

Machine Learning's Slides

Author: Pannenets.F

Date: October 6, 2020

Je reviendrai et je serai des millions. — «Spartacus»

Contents

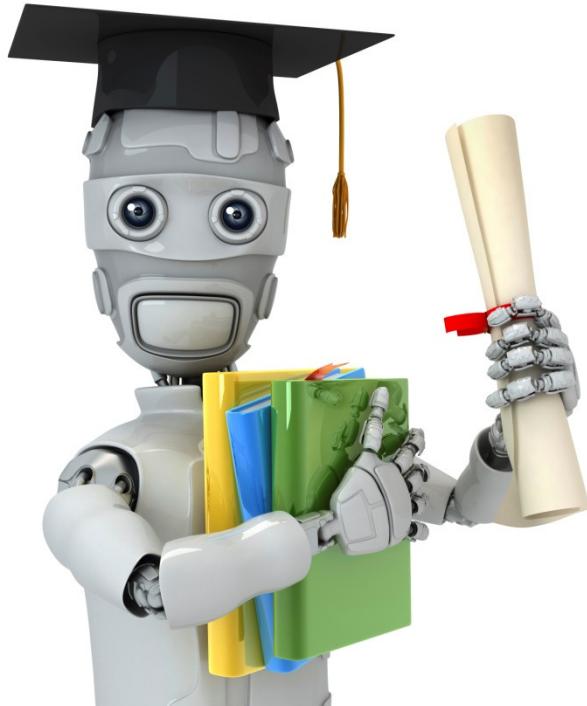
I part	1
1 Week1	2
1.1 Introduction	2
1.2 Gradient Descent	42
1.3 Linear Algebra	92
2 Week2	118
2.1 Multiple Features	118
2.2 Octave	150
3 Week3	161
3.1 Logistic Regression	161
3.2 Overfitting	193
4 Neural Networks	218
4.1 Non-linear Hypotheses	218
4.2 Learning	261
4.3 NN	295
5 SVM	344

Part I

part

Chapter 1 Week1

1.1 Introduction

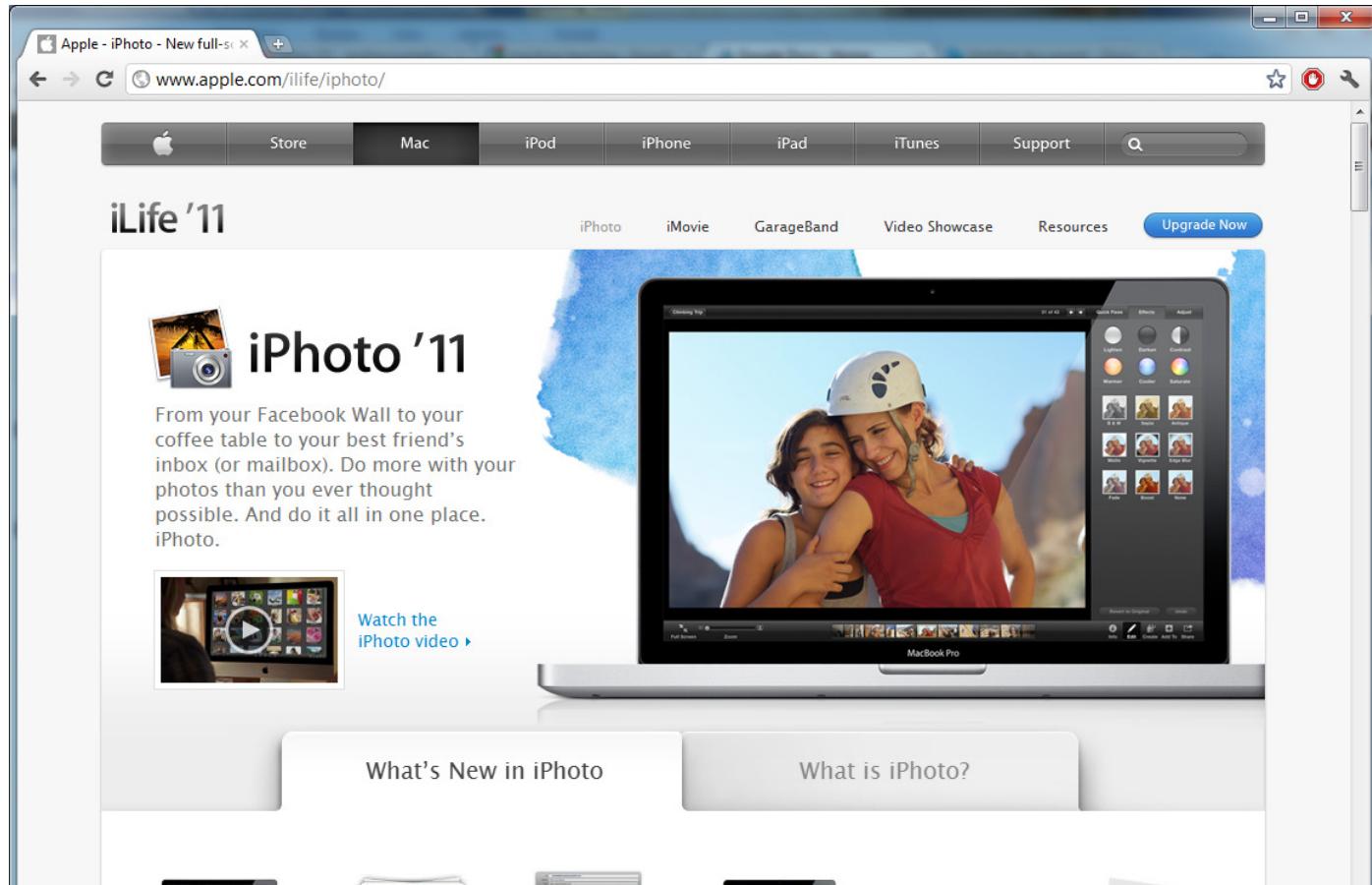


Machine Learning

Introduction

Welcome

Andrew Ng



Andrew Ng



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

Exam

-

-



ig

host of

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

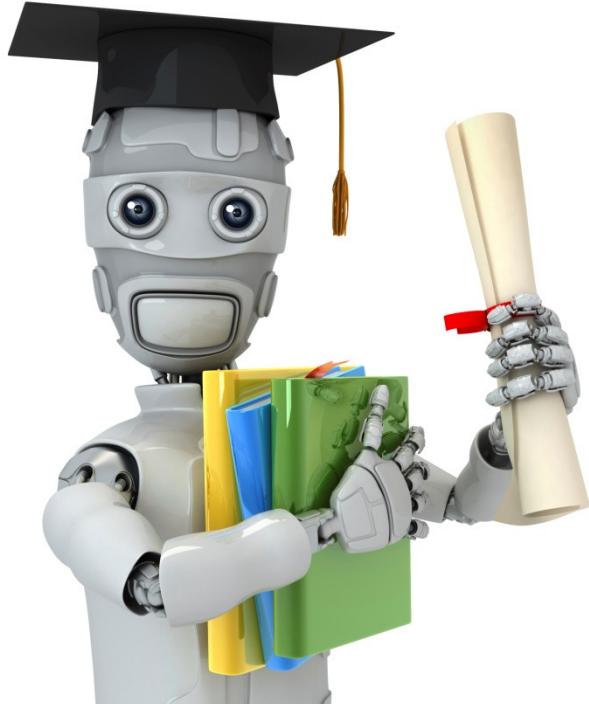
Machine Learning

- 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).

Andrew Ng



Machine Learning

Introduction

What is machine learning

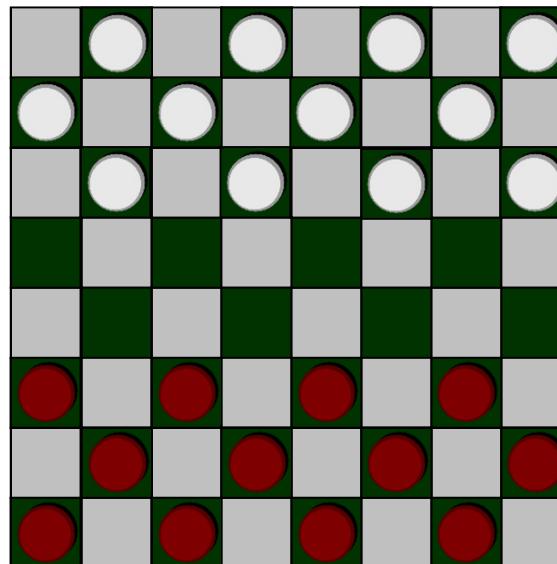
Machine Learning definition

Machine Learning definition

- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

Machine Learning definition

- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.



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.

