_1; \mu_1, \sigma_1^2)$$

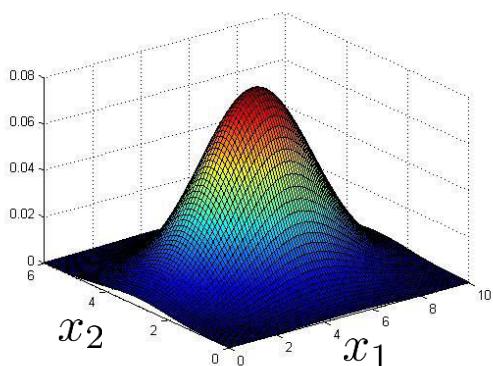


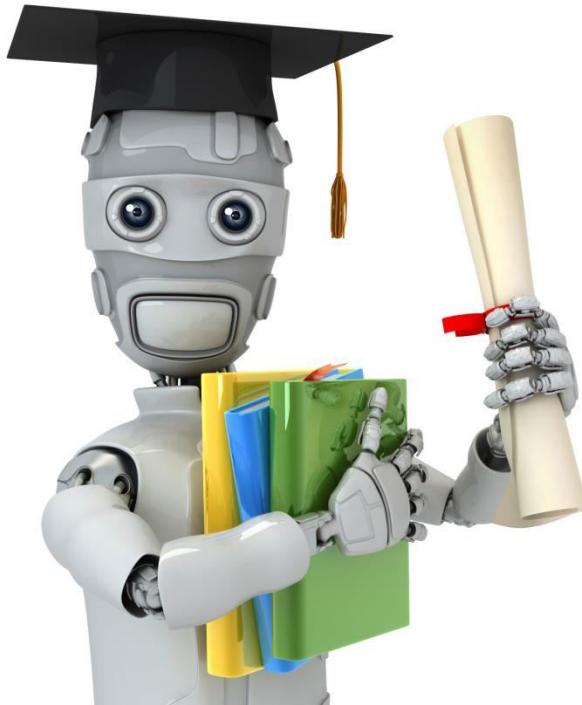
$$p(x_2; \mu_2, \sigma_2^2)$$

$$\varepsilon = 0.02$$

$$p(x_{test}^{(1)}) = 0.0426$$

$$p(x_{test}^{(2)}) = 0.0021$$





Machine Learning

Anomaly detection

Developing and evaluating an anomaly detection system

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our learning algorithm.

Assume we have some labeled data, of anomalous and non-anomalous examples. ($y = 0$ if normal, $y = 1$ if anomalous).

Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ (assume normal examples/not anomalous)

Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Aircraft engines motivating example

10000 good (normal) engines

20 flawed engines (anomalous)

Training set: 6000 good engines

CV: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Test: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Alternative:

Training set: 6000 good engines

CV: 4000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Test: 4000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Algorithm evaluation

Fit model $p(x)$ on training set $\{x^{(1)}, \dots, x^{(m)}\}$

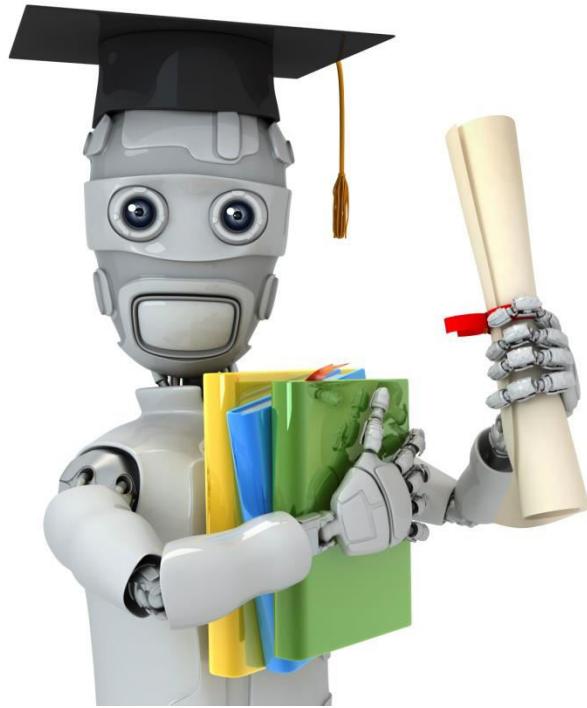
On a cross validation/test example x , predict

$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \text{ (anomaly)} \\ 0 & \text{if } p(x) \geq \varepsilon \text{ (normal)} \end{cases}$$

Possible evaluation metrics:

- True positive, false positive, false negative, true negative
- Precision/Recall
- F_1 -score

Can also use cross validation set to choose parameter ε



Machine Learning

Anomaly detection

Anomaly detection
vs. supervised
learning

Anomaly detection

vs.

Supervised learning

Very small number of positive examples ($y = 1$). (0-20 is common).

Large number of negative ($y = 0$) examples.

Many different “types” of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we’ve seen so far.

Large number of positive and negative examples.

Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set.

Anomaly detection

vs.

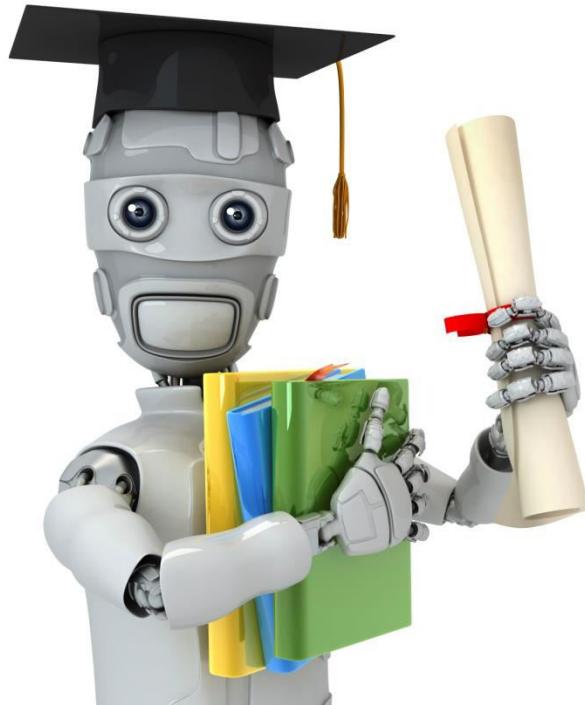
Supervised learning

- Fraud detection
- Manufacturing (e.g. aircraft engines)
- Monitoring machines in a data center

:

- Email spam classification
- Weather prediction (sunny/rainy/etc).
- Cancer classification

:

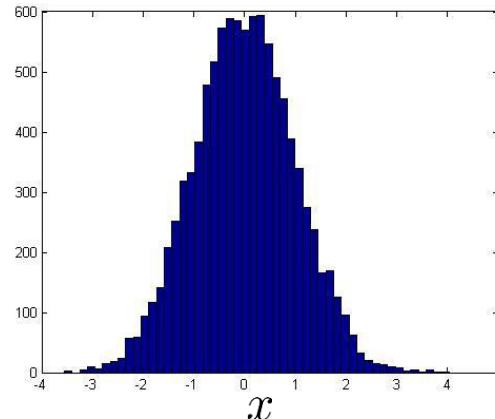
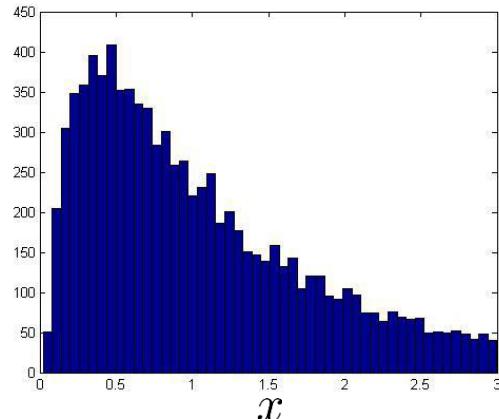
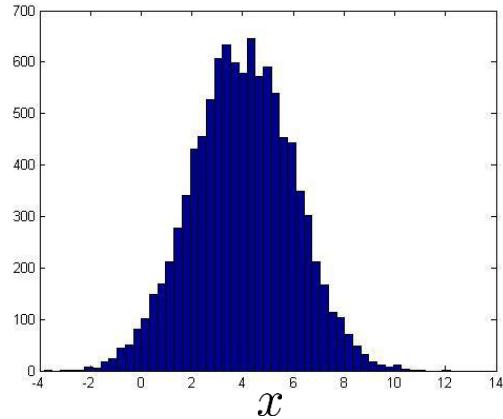


Machine Learning

Anomaly detection

Choosing what features to use

Non-gaussian features



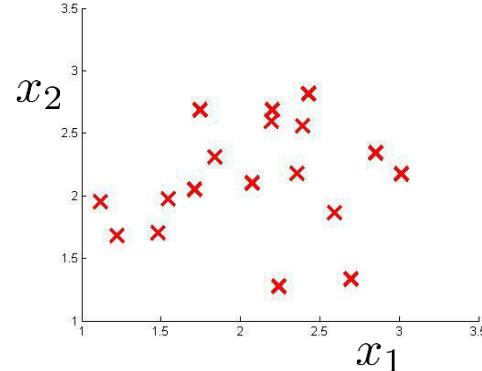
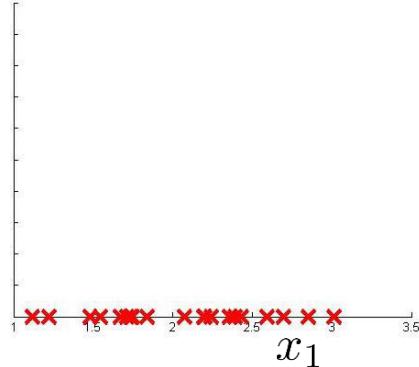
Error analysis for anomaly detection

Want $p(x)$ large for normal examples x .

$p(x)$ small for anomalous examples x .

Most common problem:

$p(x)$ is comparable (say, both large) for normal and anomalous examples



Monitoring computers in a data center

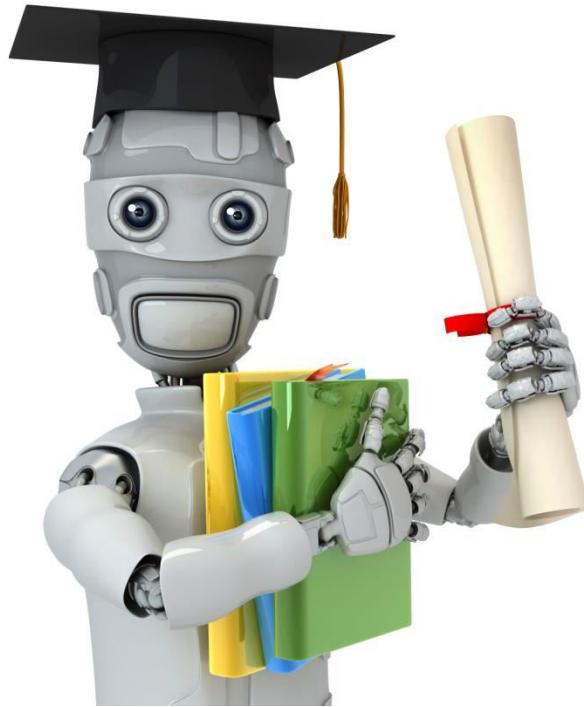
Choose features that might take on unusually large or small values in the event of an anomaly.