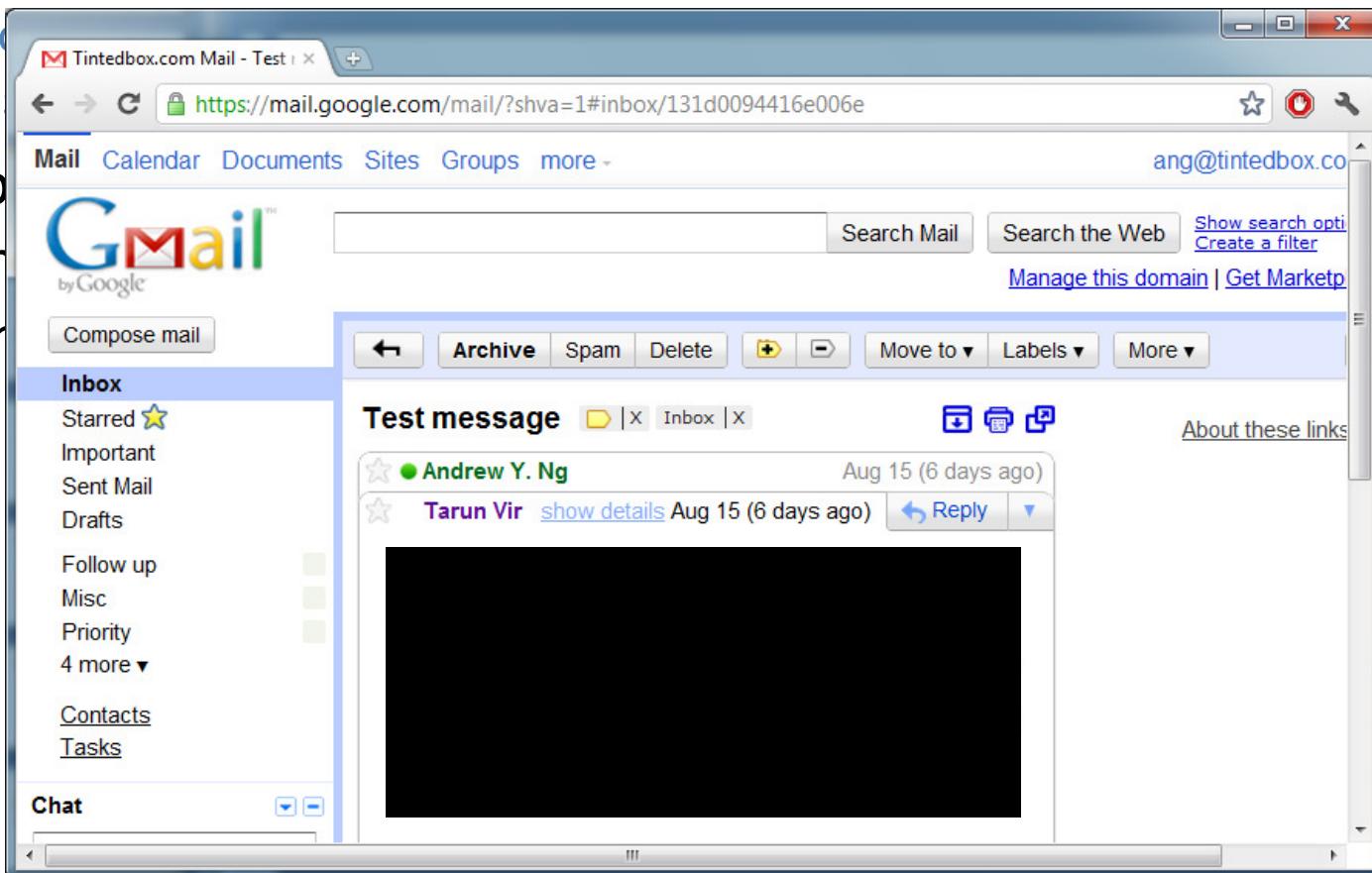
“A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

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

“A computer program is said to *learn* from experience E with respect to some class of tasks T if its performance improves with experience.”

Support
not n
spam



or do
filter

spam.

“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. $T \leftarrow$
- Watching you label emails as spam or not spam. $E \leftarrow$
- The number (or fraction) of emails correctly classified as spam/not spam.
- None of the above—this is not a machine learning problem. $P \leftarrow$

Machine learning algorithms:

- Supervised learning
- Unsupervised learning

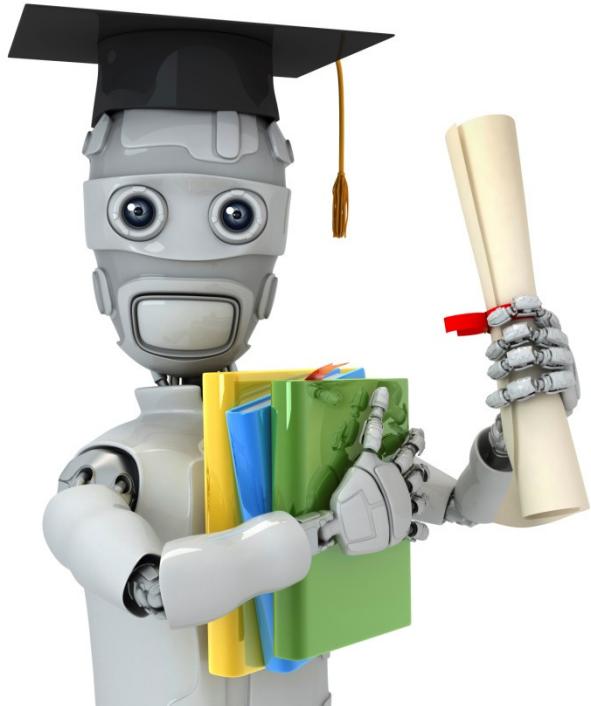


Others: Reinforcement learning, recommender systems.

Also talk about: Practical advice for applying learning algorithms.



Andrew Ng

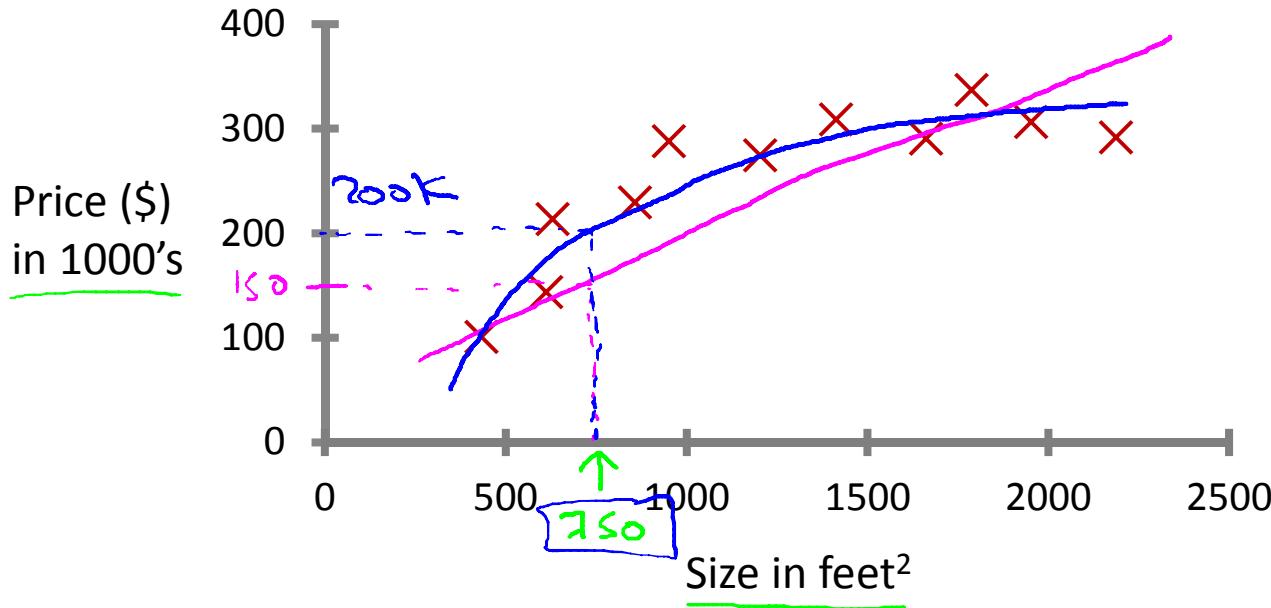


Machine Learning

Introduction

Supervised Learning

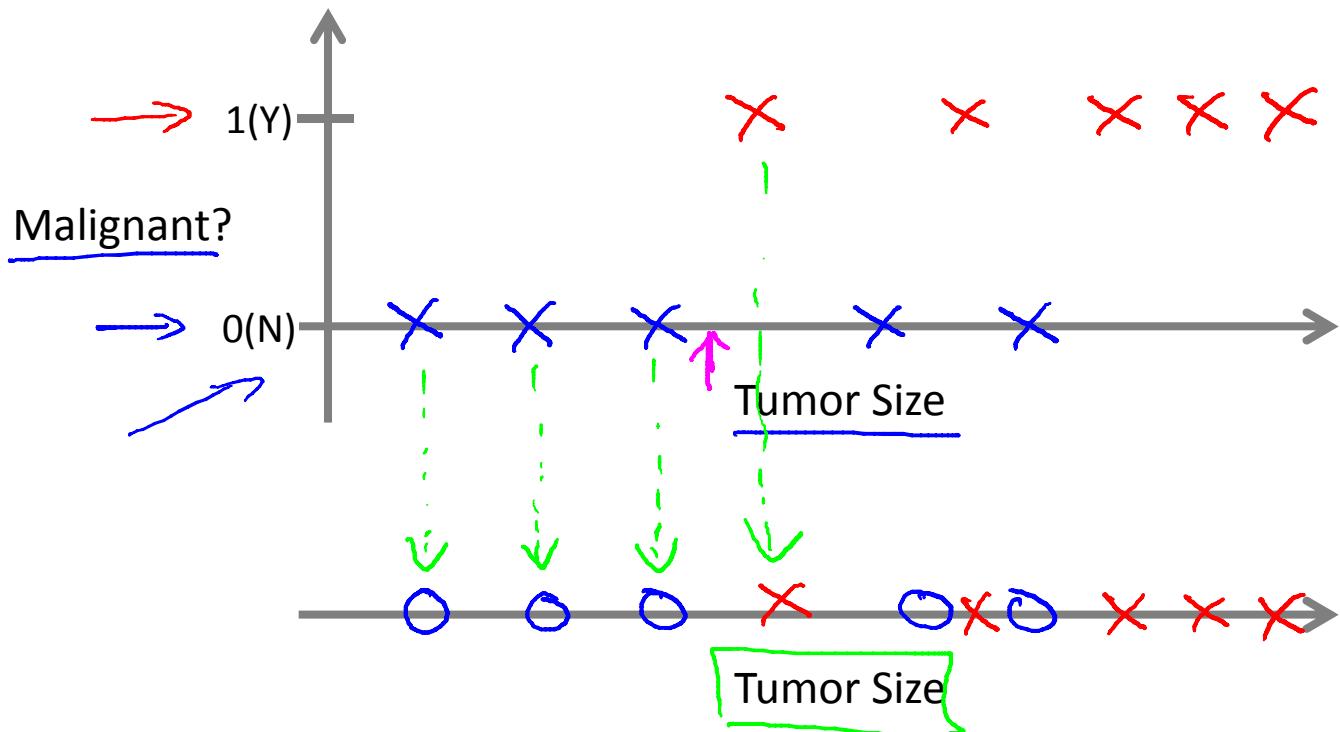
Housing price prediction.