x_1 = memory use of computer

x_2 = number of disk accesses/sec

x_3 = CPU load

x_4 = network traffic

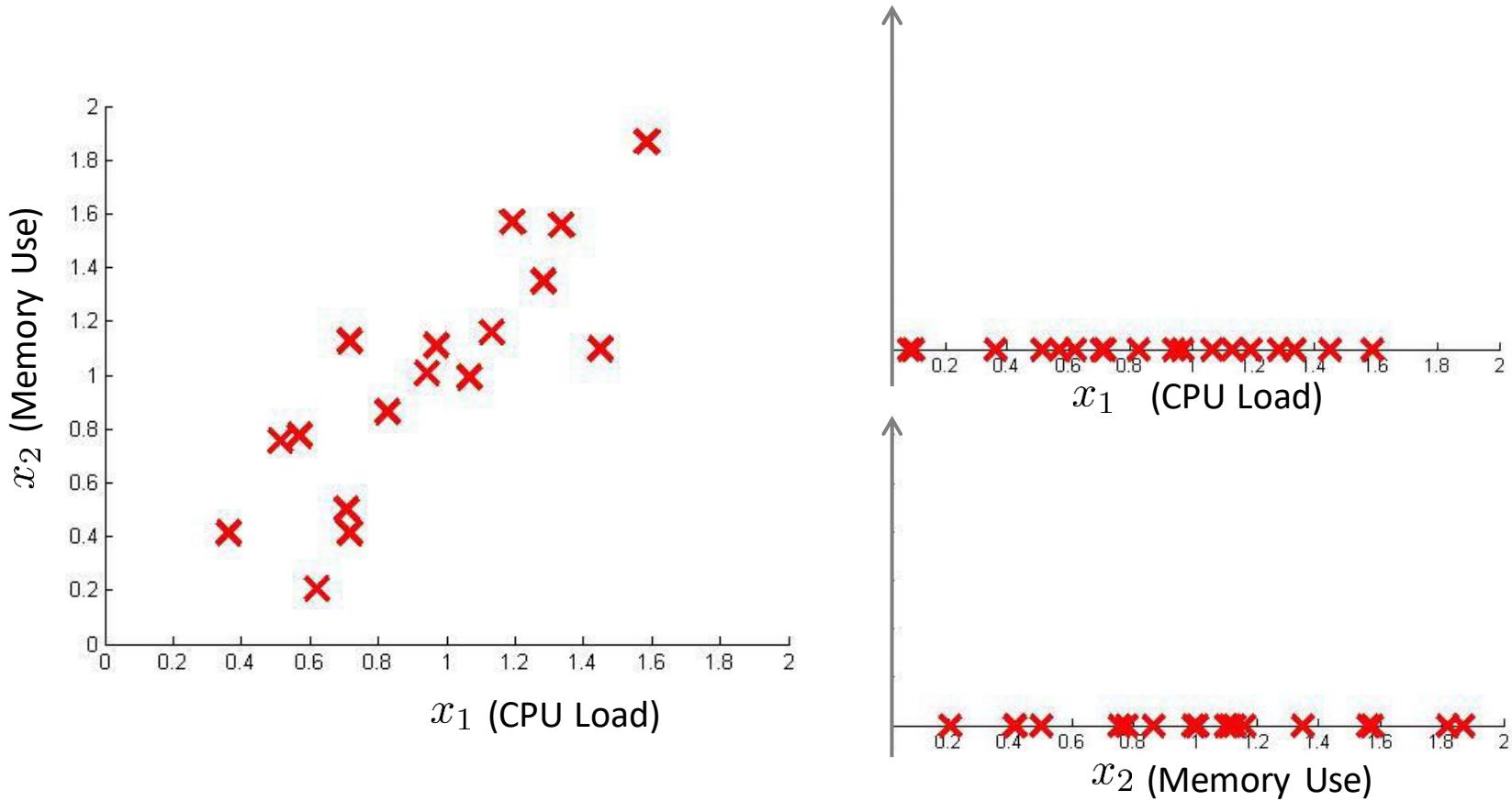


Machine Learning

Anomaly detection

Multivariate
Gaussian distribution

Motivating example: Monitoring machines in a data center



Multivariate Gaussian (Normal) distribution

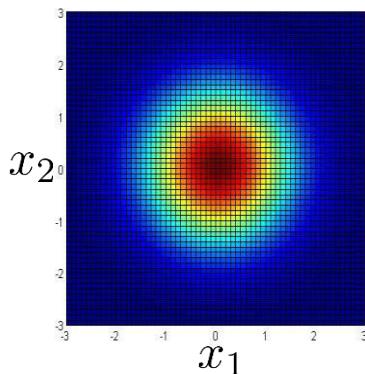
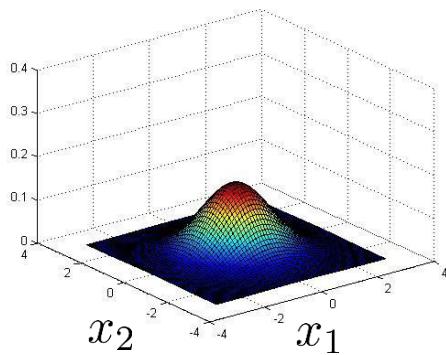
$x \in \mathbb{R}^n$. Don't model $p(x_1), p(x_2), \dots$, etc. separately.

Model $p(x)$ all in one go.

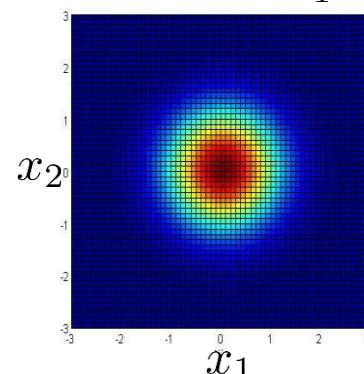
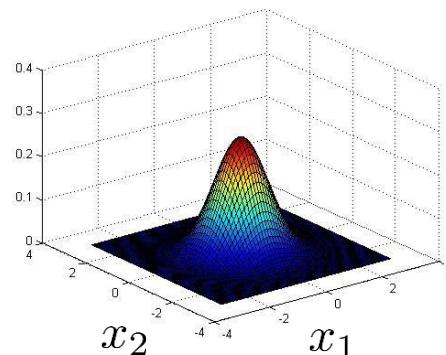
Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix)

Multivariate Gaussian (Normal) examples

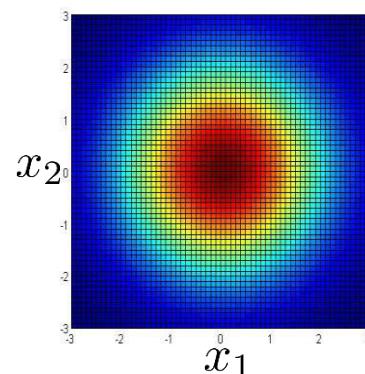
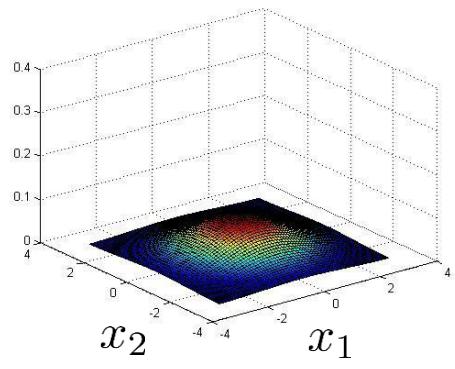
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$$

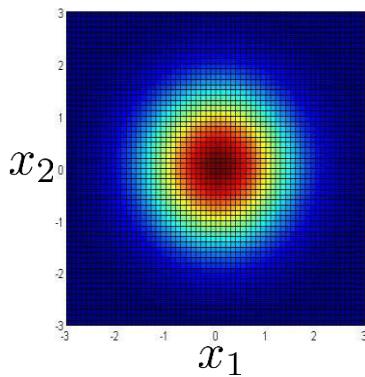
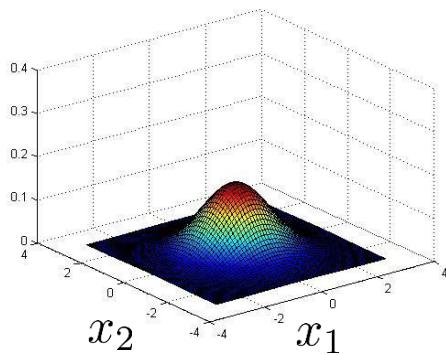


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

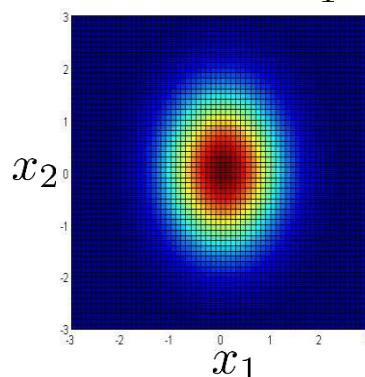
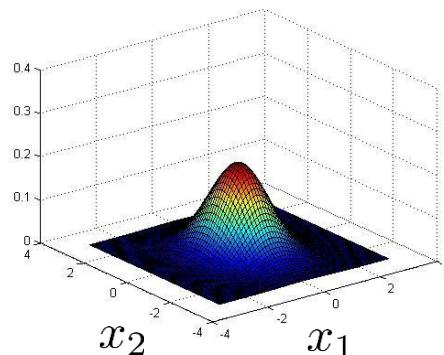


Multivariate Gaussian (Normal) examples

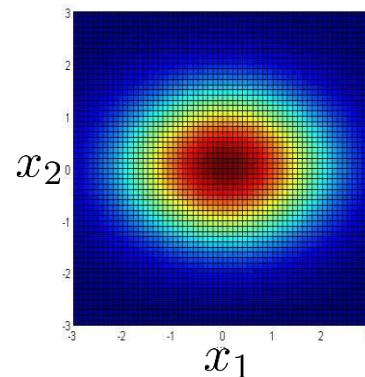
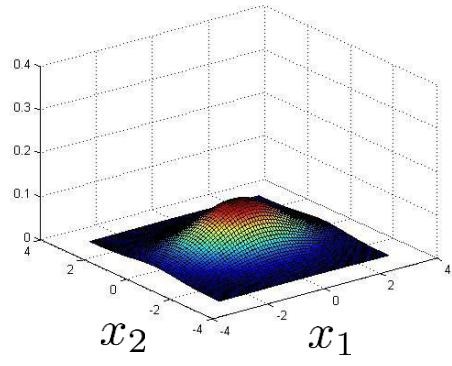
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$$

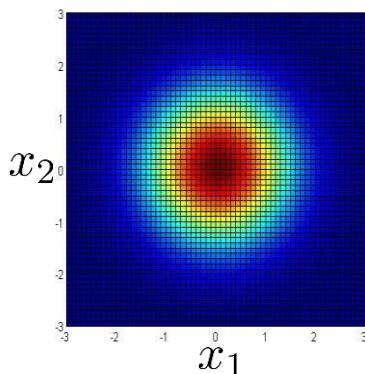
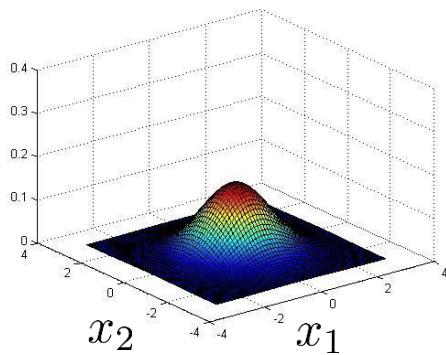


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

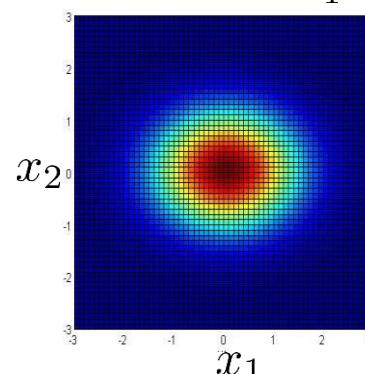
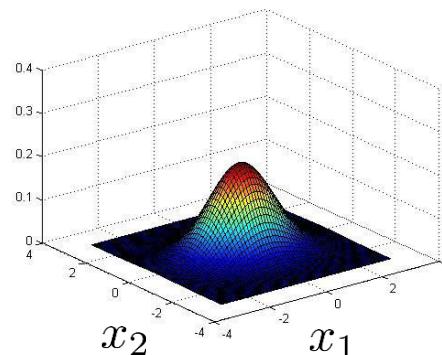


Multivariate Gaussian (Normal) examples

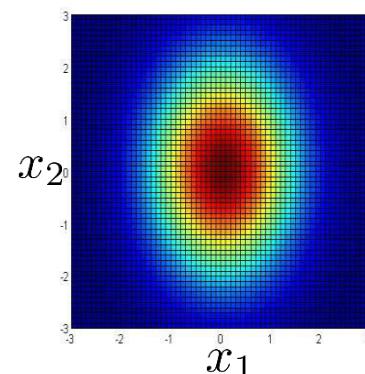
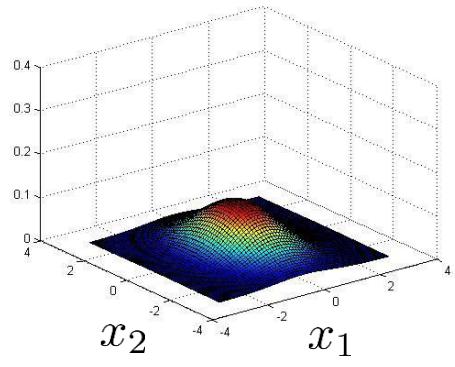
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \end{bmatrix}$$

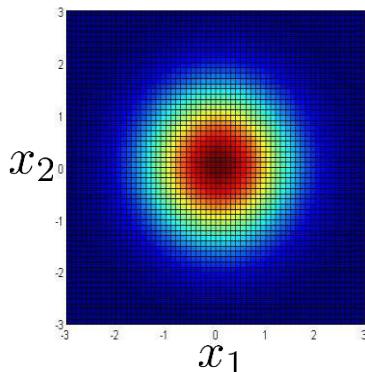
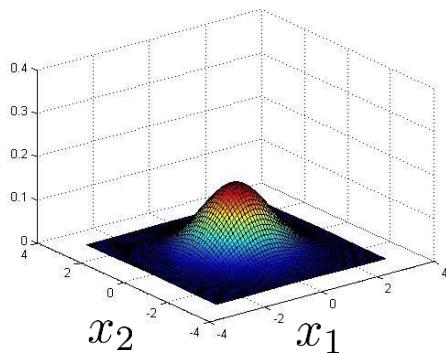


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

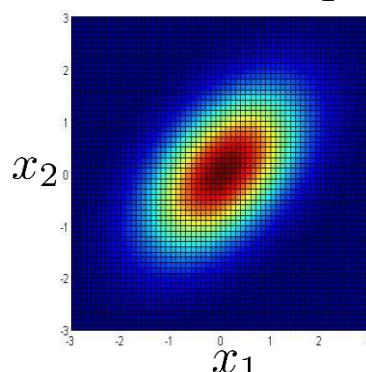
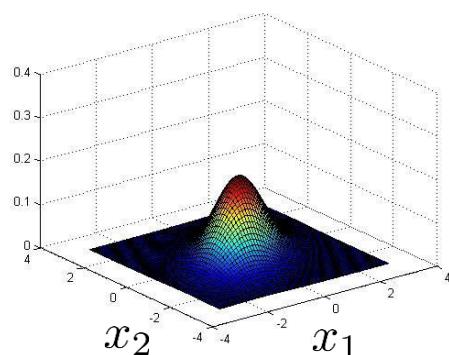