Supervised Learning
‘right answers’ given

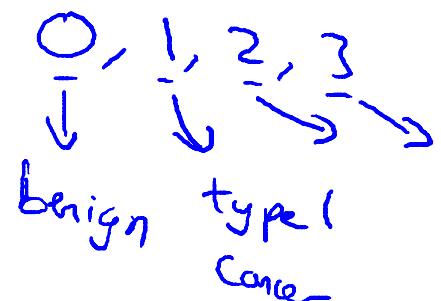
Regression: Predict continuous valued output (price)

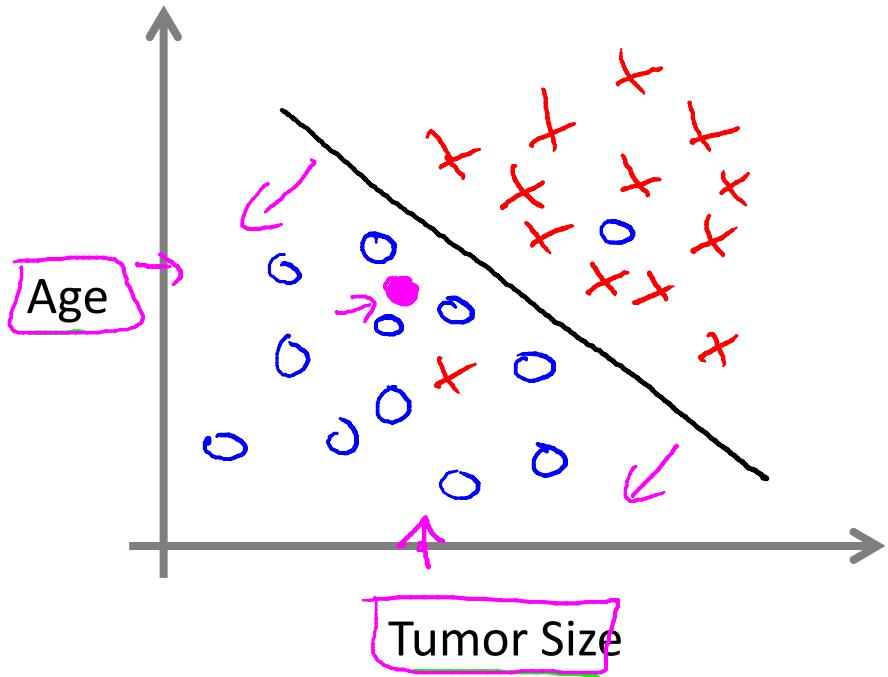
Breast cancer (malignant, benign)



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.

1000's

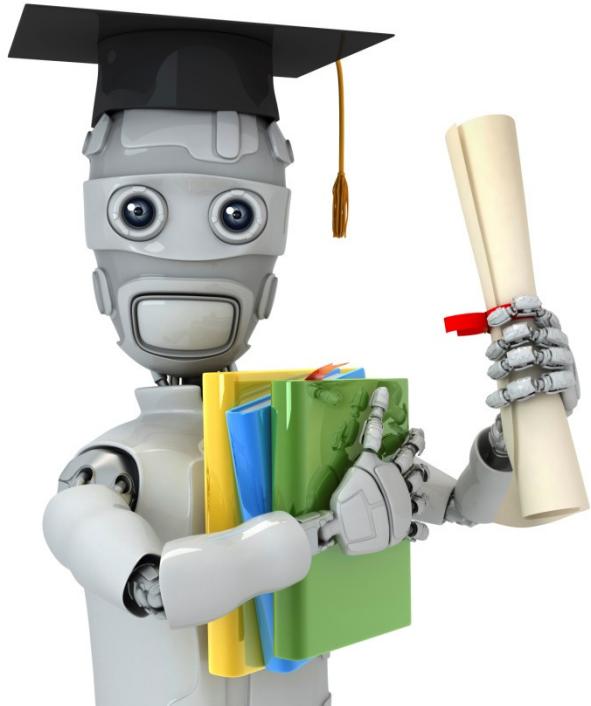
- Problem 1: You have a large inventory of identical items. You want to predict how many of these items will sell over the next 3 months.
- Problem 2: You'd like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised.

→ 0 - not hacked
→ 1 - hacked

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.

Andrew Ng

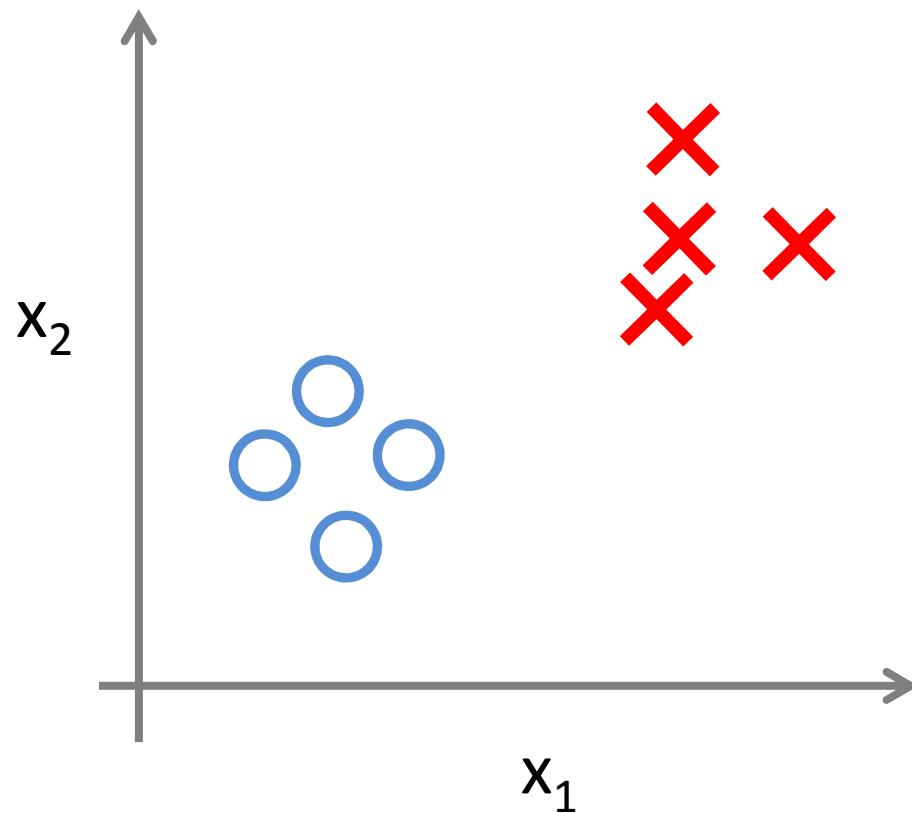


Machine Learning

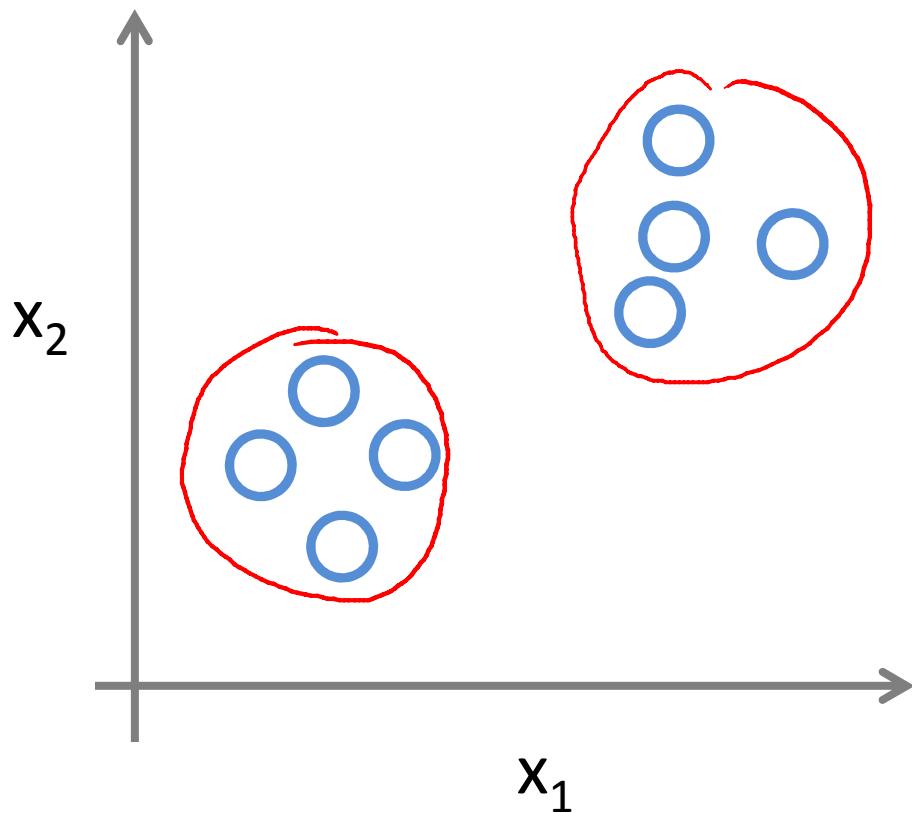
Introduction

Unsupervised Learning

Supervised Learning



Unsupervised Learning



Google News news.google.com andrewyantakng@gmail.com | Web History | Settings ▾ | Sign out

Web Images Videos Maps News Shopping Gmail more ▾ 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 misplaced 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 »

 CNN Interna...
 MyFox Phila...
 Reuters

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

Top Stories

- Deepwater Horizon
- Fed meeting
- forex exchange market
- Lindsay Lohan
- IBM
- Vinny Brady
- Toronto International Film Festival
- Pars Hilton
- India
- Hurricane Igor
- Starred**
- San Francisco Bay Area**
- World**
- U.S.**
- Business**
- SciTech
- More Top Stories

Top Stories

Christine O'Donnell » **White House official denies Tea Party-focused ad campaign**
 U.S. stocks end up after two days of losses - 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 says the White House is not considering an ad campaign that claims President Obama's political advisers are weighing a national Tea Party is misplacing the blame, former President Bill Clinton claims.
 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 Stock Climb After Recession Called Over, Hurricane Irene Gain
 MarketWatch - Kristina Peterson - 16 minutes ago
 NEW YORK (MarketWatch) — U.S. stocks climbed Monday, gaining speed after a two-day decline, as investors grew 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 - Dowtown Was Longest in Decades, Panel Confirms New York Times Wall Street Journal - AFP - CNN - USA Today all 2,292 news articles »

Deepwater Horizon
BP Oil Well, Site of National Catastrophe, Dies at One
 Vanity Fair - July Weiner - 22 minutes ago
 The BP oil well site of the Deepwater Horizon explosion is now 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 - The Wall Street Journal (blog) - The Guardian - New York Times all 2,292 news articles »

San Francisco Bay Area - Edit
 Click here to edit...
 San Francisco Buzz - Cloenox close to selling STP...
 Among All San Jose Mercury News - 48 minutes ago
 all 23 articles »

George W. Bush's official biographer keeps the company
 Business - Mercury News - Bruce Newman - 1 hour ago
 Joe Sylva -
 Most women 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 - September 2010

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

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

CNN

HOME | VIDEO | WORLD | U.S. | MEXICO | ARABIC | Set edition preference

Africa Asia Europe Latin America Middle East Business

Click to play

What next for Gulf oil spill?