Multivariate Gaussian (Normal) examples

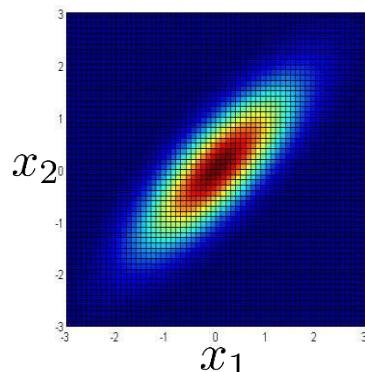
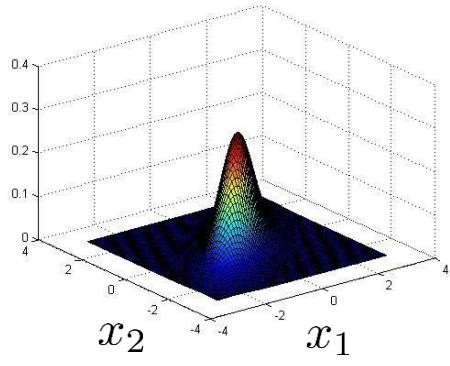
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

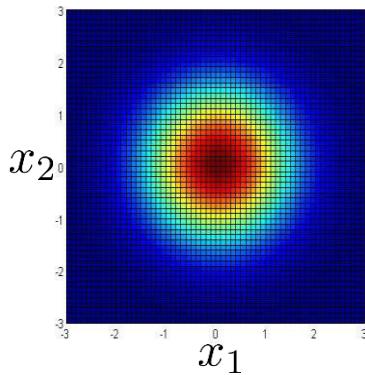
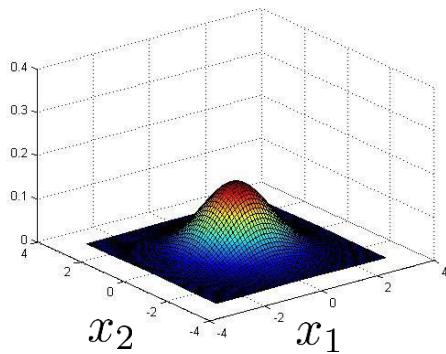


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

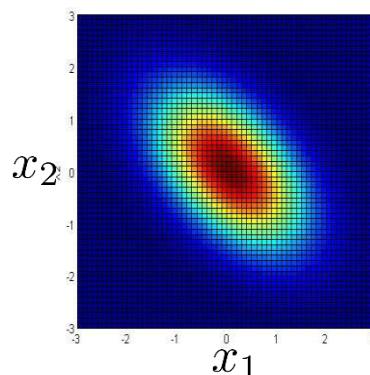
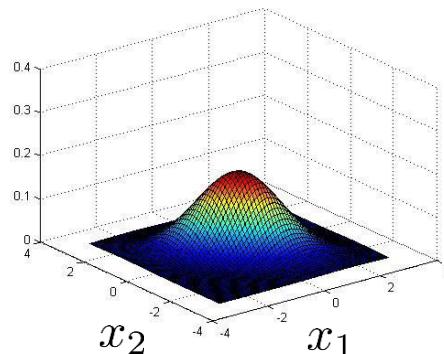


Multivariate Gaussian (Normal) examples

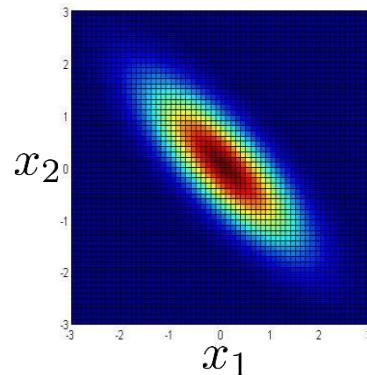
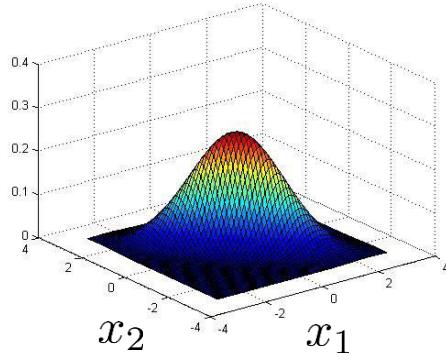
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

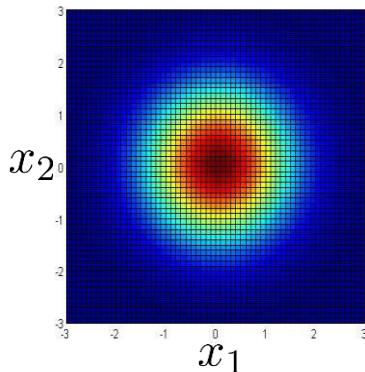
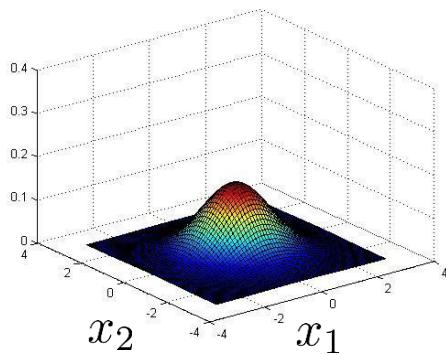


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$

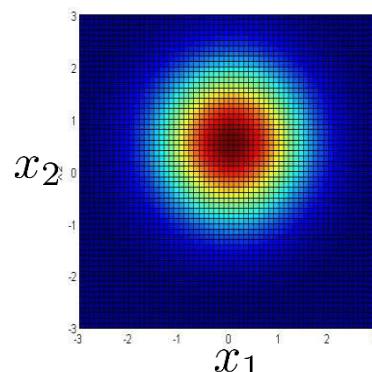
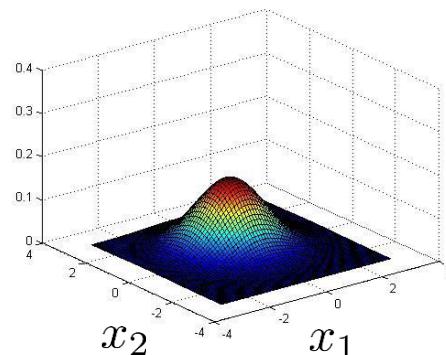


Multivariate Gaussian (Normal) examples

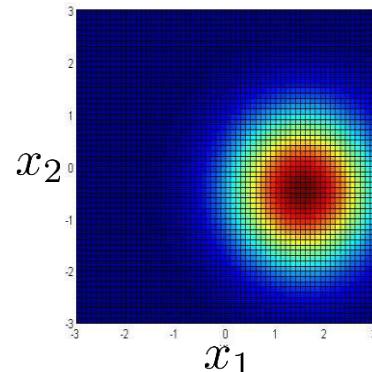
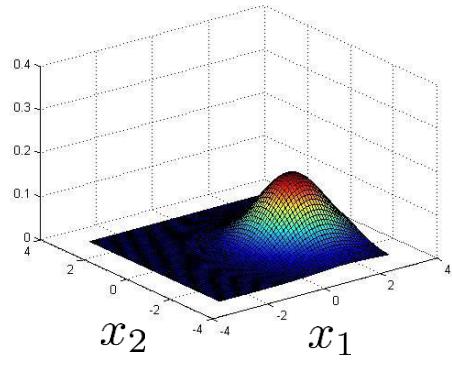
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

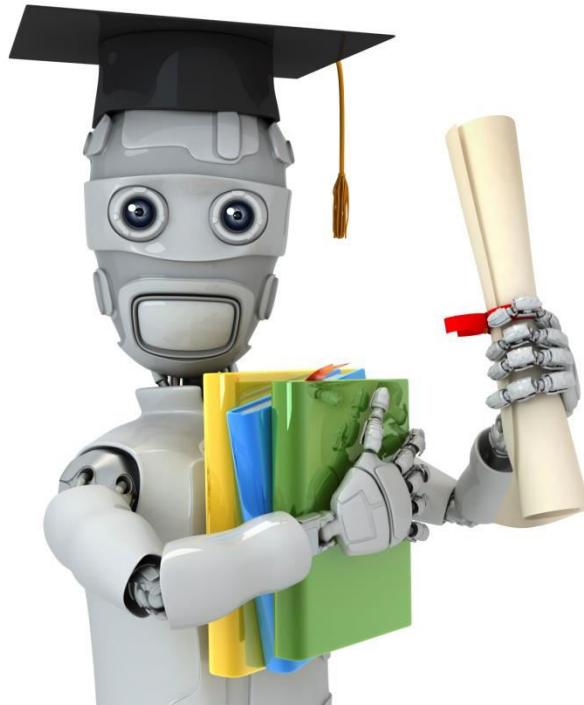


$$\mu = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$





Machine Learning

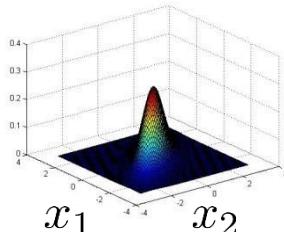
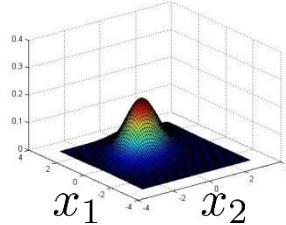
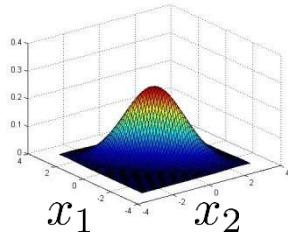
Anomaly detection

Anomaly detection using
the multivariate
Gaussian distribution

Multivariate Gaussian (Normal) distribution

Parameters μ, Σ

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$



Parameter fitting:

Given training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)} \quad \Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

Anomaly detection with the multivariate Gaussian

1. Fit model $p(x)$ by setting

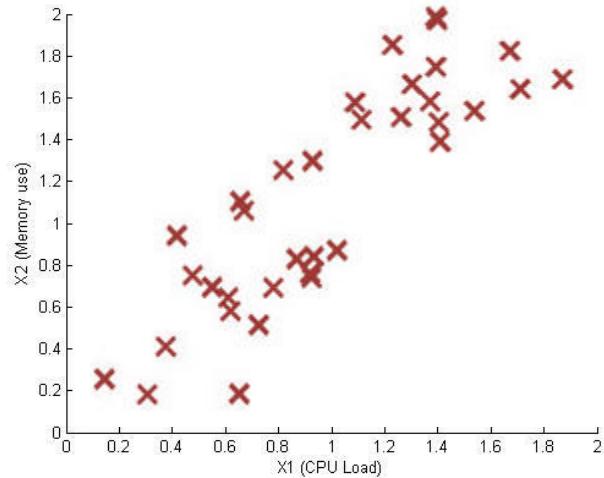
$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

2. Given a new example x , compute

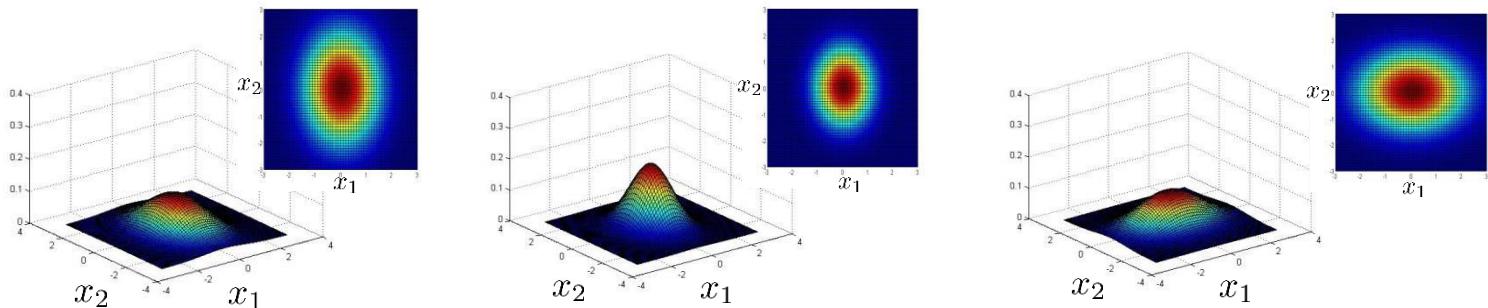
$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

Flag an anomaly if $p(x) < \varepsilon$



Relationship to original model

Original model: $p(x) = p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$



Corresponds to multivariate Gaussian

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

where

Original model

$$p(x_1; \mu_1, \sigma_1^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$$

Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values.

Computationally cheaper (alternatively, scales better to large n)

OK even if m (training set size) is small

vs.