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

BP Kills Macondo, But Its Legacy Lives On blogs.wsj.com/source/2010/09/20/bp-kills-macondo-but-its-legacy-lives-on/ THE WALL STREET JOURNAL Digital Network WSJ.com MarketWatch BARRON'S AllThings.Digital FIN'S SmartMoney More Log In Register For Free Subscribe Now, Get 2 Weeks Free SEARCH

THE SOURCE

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

SEPTEMBER 20, 2010, 12:44 PM GMT

BP Kills Macondo, But Its Legacy Lives On

Article Comments (2)

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 spread responsibility across

San Francisco Bay Area

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 totaling more than \$240m

Julia Kollewe guardian.co.uk, Monday 20 September 2010 08:33 BST Article history

BP oil spill cost hits nearly \$10bn

BP oil spill cost hits nearly \$10bn

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

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 totaling more than \$240m

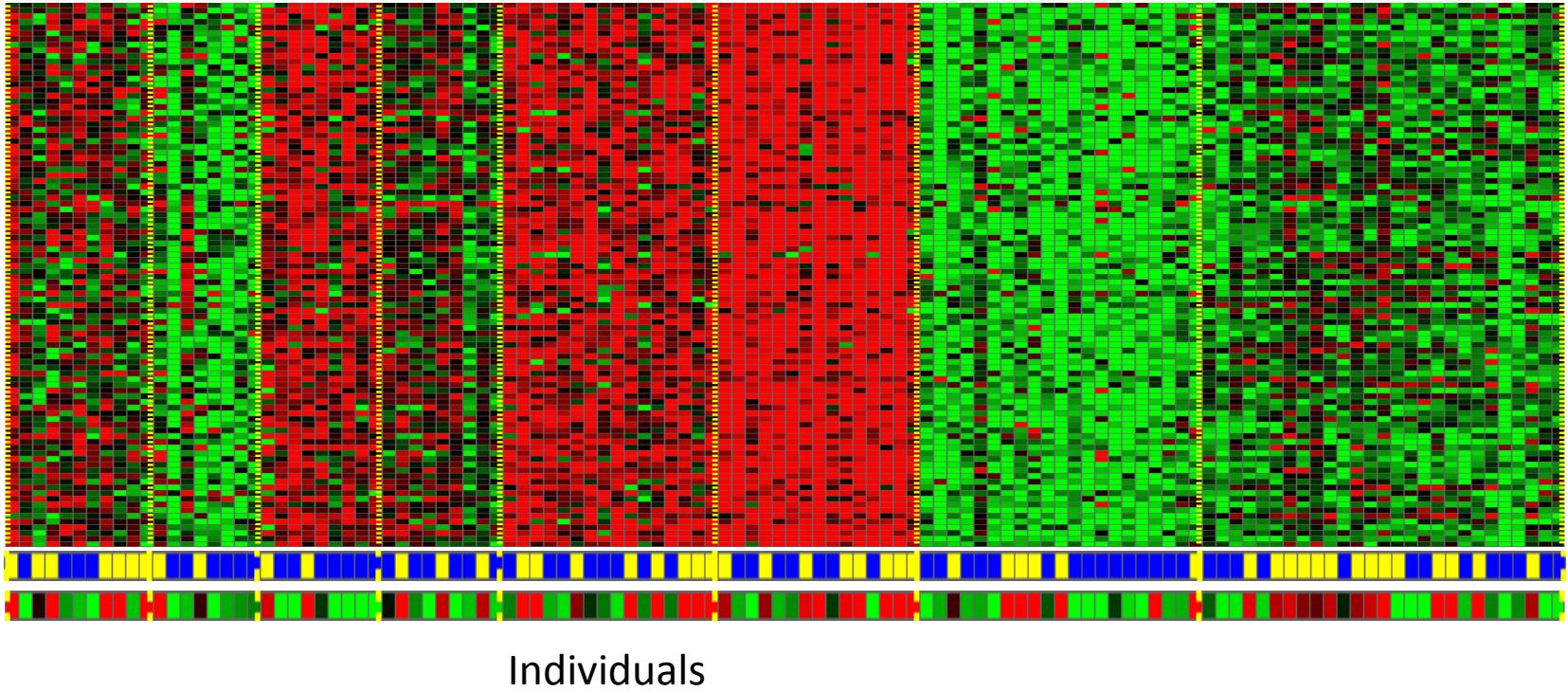
Julia Kollewe guardian.co.uk, Monday 20 September 2010 08:33 BST Article history

BP oil spill cost hits nearly \$10bn

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

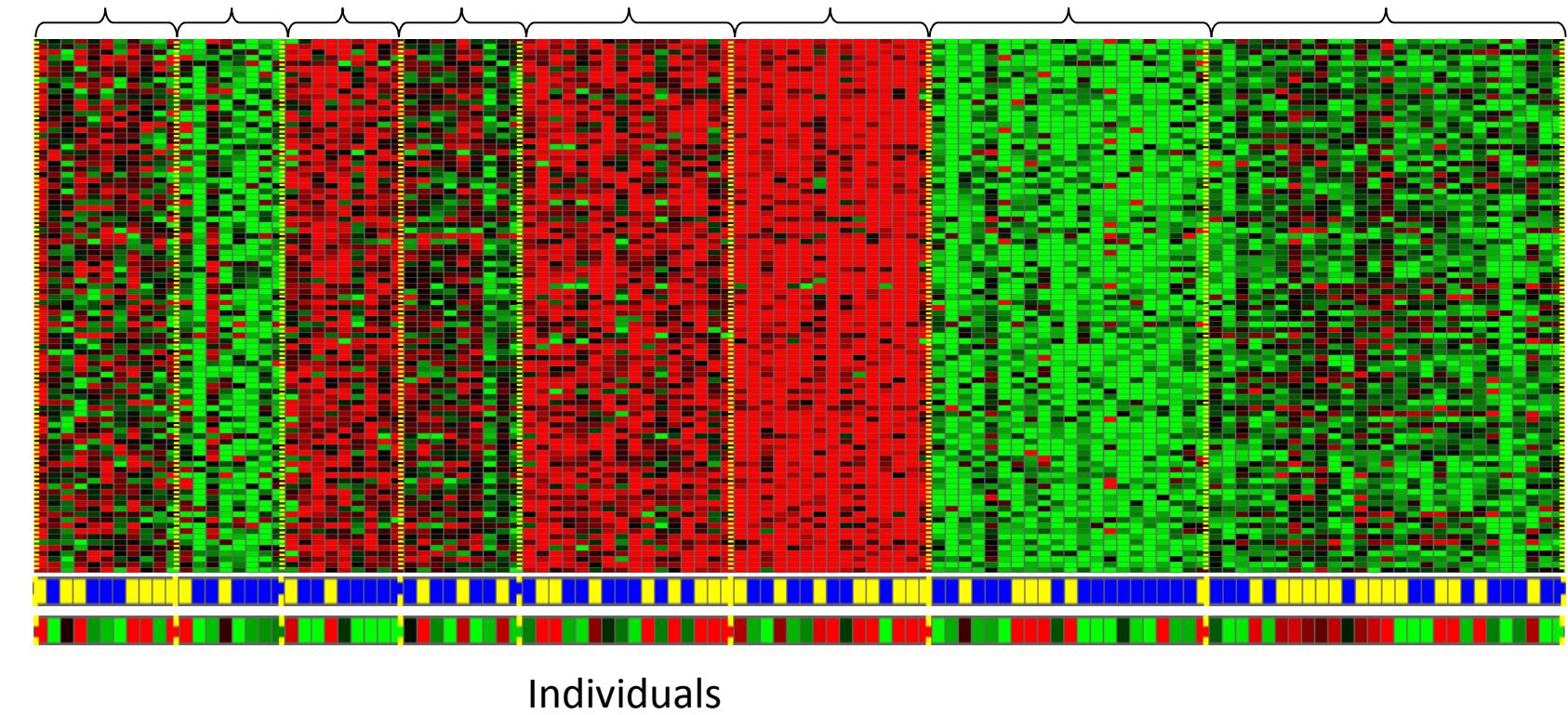
Andrew Ng

Genes



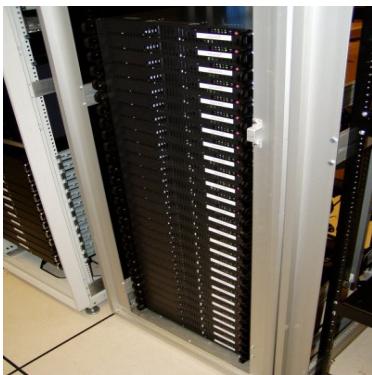
[Source: Daphne Koller]