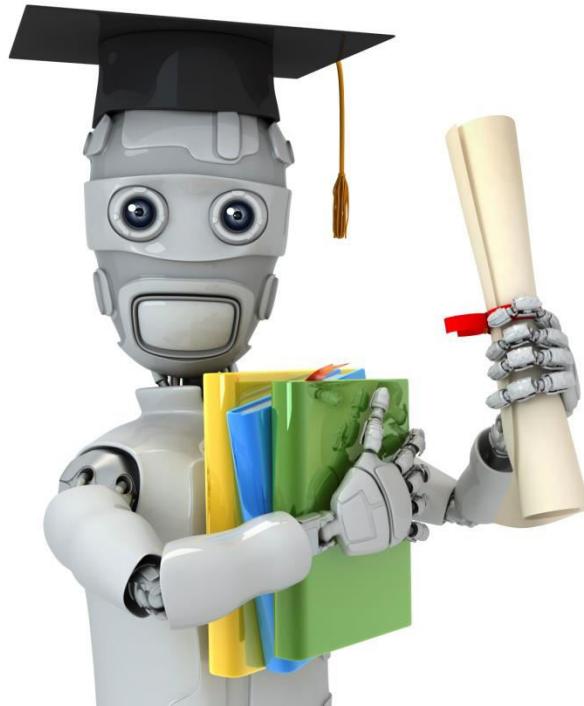
Multivariate Gaussian

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

Automatically captures correlations between features

Computationally more expensive

Must have $m > n$, or else Σ is non-invertible.



Machine Learning

Recommender Systems

Problem formulation

Example: Predicting movie ratings

User rates movies using one to five stars



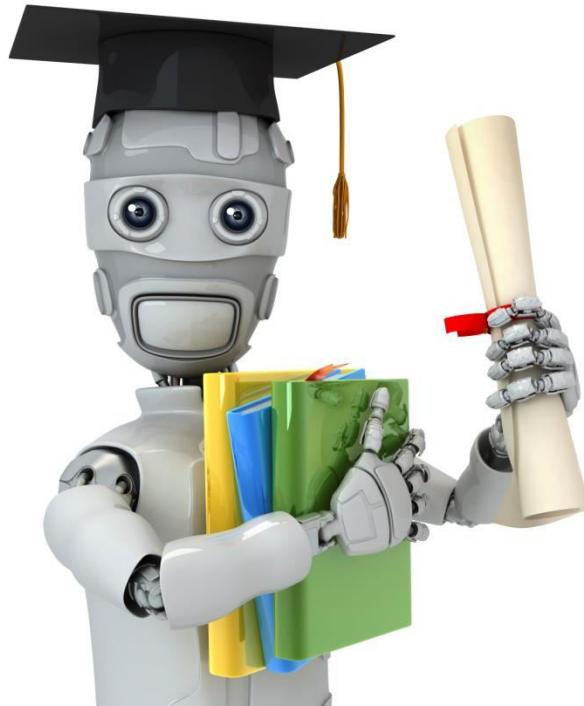
Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last				
Romance forever				
Cute puppies of love				
Nonstop car chases				
Swords vs. karate				

n_u = no. users

n_m = no. movies

$r(i, j) = 1$ if user j has
rated movie i

$y^{(i,j)}$ = rating given by
user j to movie i
(defined only if
 $r(i, j) = 1$)



Machine Learning

Recommender Systems

Content-based
recommendations

Content-based recommender systems

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

For each user j , learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars.

Problem formulation

$r(i, j) = 1$ if user j has rated movie i (0 otherwise)

$y^{(i,j)}$ = rating by user j on movie i (if defined)

$\theta^{(j)}$ = parameter vector for user j

$x^{(i)}$ = feature vector for movie i

For user j , movie i , predicted rating: $(\theta^{(j)})^T(x^{(i)})$

$m^{(j)}$ = no. of movies rated by user j

To learn $\theta^{(j)}$:

Optimization objective:

To learn $\theta^{(j)}$ (parameter for user j):

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^n (\theta_k^{(j)})^2$$

To learn $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

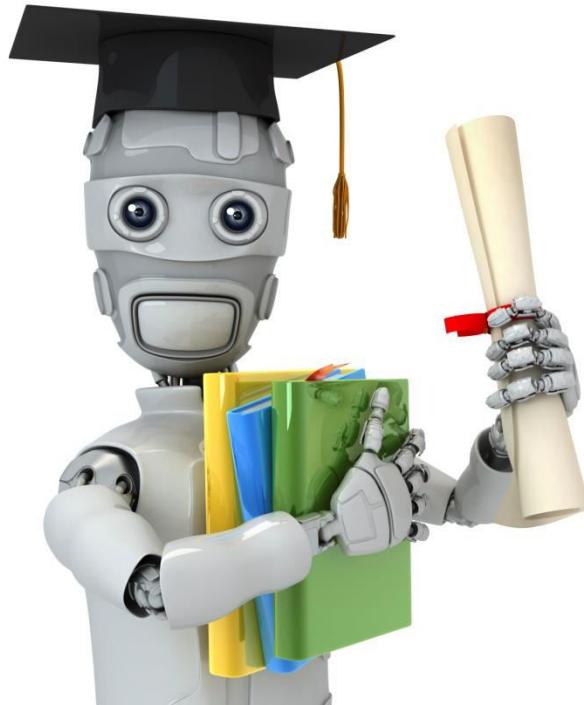
Optimization algorithm:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} \text{ (for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \text{ (for } k \neq 0)$$



Machine Learning

Recommender Systems

Collaborative filtering

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs. karate	0	0	5	?	?	?

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

Optimization algorithm

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^n (x_k^{(i)})^2$$

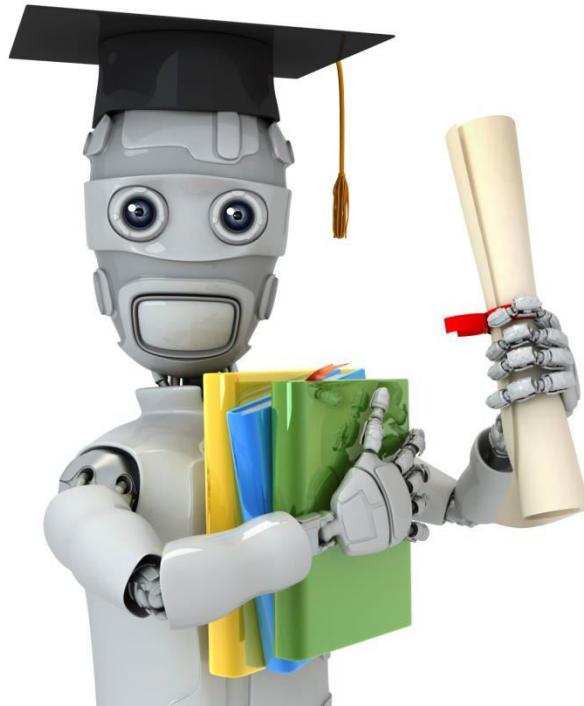
Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, to learn $x^{(1)}, \dots, x^{(n_m)}$:

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

Collaborative filtering

Given $x^{(1)}, \dots, x^{(n_m)}$ (and movie ratings),
can estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$,
can estimate $x^{(1)}, \dots, x^{(n_m)}$



Machine Learning

Recommender Systems

Collaborative
filtering algorithm

Collaborative filtering optimization objective

Given $x^{(1)}, \dots, x^{(n_m)}$, estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$, estimate $x^{(1)}, \dots, x^{(n_m)}$:

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

Minimizing $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$ simultaneously:

$$J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

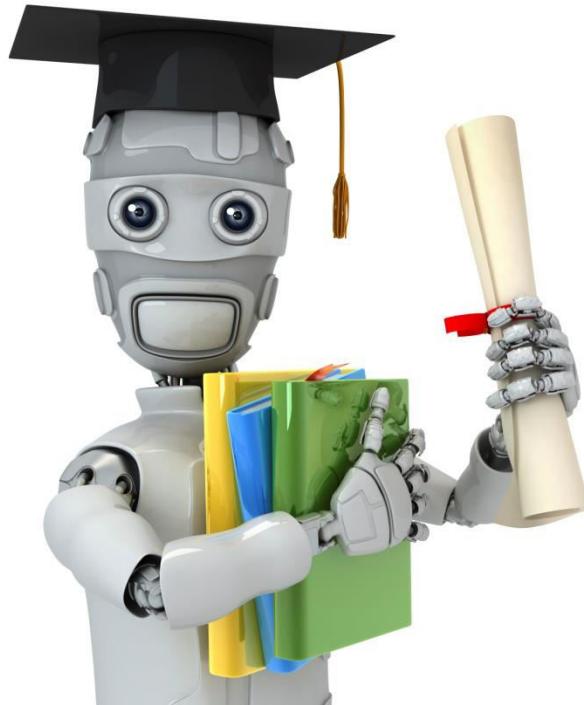
$$\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$$

Collaborative filtering algorithm

1. Initialize $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \dots, n_u, i = 1, \dots, n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$
$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

3. For a user with parameters θ and a movie with (learned) features x , predict a star rating of $\theta^T x$.



Machine Learning

Recommender Systems

Vectorization:
Low rank matrix
factorization

Collaborative filtering

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	?	?	0
Cute puppies of love	?	4	0	?
Nonstop car chases	0	0	5	4
Swords vs. karate	0	0	5	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Collaborative filtering

Predicted ratings:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix} \quad \begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \dots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \dots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \dots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}$$

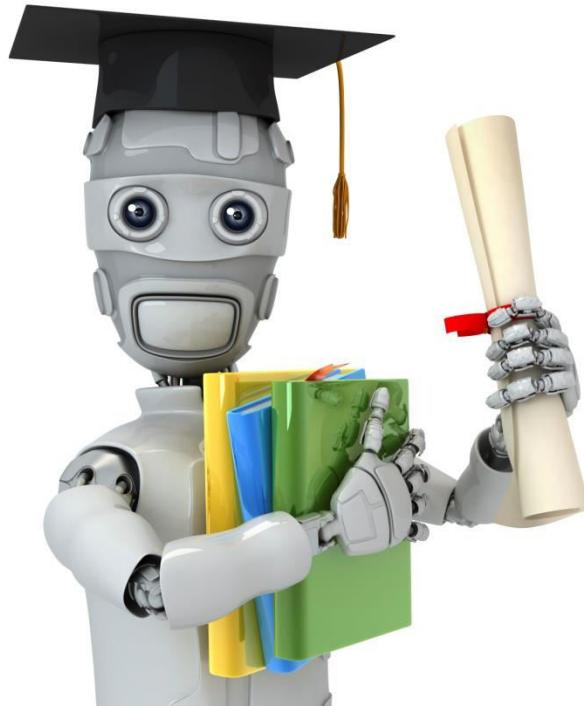
Finding related movies

For each product i , we learn a feature vector $x^{(i)} \in \mathbb{R}^n$.

How to find movies j related to movie i ?

5 most similar movies to movie i :

Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.



Machine Learning

Recommender Systems

Implementational detail: Mean normalization

Users who have not rated any movies

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)	
Love at last	5	5	0	0	?	$\begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \end{bmatrix}$
Romance forever	5	?	?	0	?	$\begin{bmatrix} ? & 4 & 0 & ? & ? \end{bmatrix}$
Cute puppies of love	?	4	0	?	?	$\begin{bmatrix} 0 & 0 & 5 & 4 & ? \end{bmatrix}$
Nonstop car chases	0	0	5	4	?	$\begin{bmatrix} 0 & 0 & 5 & 0 & ? \end{bmatrix}$
Swords vs. karate	0	0	5	?	?	

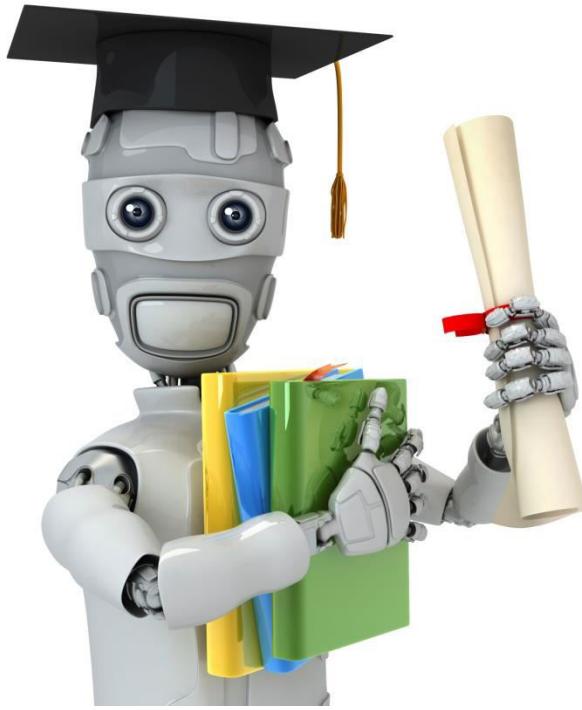
$$\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} \frac{1}{2} \sum_{(i,j): r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (\theta_k^{(j)})^2$$

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \quad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j , on movie i predict:

User 5 (Eve):



Machine Learning

Large scale
machine learning

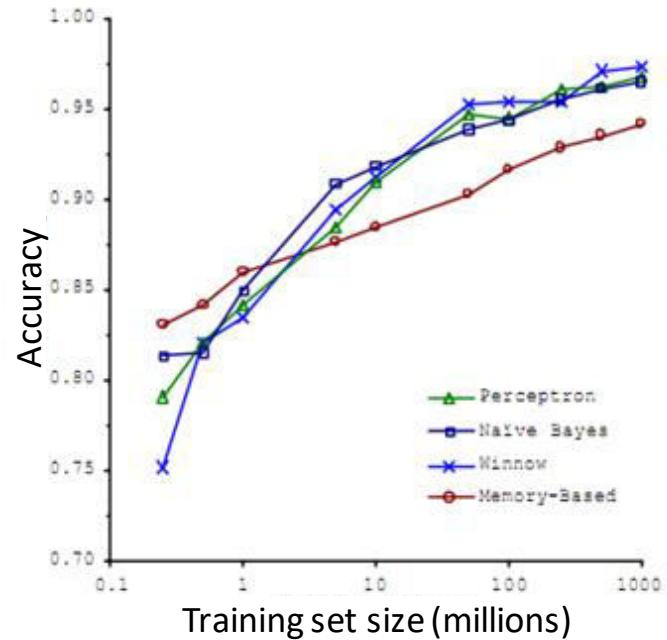
Learning with
large datasets

Machine learning and data

Classify between confusable words.

E.g., {to, two, too}, {then, than}.

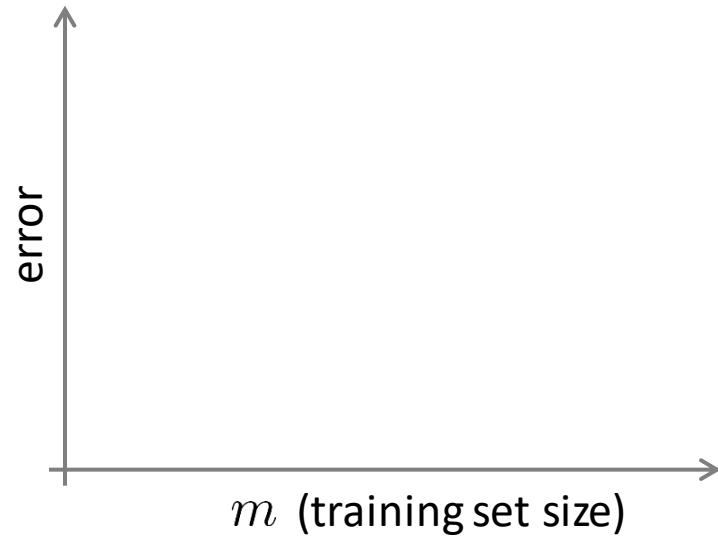
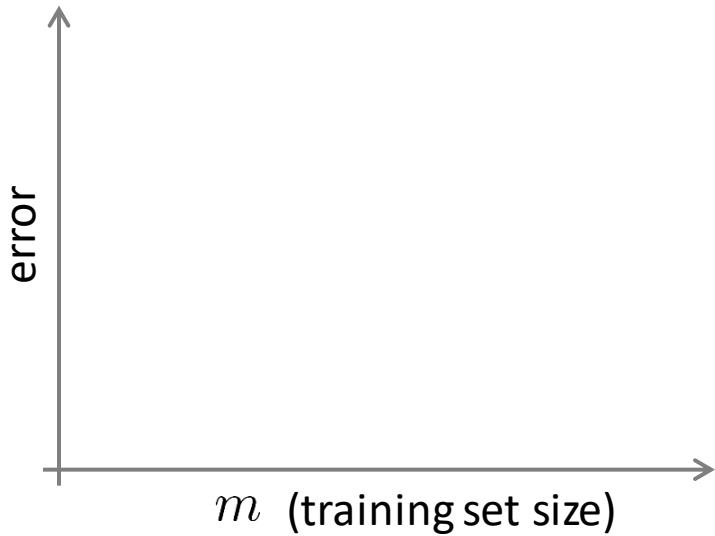
For breakfast I ate _____ eggs.

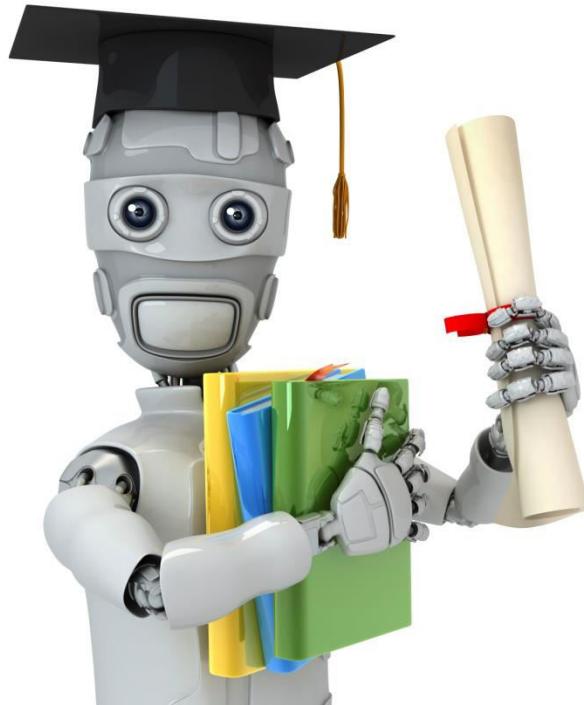


“It’s not who has the best algorithm that wins.
It’s who has the most data.”

Learning with large datasets

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





Machine Learning

Large scale
machine learning

Stochastic
gradient descent

Linear regression with gradient descent

$$h_{\theta}(x) = \sum_{j=0}^n \theta_j x_j$$

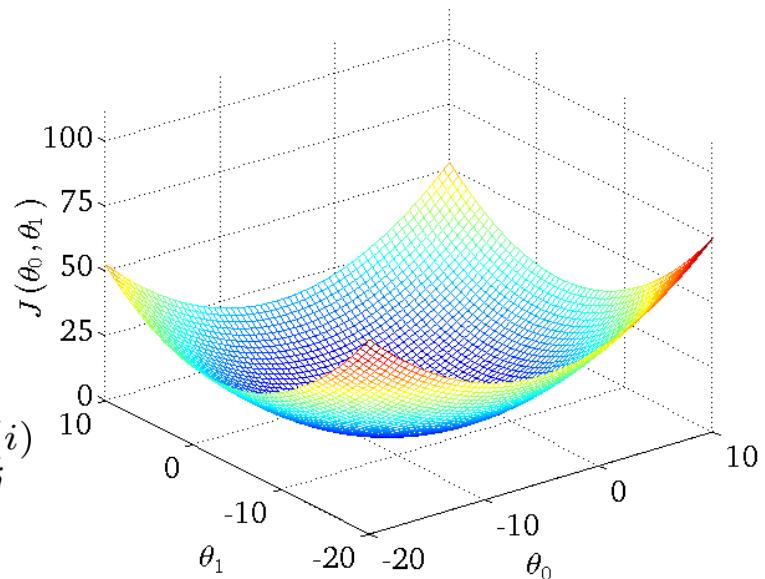
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Repeat {

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

(for every $j = 0, \dots, n$)

}



Linear regression with gradient descent

$$h_{\theta}(x) = \sum_{j=0}^n \theta_j x_j$$

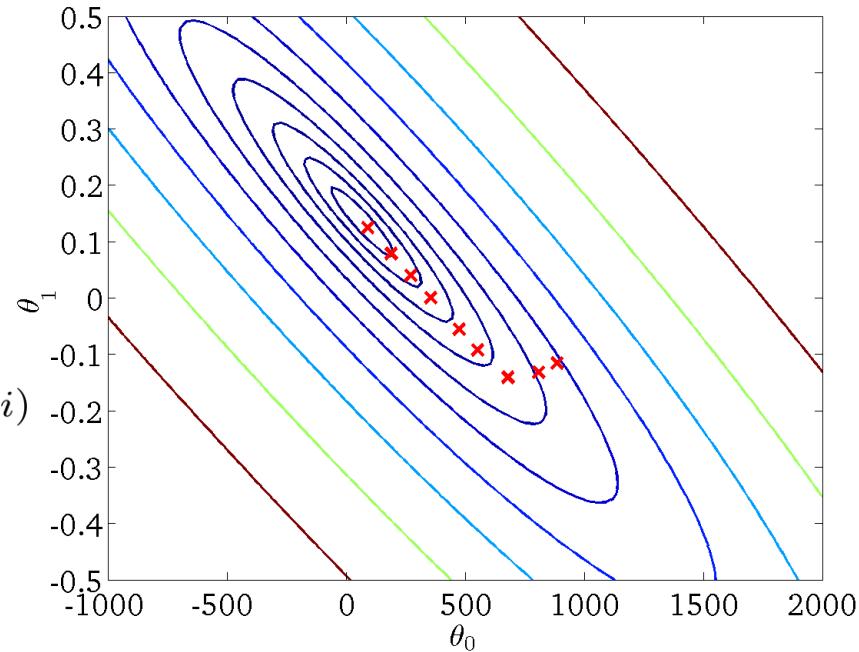
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Repeat {

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

(for every $j = 0, \dots, n$)

}



Batch gradient descent

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

Repeat {

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

(for every $j = 0, \dots, n$)

}

Stochastic gradient descent

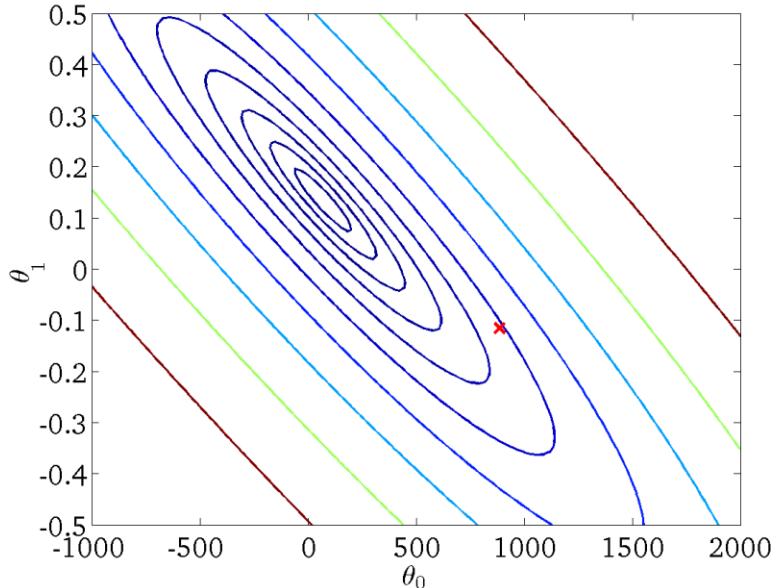
$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

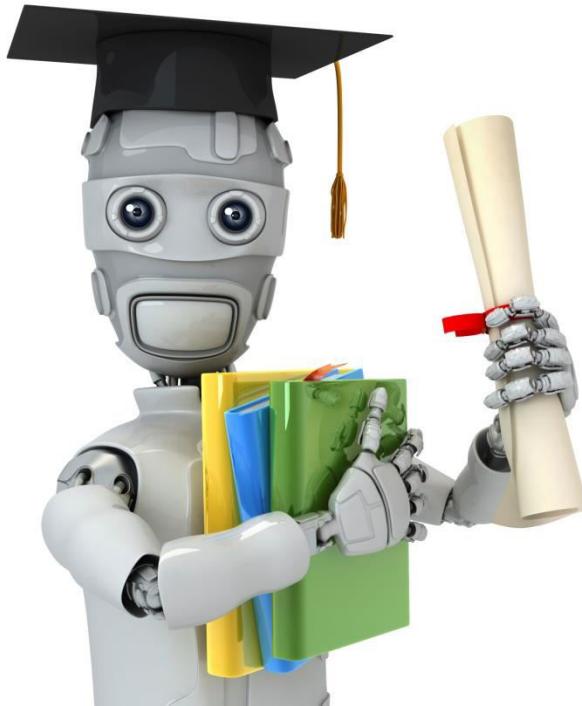
$$J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$$

Stochastic gradient descent

1. Randomly shuffle (reorder) training examples

2. Repeat {
 for $i := 1, \dots, m$ {
 $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
 (for every $j = 0, \dots, n$)
 }
}
}





Machine Learning

Large scale
machine learning

Mini-batch
gradient descent

Mini-batch gradient descent

Batch gradient descent: Use all m examples in each iteration

Stochastic gradient descent: Use 1 example in each iteration

Mini-batch gradient descent: Use b examples in each iteration

Mini-batch gradient descent

Say $b = 10, m = 1000$.

Repeat {

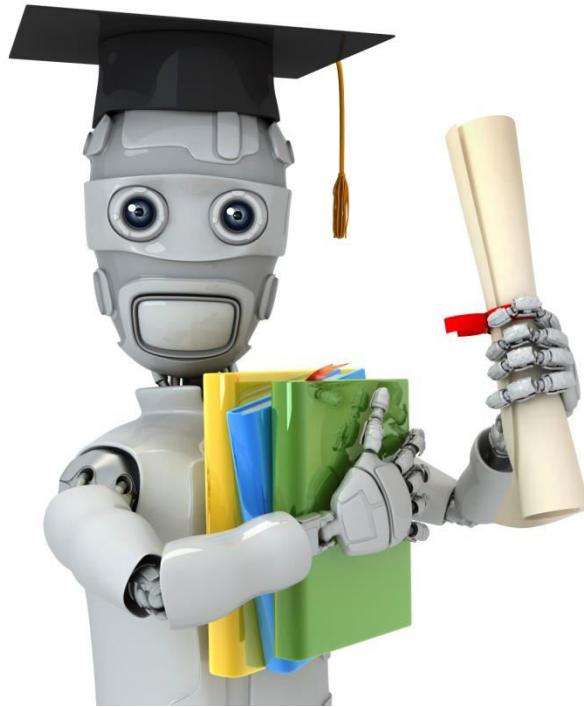
for $i = 1, 11, 21, 31, \dots, 991$ {

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

(for every $j = 0, \dots, n$)

}

}



Machine Learning

Large scale
machine learning

Stochastic
gradient descent
convergence

Checking for convergence

Batch gradient descent:

Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.