Andrew Ng

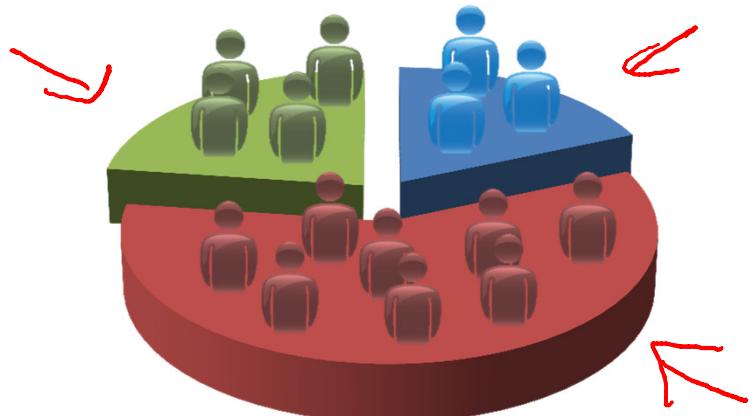


[Source: Daphne Koller]

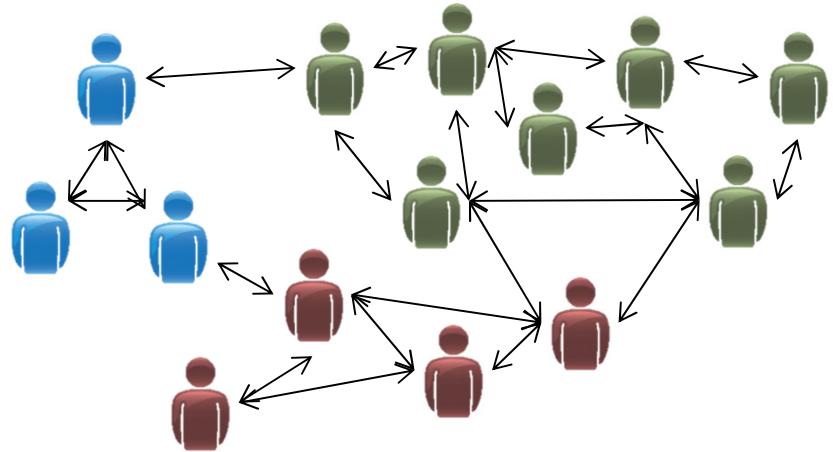
Andrew Ng



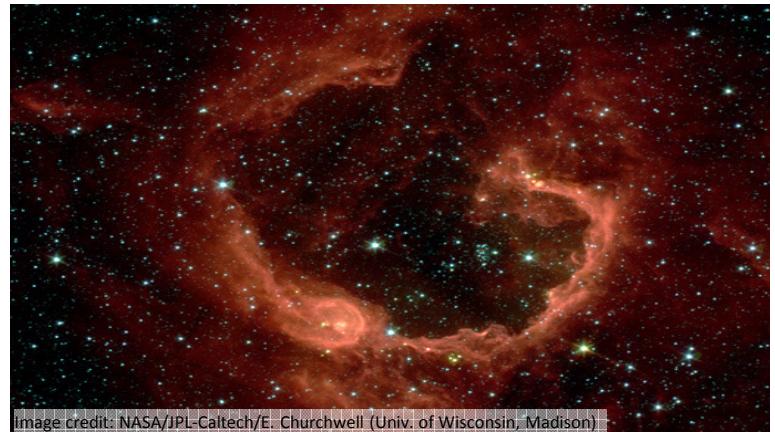
Organize computing clusters



Market segmentation



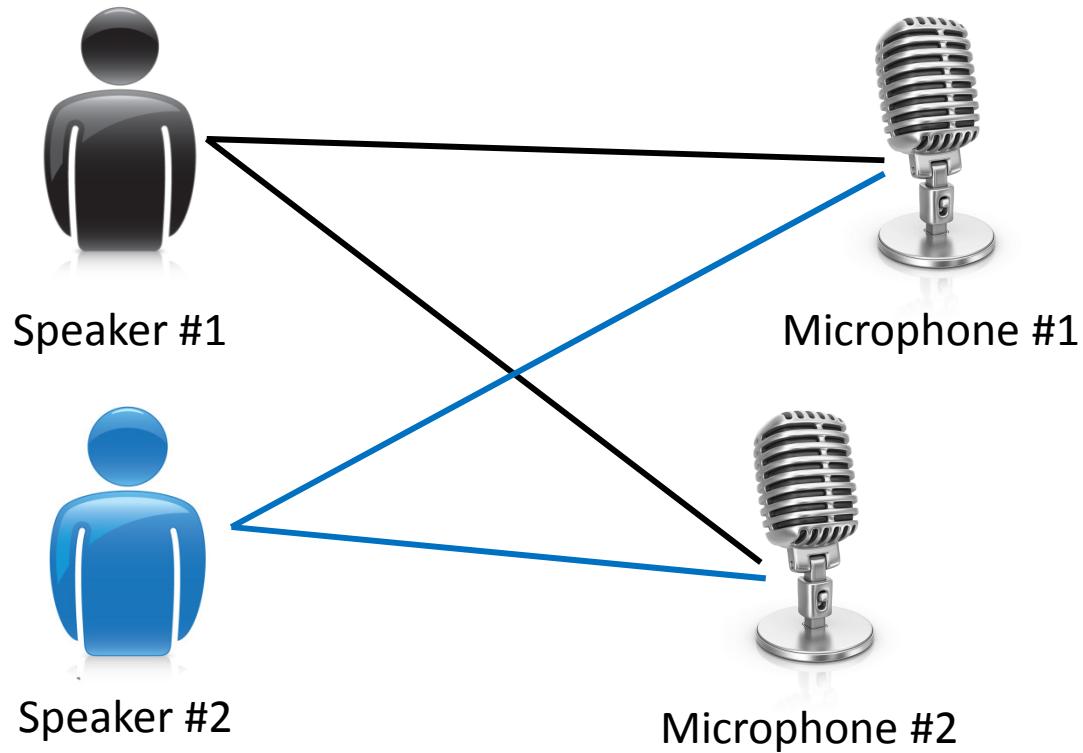
Social network analysis



Astronomical data analysis

Andrew Ng

Cocktail party problem



Microphone #1: 

Output #1: 

Microphone #2: 

Output #2: 

Microphone #1: 

Output #1: 

Microphone #2: 

Output #2: 

[Audio clips courtesy of Te-Won Lee.]

Andrew Ng

Cocktail party problem algorithm

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

[Source: Sam Roweis, Yair Weiss & Eero Simoncelli]

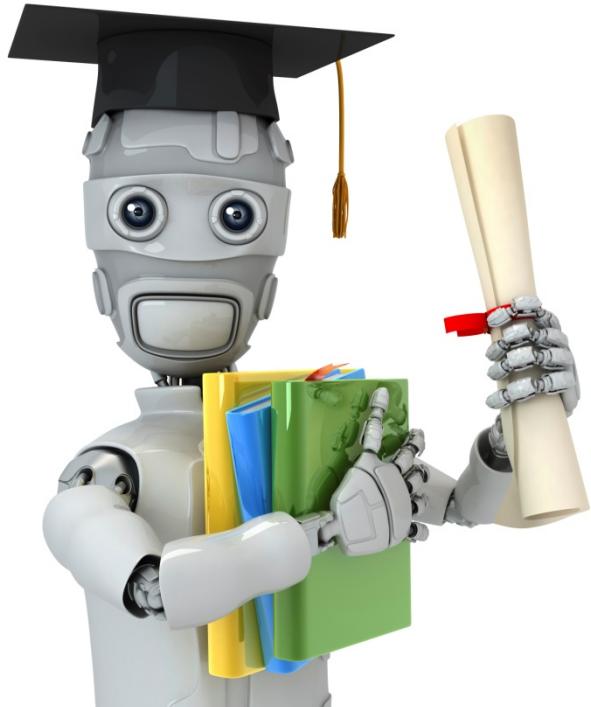
Andrew Ng

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.

Andrew Ng

1.2 Gradient Descent



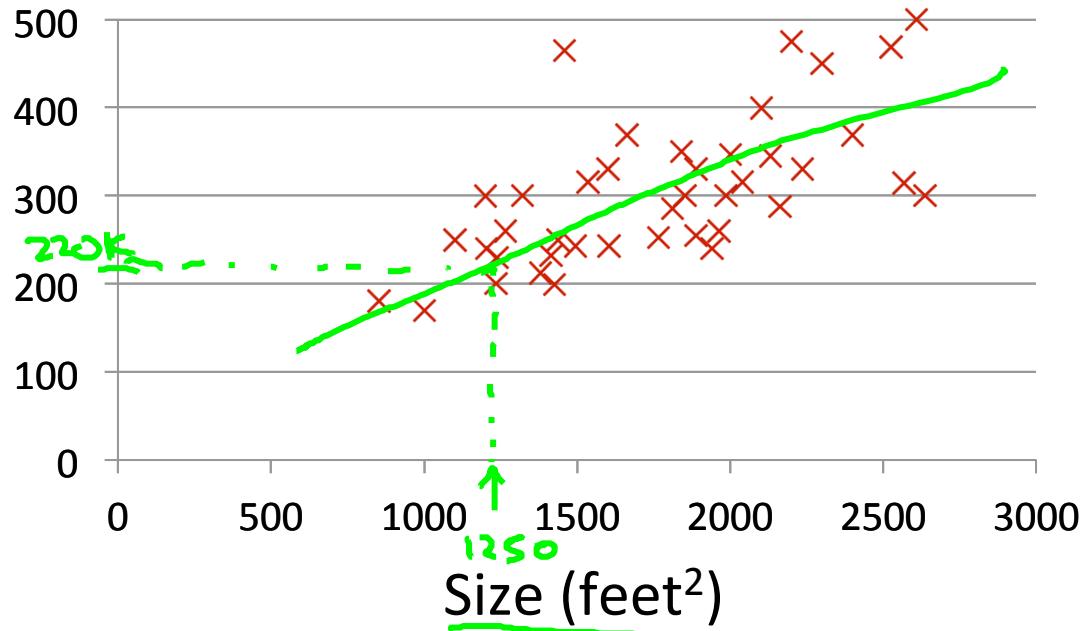
Machine Learning

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

Classification: Discrete-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
...	...
i	i