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

Stochastic gradient descent:

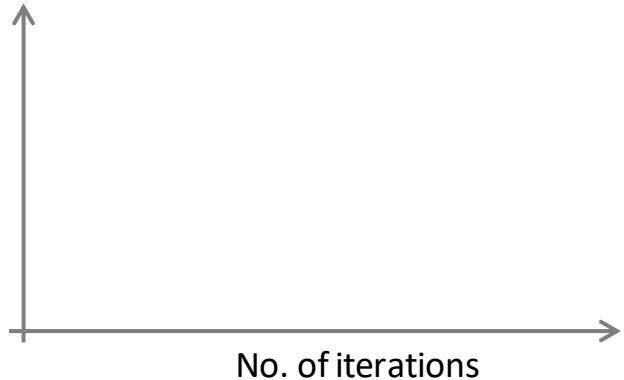
$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

During learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.

Every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.

Checking for convergence

Plot $\text{cost}(\theta, (x^{(i)}, y^{(i)}))$, averaged over the last 1000 (say) examples



Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.

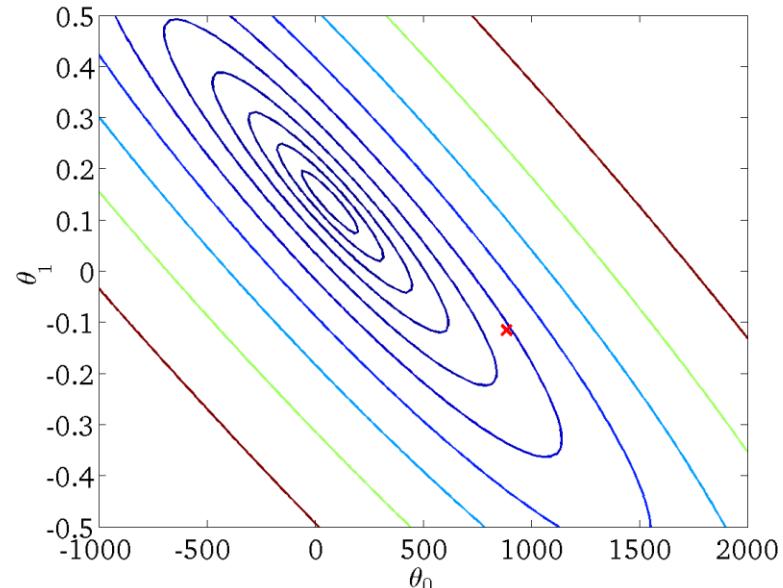
2. Repeat {

```
for i := 1, ..., m      {
    theta_j := theta_j - alpha(h_theta(x^{(i)}) - y^{(i)})x_j^{(i)}
```

```
    (for j = 0, ..., n)
```

```
}
```

```
}
```



Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge. (E.g. $\alpha = \frac{\text{const1}}{\text{iterationNumber} + \text{const2}}$)

Stochastic gradient descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.

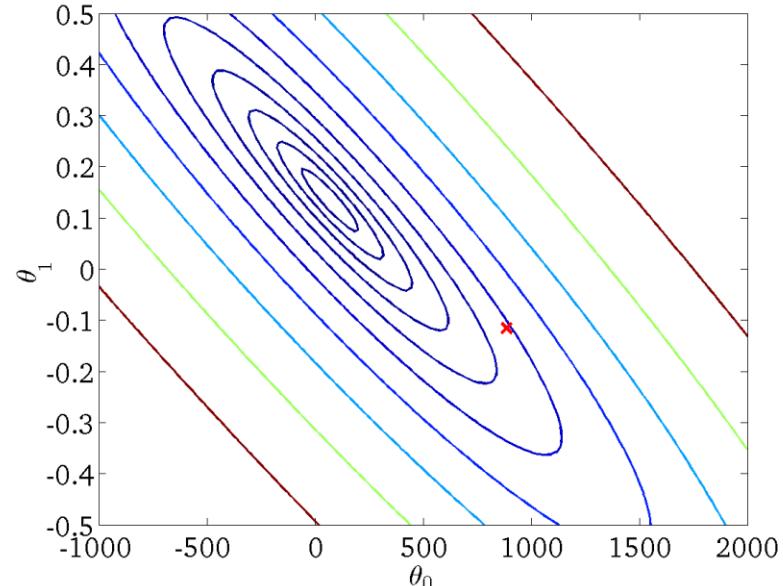
2. Repeat {

```
for i := 1, ..., m      {
    theta_j := theta_j - alpha(h_theta(x^{(i)}) - y^{(i)})x_j^{(i)}
```

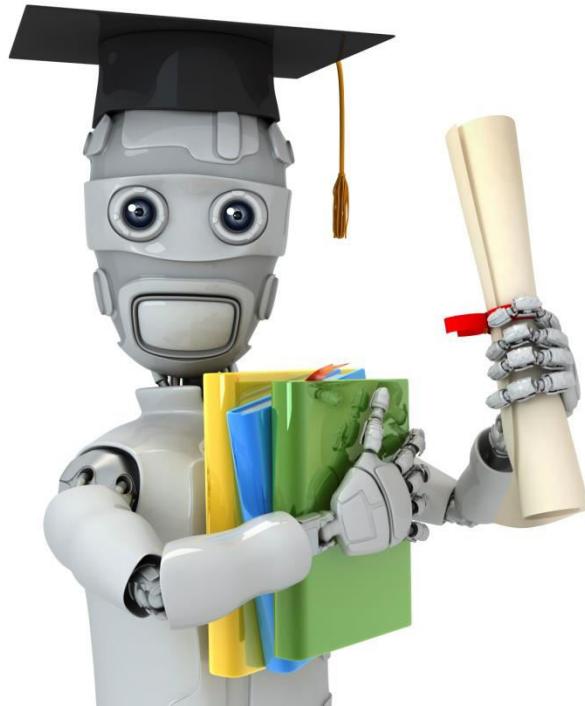
```
    (for j = 0, ..., n)
```

```
}
```

```
}
```



Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge. (E.g. $\alpha = \frac{\text{const1}}{\text{iterationNumber} + \text{const2}}$)



Machine Learning

Large scale
machine learning

Online learning

Online learning

Shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, and users sometimes choose to use your shipping service ($y = 1$), sometimes not ($y = 0$).

Features x capture properties of user, of origin/destination and asking price. We want to learn $p(y = 1|x; \theta)$ to optimize price.

Other online learning example:

Product search (learning to search)

User searches for “Android phone 1080p camera”

Have 100 phones in store. Will return 10 results.

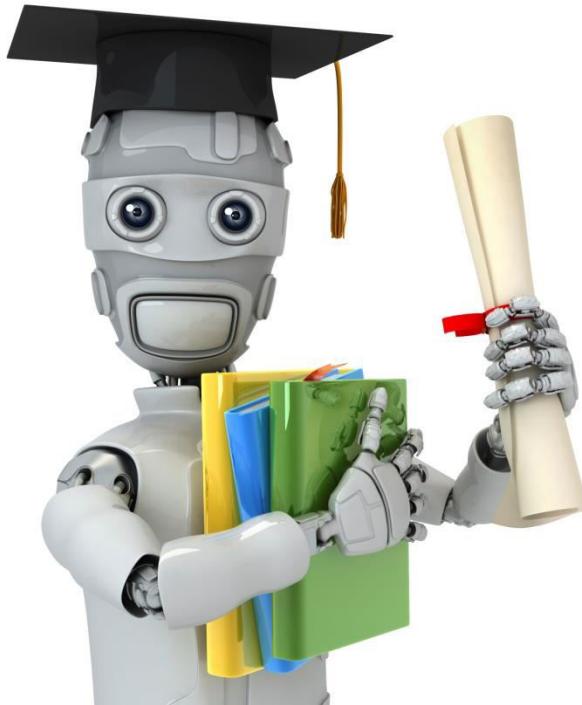
x = features of phone, how many words in user query match name of phone, how many words in query match description of phone, etc.

$y = 1$ if user clicks on link. $y = 0$ otherwise.

Learn $p(y = 1|x; \theta)$.

Use to show user the 10 phones they’re most likely to click on.

Other examples: Choosing special offers to show user; customized selection of news articles; product recommendation; ...



Machine Learning

Large scale machine learning

Map-reduce and data parallelism

Map-reduce

Batch gradient descent: $\theta_j := \theta_j - \alpha \frac{1}{400} \sum_{i=1}^{400} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$

Machine 1: Use $(x^{(1)}, y^{(1)}), \dots, (x^{(100)}, y^{(100)})$.

Machine 2: Use $(x^{(101)}, y^{(101)}), \dots, (x^{(200)}, y^{(200)})$.

$$temp_j^{(2)} = \sum_{i=101}^{200} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

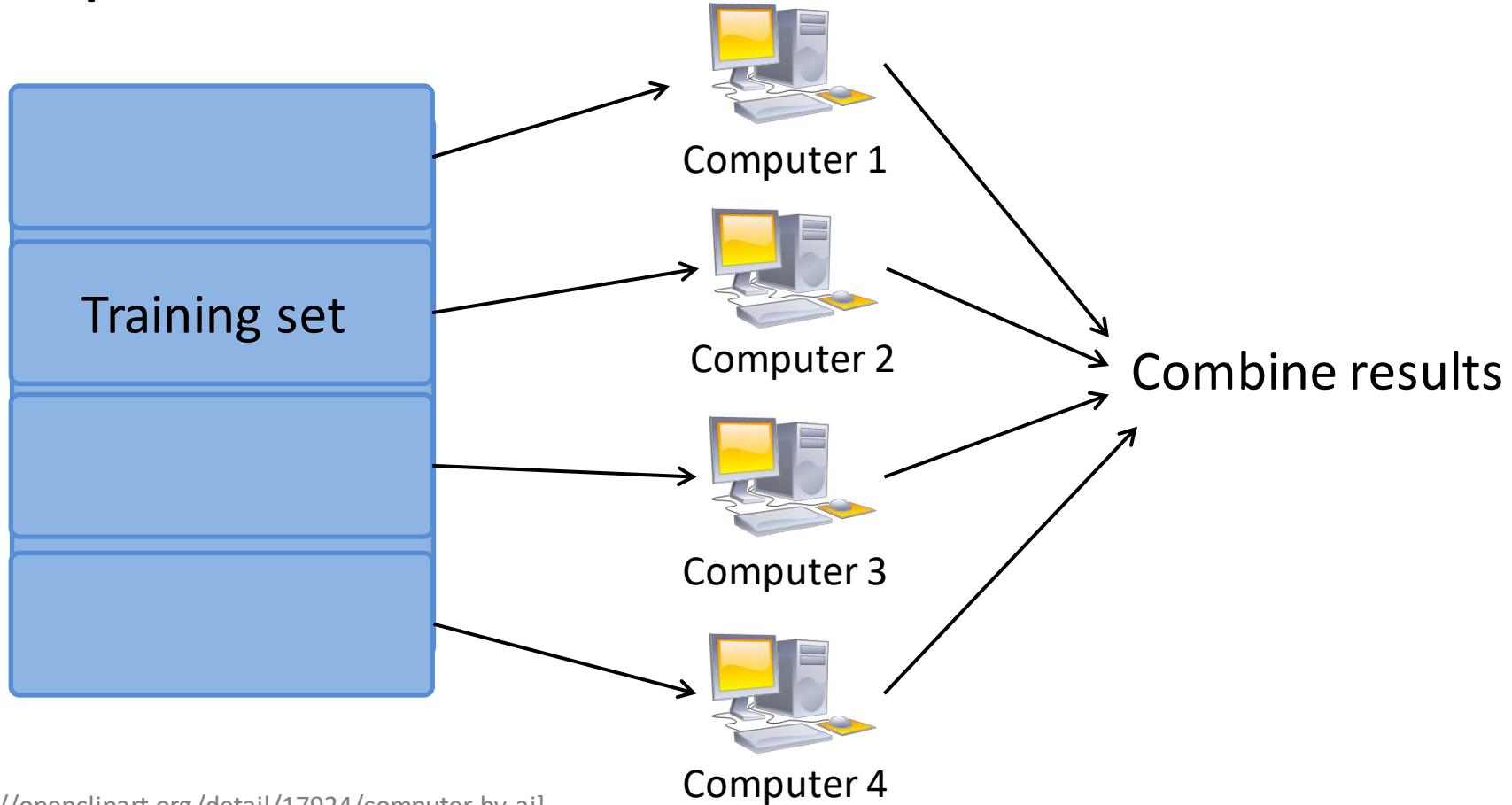
Machine 3: Use $(x^{(201)}, y^{(201)}), \dots, (x^{(300)}, y^{(300)})$.

$$temp_j^{(3)} = \sum_{i=201}^{300} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

Machine 4: Use $(x^{(301)}, y^{(301)}), \dots, (x^{(400)}, y^{(400)})$.

$$temp_j^{(4)} = \sum_{i=301}^{400} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