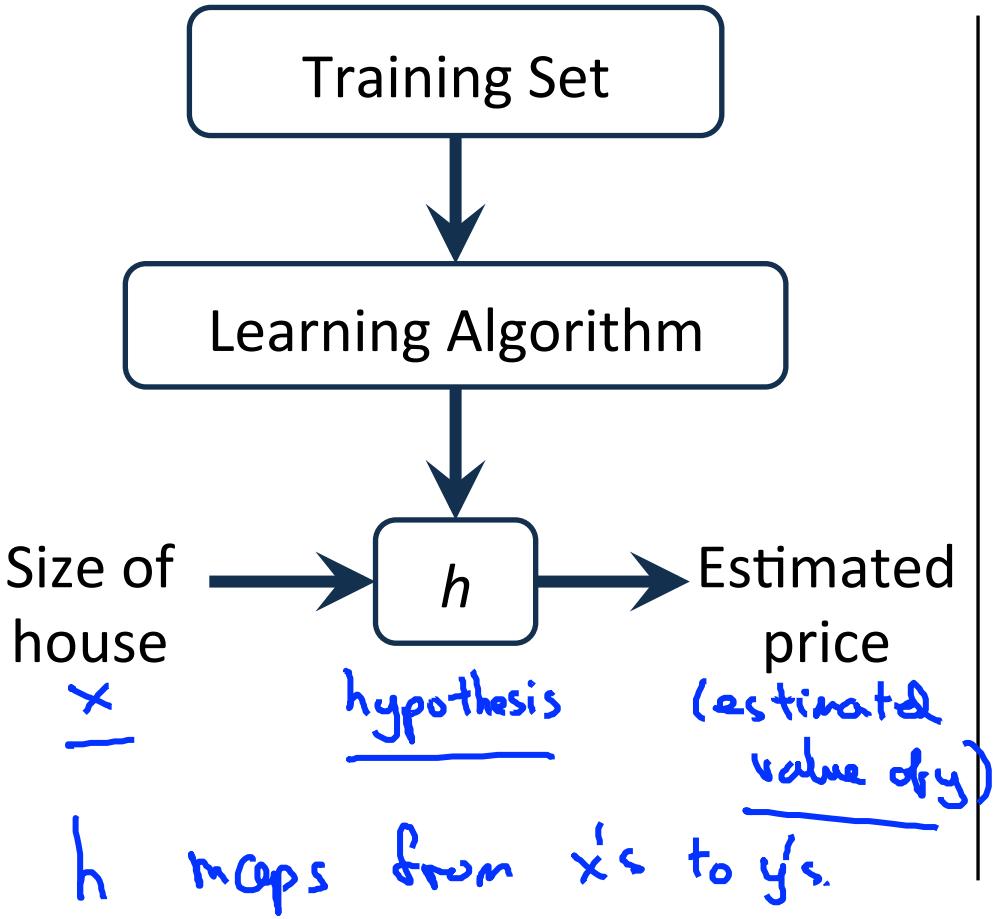
Notation:

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

(x, y) - one training example

$(x^{(i)}, y^{(i)})$ - ith training example

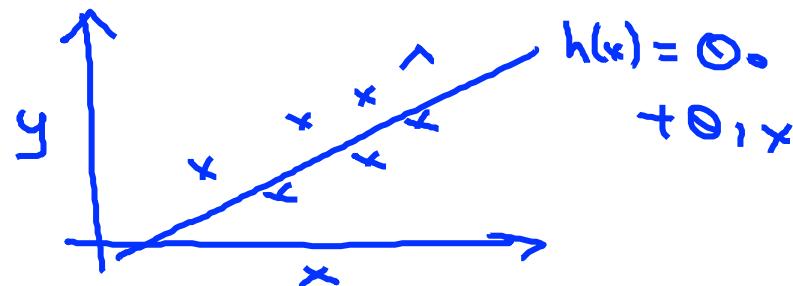
$$\begin{cases} x^{(1)} = 2104 \\ x^{(2)} = 1416 \\ \vdots \\ y^{(1)} = 460 \end{cases}$$



How do we represent h ?

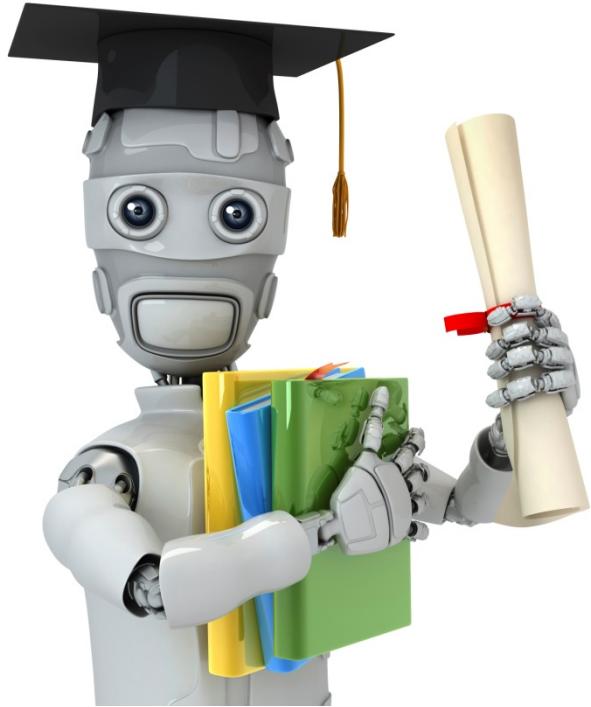
$$h_{\theta}(x) = \underline{\theta_0 + \theta_1 x}$$

Shorthand: $h(x)$



Linear regression with one variable.
Univariate linear regression.

one variable



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
...	...

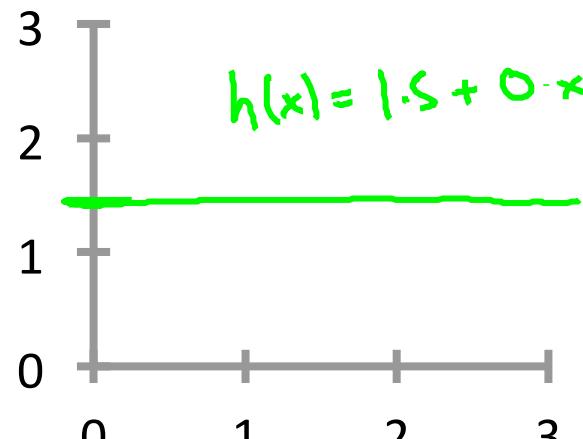
$$m = 47$$

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

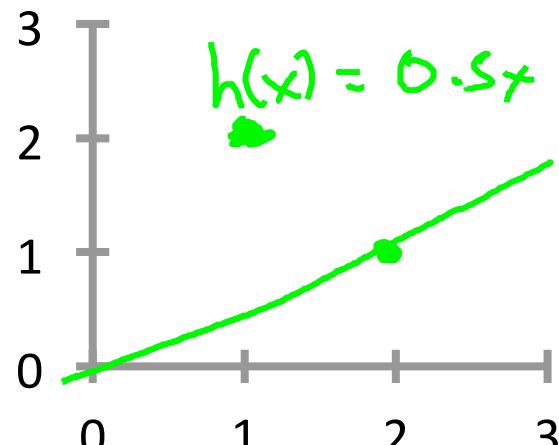
θ_i 's: Parameters

How to choose θ_i 's ?

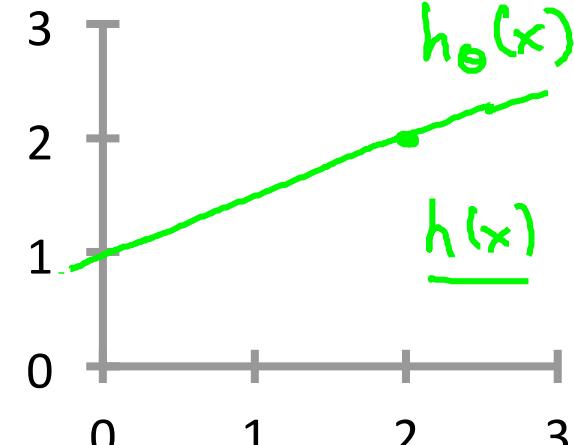
$$\underline{h_{\theta}(x) = \theta_0 + \theta_1 x}$$



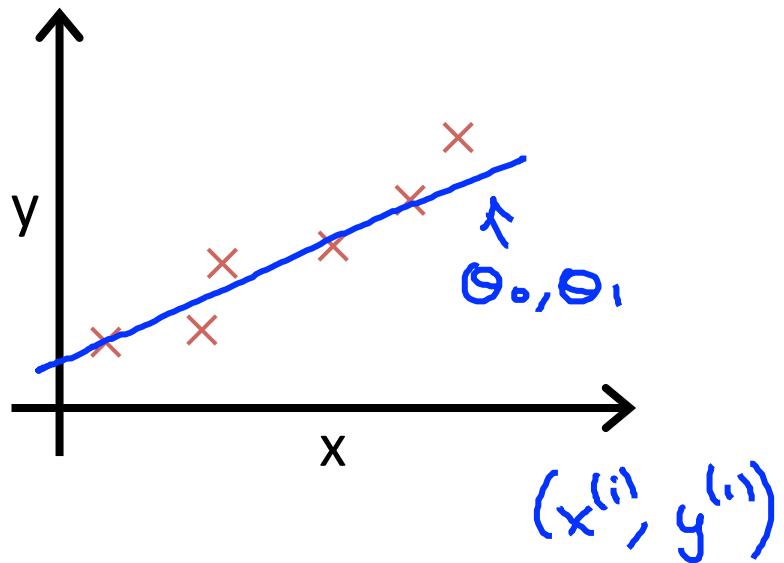
$$\rightarrow \theta_0 = 1.5$$
$$\rightarrow \theta_1 = 0$$



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



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



Idea: Choose $\underline{\theta_0}, \underline{\theta_1}$ so that
 $\underline{h_\theta(x)}$ is close to \underline{y} for our
 training examples $\underline{(x, y)}$

x, y

minimize $\underline{\theta_0, \theta_1}$

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

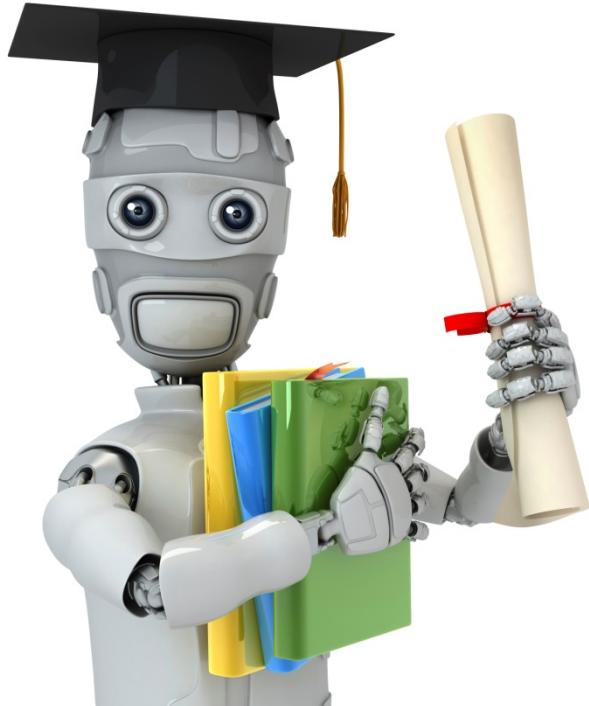
#training examples

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

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

minimize $\underline{\theta_0, \theta_1}$ $J(\theta_0, \theta_1)$

Squared error function



Machine Learning

Linear regression with one variable

Cost function intuition I

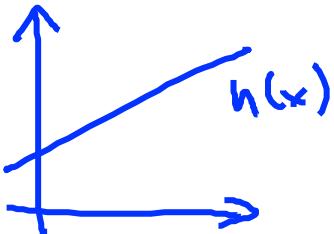
Simplified

Hypothesis:

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

Parameters:

$$\underline{\theta_0, \theta_1}$$



Cost Function:

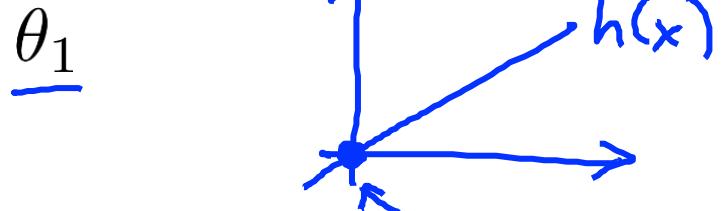
$$\rightarrow J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Goal: minimize $J(\theta_0, \theta_1)$

$$\nearrow \theta_0, \theta_1$$

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

$$\underline{\theta_0 = 0}$$



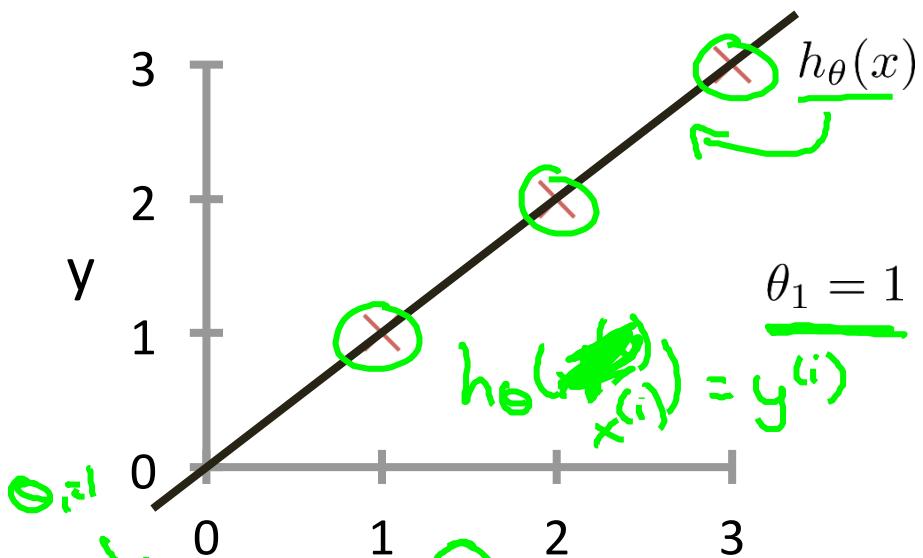
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

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

$$\underline{\theta_1, x^{(i)}}$$

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

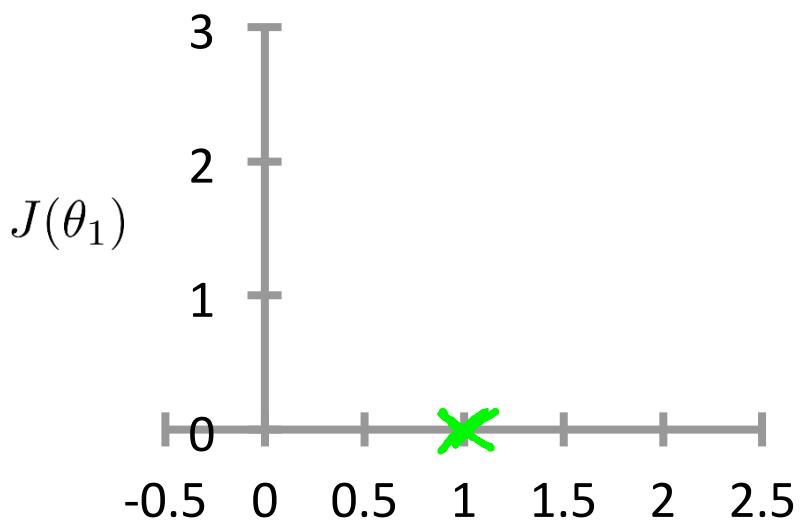
(for fixed θ_1 , this is a function of x)



$$\begin{aligned} J(\theta_1) &= \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (\theta_0 x^{(i)} - y^{(i)})^2 = \frac{1}{2m} (0^2 + 0^2 + 0^2) = 0^2 \end{aligned}$$

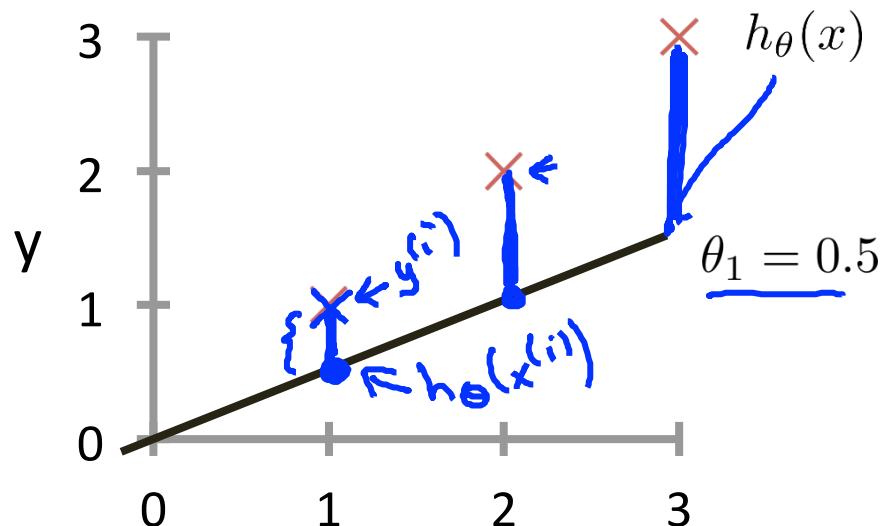
$\rightarrow J(\theta_1)$

(function of the parameter θ_1)