Map-reduce



Map-reduce and summation over the training set

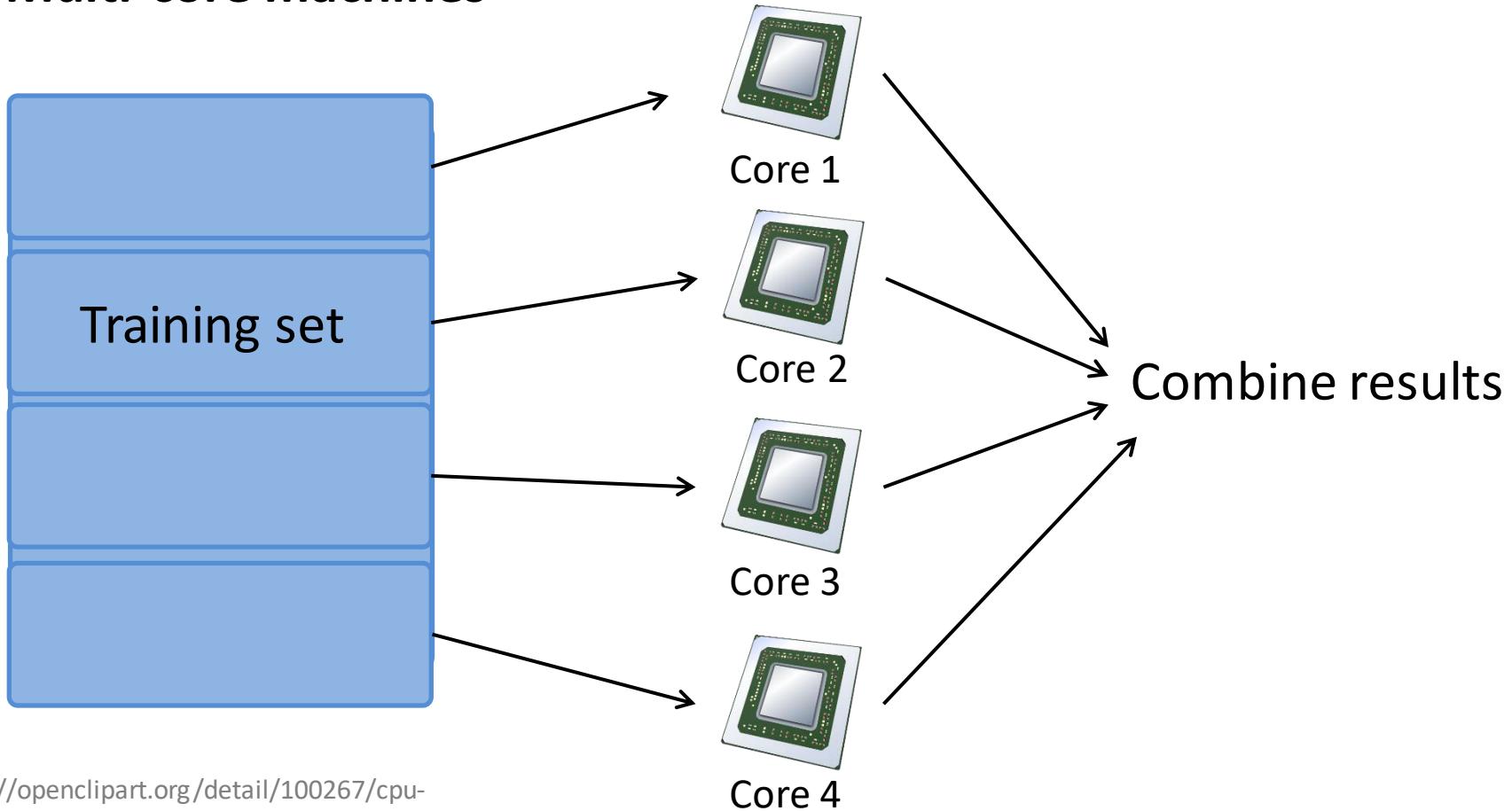
Many learning algorithms can be expressed as computing sums of functions over the training set.

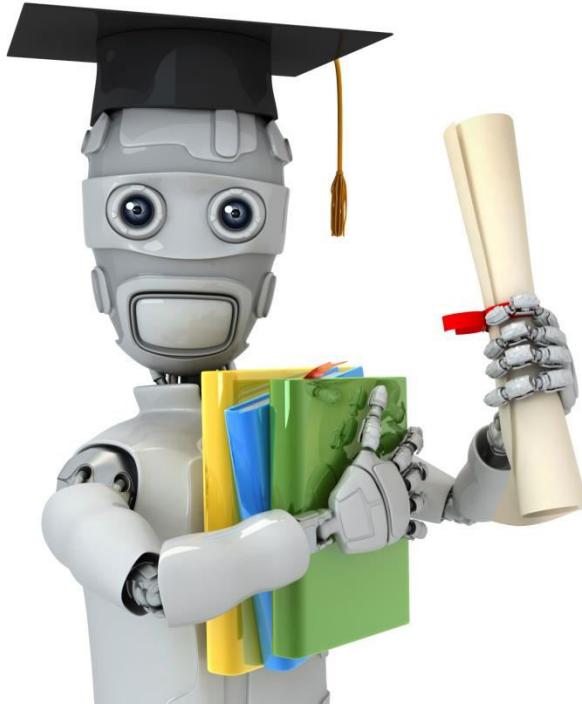
E.g. for advanced optimization, with logistic regression, need:

$$J_{train}(\theta) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log h_\theta(x^{(i)}) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

$$\frac{\partial}{\partial \theta_j} J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

Multi-core machines





Machine Learning

Application example: Photo OCR

Problem description and pipeline

The Photo OCR problem



Photo OCR pipeline

1. Text detection



2. Character segmentation



3. Character classification

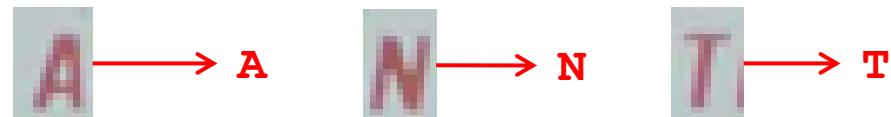
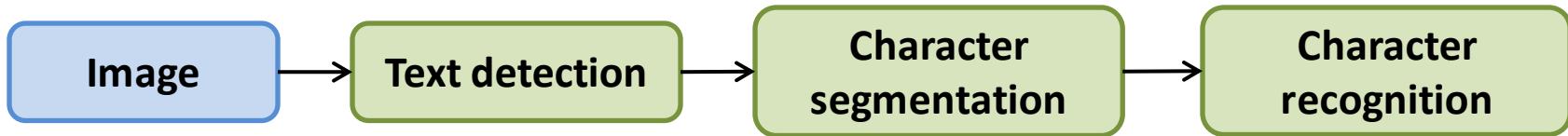
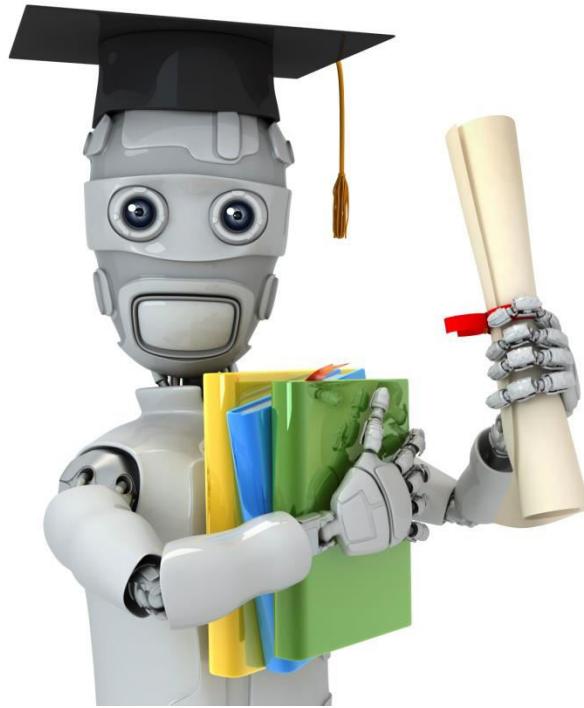


Photo OCR pipeline





Machine Learning

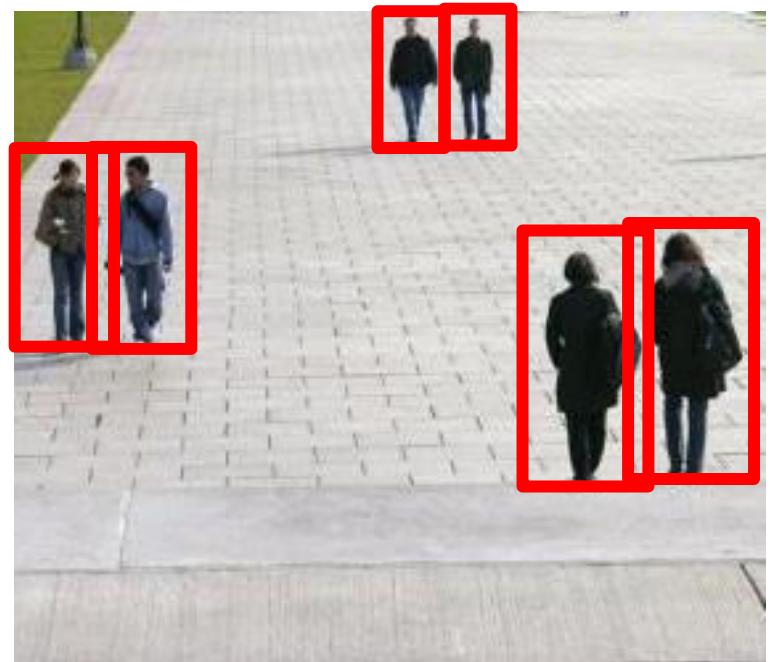
Application example:
Photo OCR

Sliding windows

Text detection



Pedestrian detection



Supervised learning for pedestrian detection

x = pixels in 82x36 image patches



Positive examples ($y = 1$)



Negative examples ($y = 0$)

Sliding window detection



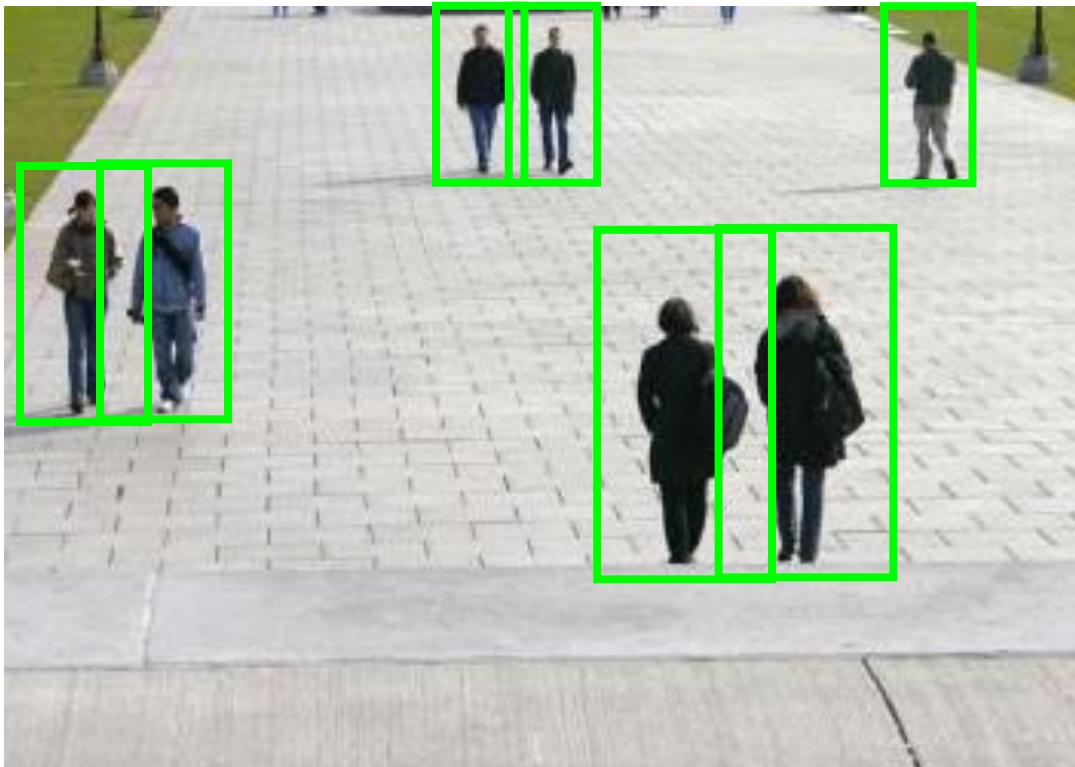
Sliding window detection



Sliding window detection



Sliding window detection



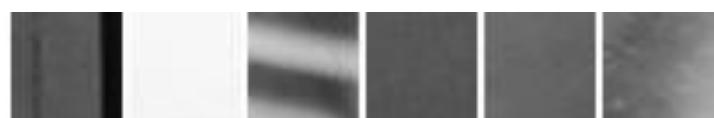
Text detection



Text detection

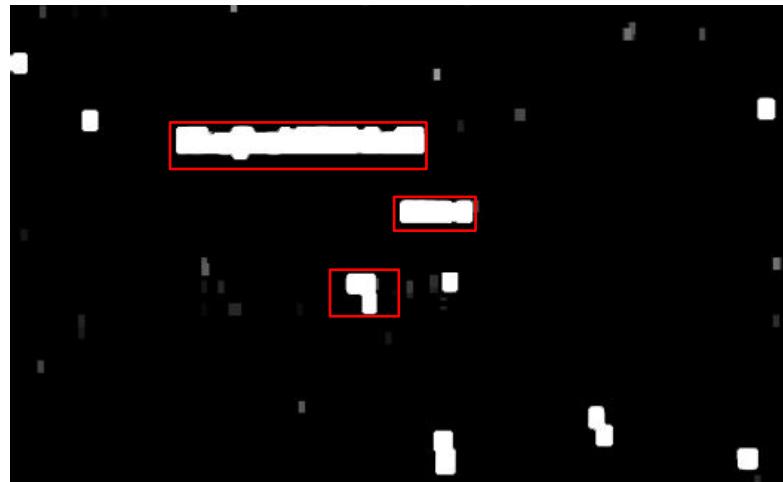


Positive examples ($y = 1$)



Negative examples ($y = 0$)

Text detection



1D Sliding window for character segmentation



Positive examples ($y = 1$)

Negative examples ($y = 0$)

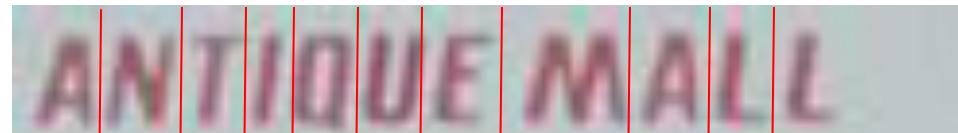


Photo OCR pipeline

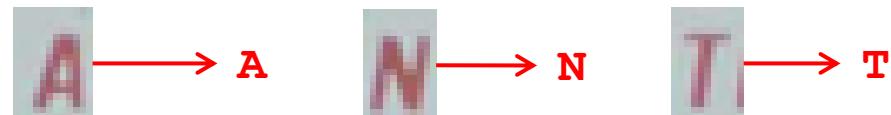
1. Text detection

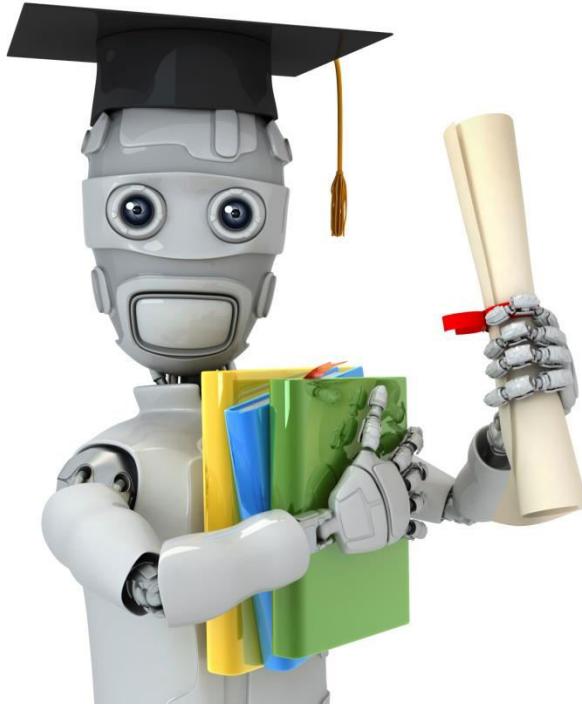


2. Character segmentation



3. Character classification





Machine Learning

Application example: Photo OCR

Getting lots of data: Artificial data synthesis

Character recognition



→ A



→ N



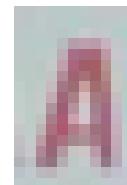
→ T



→ I



→ O



→ A

Artificial data synthesis for photo OCR



Real data

Abcdefg

Abcdefg

Abcdefg

Abcdefg

Abcdefg

Artificial data synthesis for photo OCR

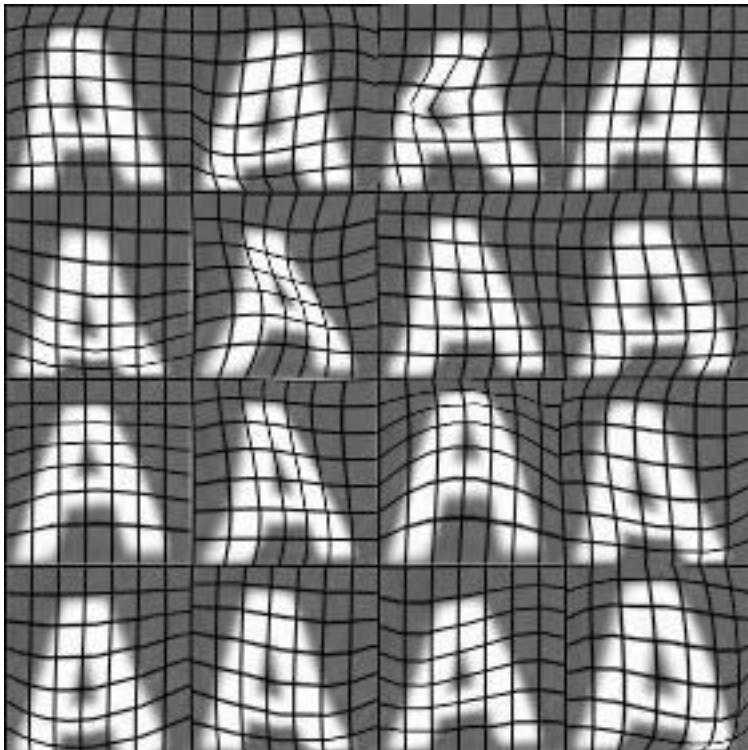
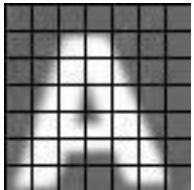


Real data



Synthetic data

Synthesizing data by introducing distortions



Synthesizing data by introducing distortions: Speech recognition



Original audio:



Audio on bad cellphone connection



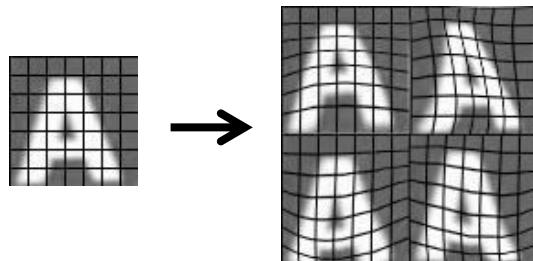
Noisy background: Crowd



Noisy background: Machinery

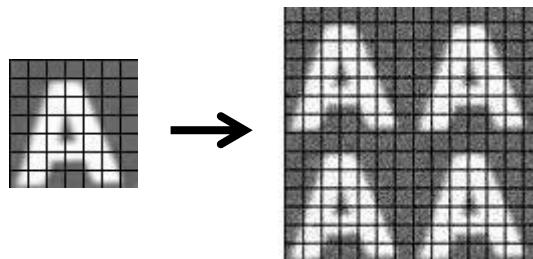
Synthesizing data by introducing distortions

Distortion introduced should be representation of the type of noise/distortions in the test set.



Audio:
Background noise,
bad cellphone connection

Usually does not help to add purely random/meaningless noise to your data.

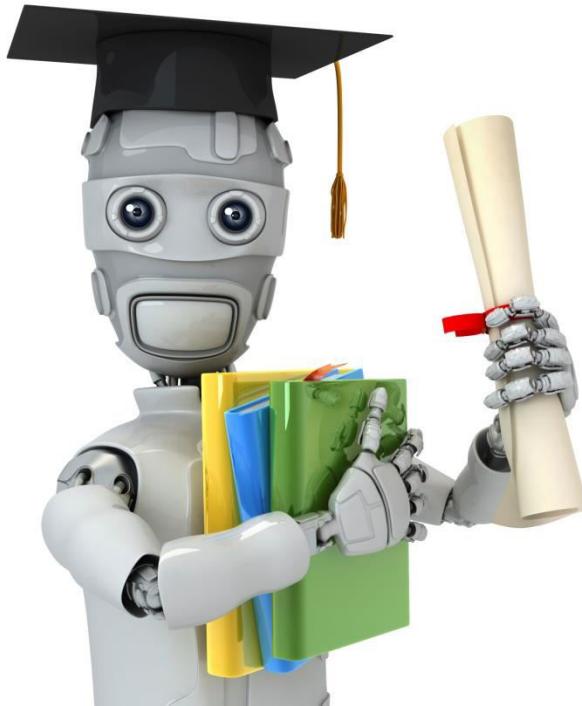

$$x_i = \text{intensity (brightness) of pixel } i$$
$$x_i \leftarrow x_i + \text{random noise}$$

Discussion on getting more data

1. Make sure you have a low bias classifier before expending the effort. (Plot learning curves). E.g. keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier.
2. “How much work would it be to get 10x as much data as we currently have?”
 - Artificial data synthesis
 - Collect/label it yourself
 - “Crowd source” (E.g. Amazon Mechanical Turk)

Discussion on getting more data

1. Make sure you have a low bias classifier before expending the effort. (Plot learning curves). E.g. keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier.
2. “How much work would it be to get 10x as much data as we currently have?”
 - Artificial data synthesis
 - Collect/label it yourself
 - “Crowd source” (E.g. Amazon Mechanical Turk)

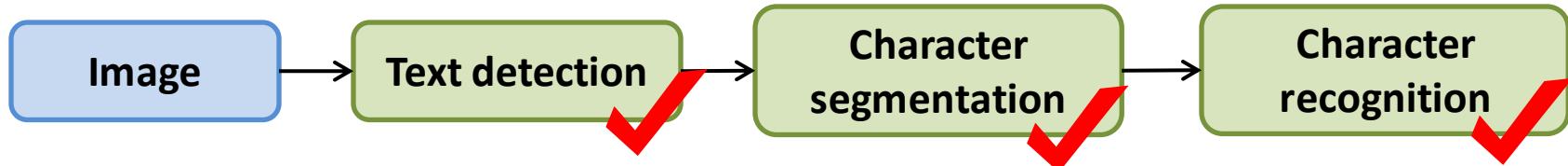


Machine Learning

Application example: Photo OCR

Ceiling analysis: What
part of the pipeline to
work on next

Estimating the errors due to each component (ceiling analysis)

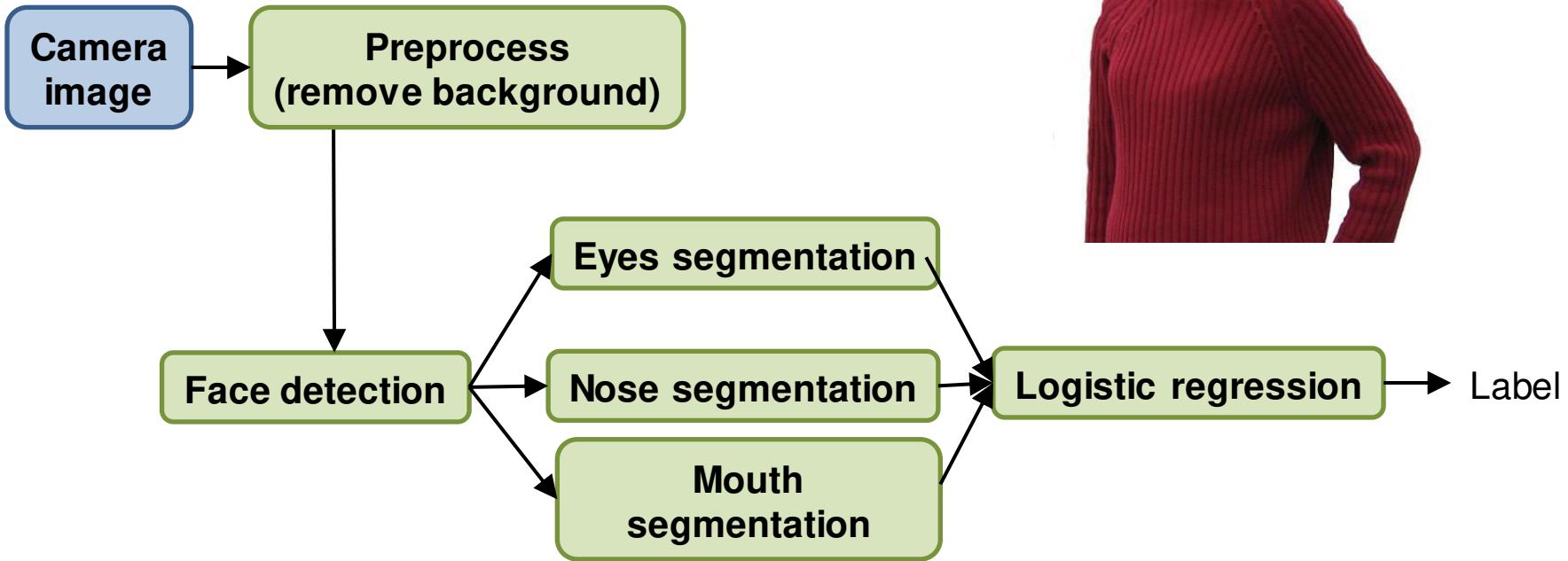


What part of the pipeline should you spend the most time trying to improve?

Component	Accuracy
Overall system	72%

Another ceiling analysis example

Face recognition from images
(Artificial example)



Another ceiling analysis example