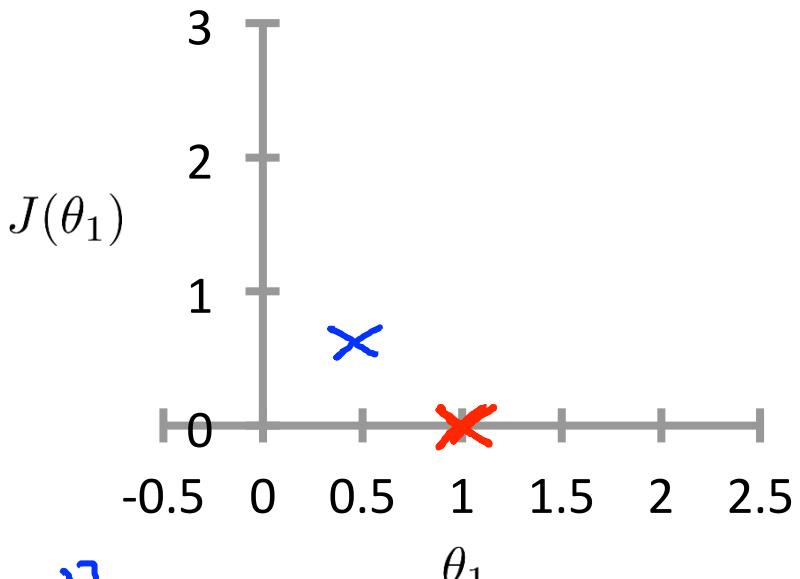
$$J(1) = 0$$

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

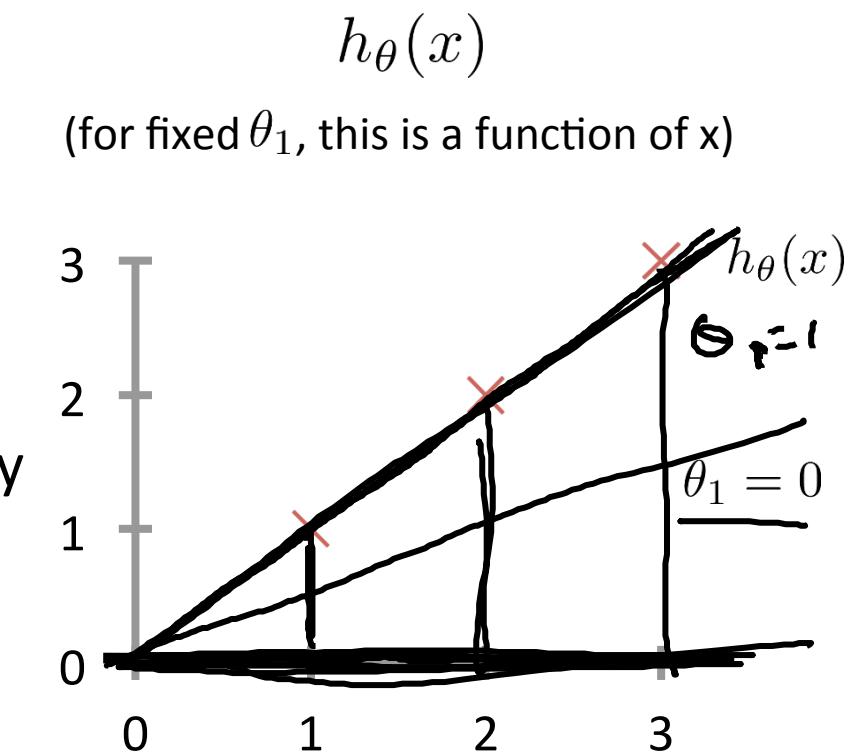


$$\begin{aligned} J(0.5) &= \frac{1}{2m} \sum_{i=1}^m [(0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2] \\ &= \frac{1}{2 \times 3} (3.5) = \frac{3.5}{6} \approx \underline{0.58} \end{aligned}$$

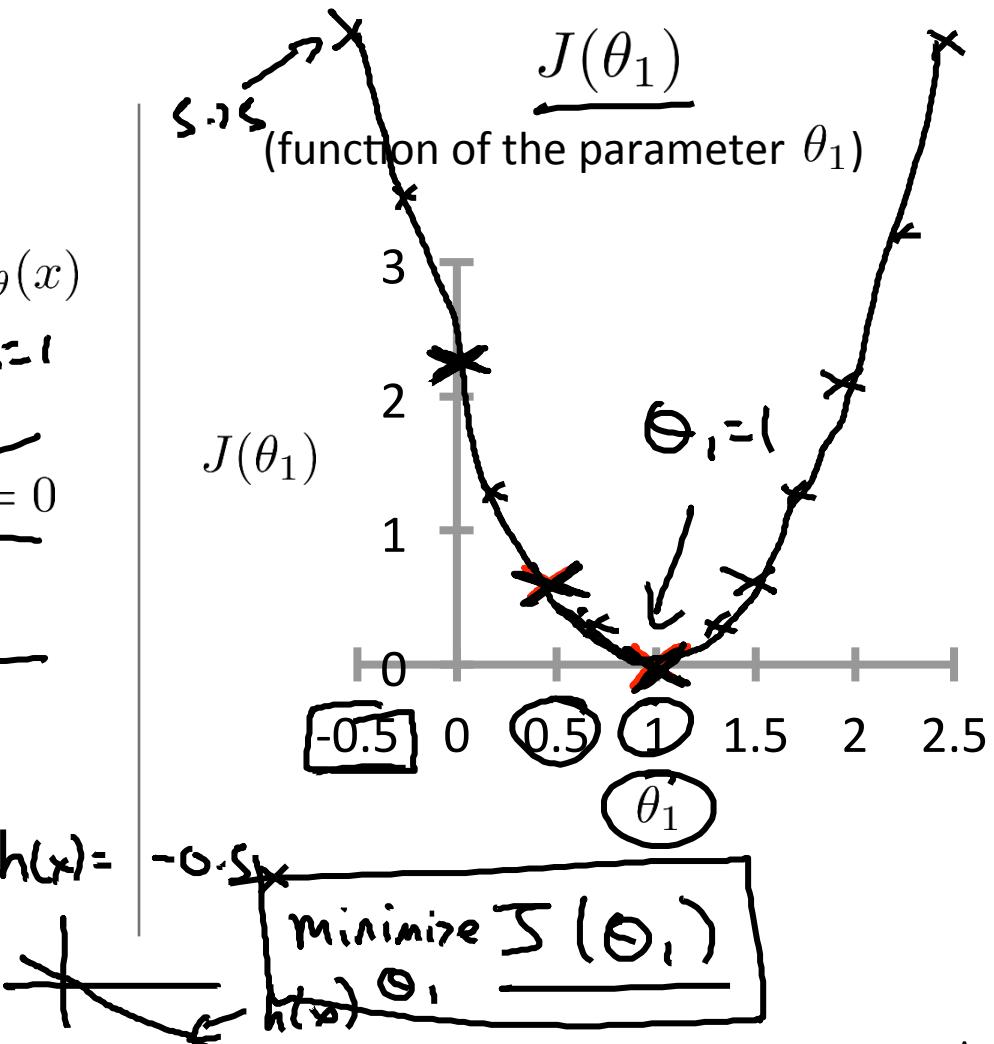
$J(\theta_1)$
(function of the parameter θ_1)

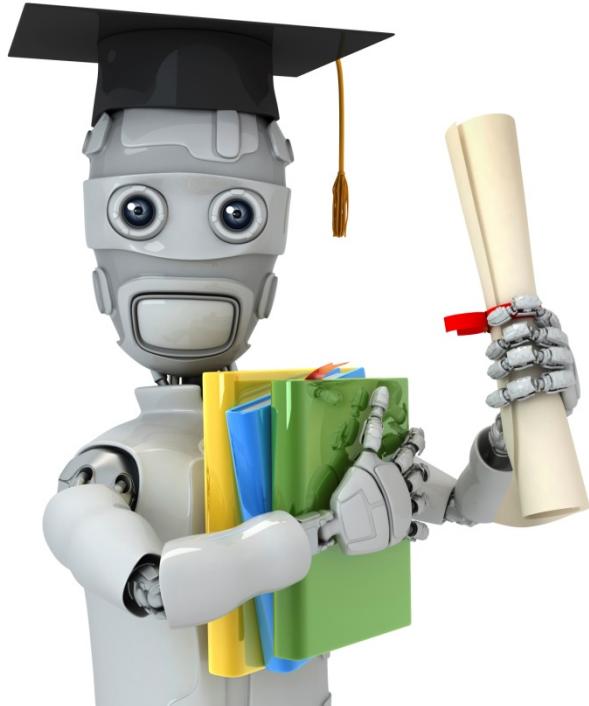


$$\begin{aligned} \theta_1 &=? \\ J(0) &=? \end{aligned}$$



$$\begin{aligned}
 J(0) &= \frac{1}{2m} (1^2 + 2^2 + 3^2) \\
 &= \frac{1}{6} \cdot 14 \approx 2.3
 \end{aligned}$$





Machine Learning

Linear regression with one variable

Cost function intuition II

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

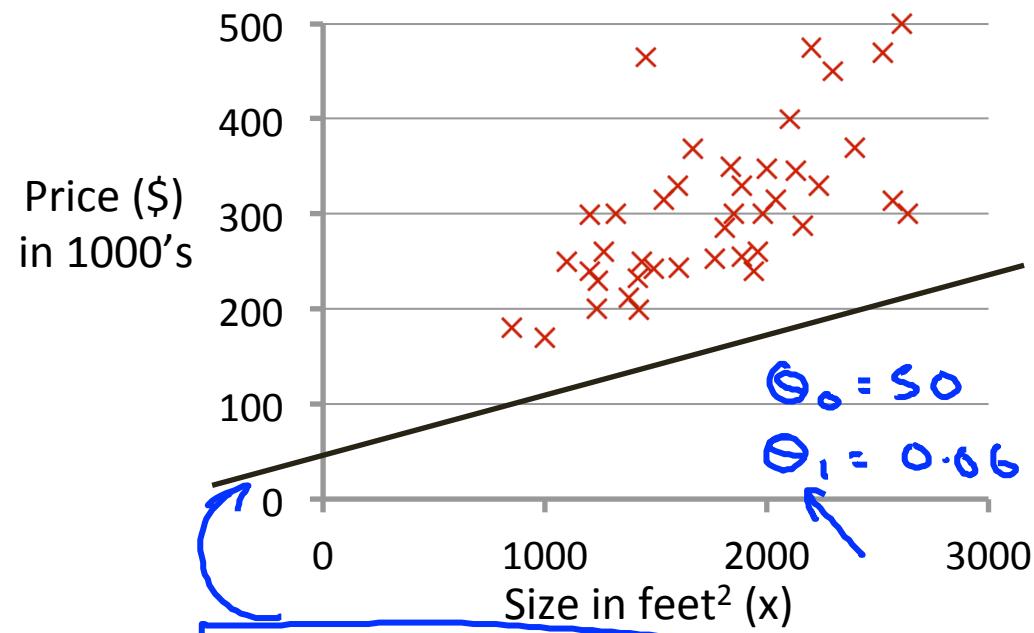
Parameters: θ_0, θ_1

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

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

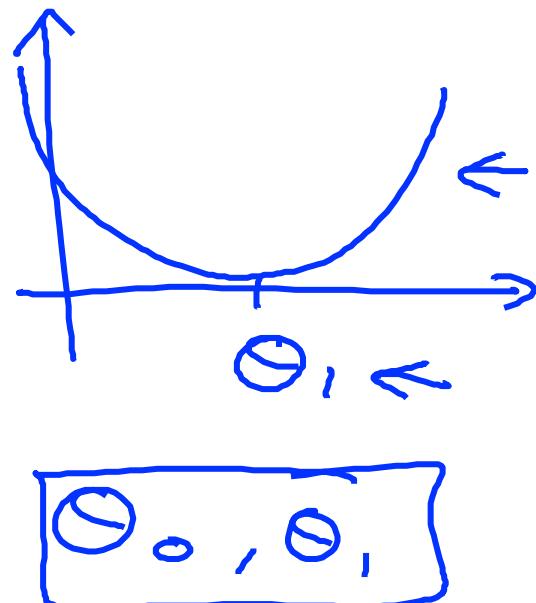
.

$h_{\theta}(x)$
(for fixed θ_0, θ_1 , this is a function of x)

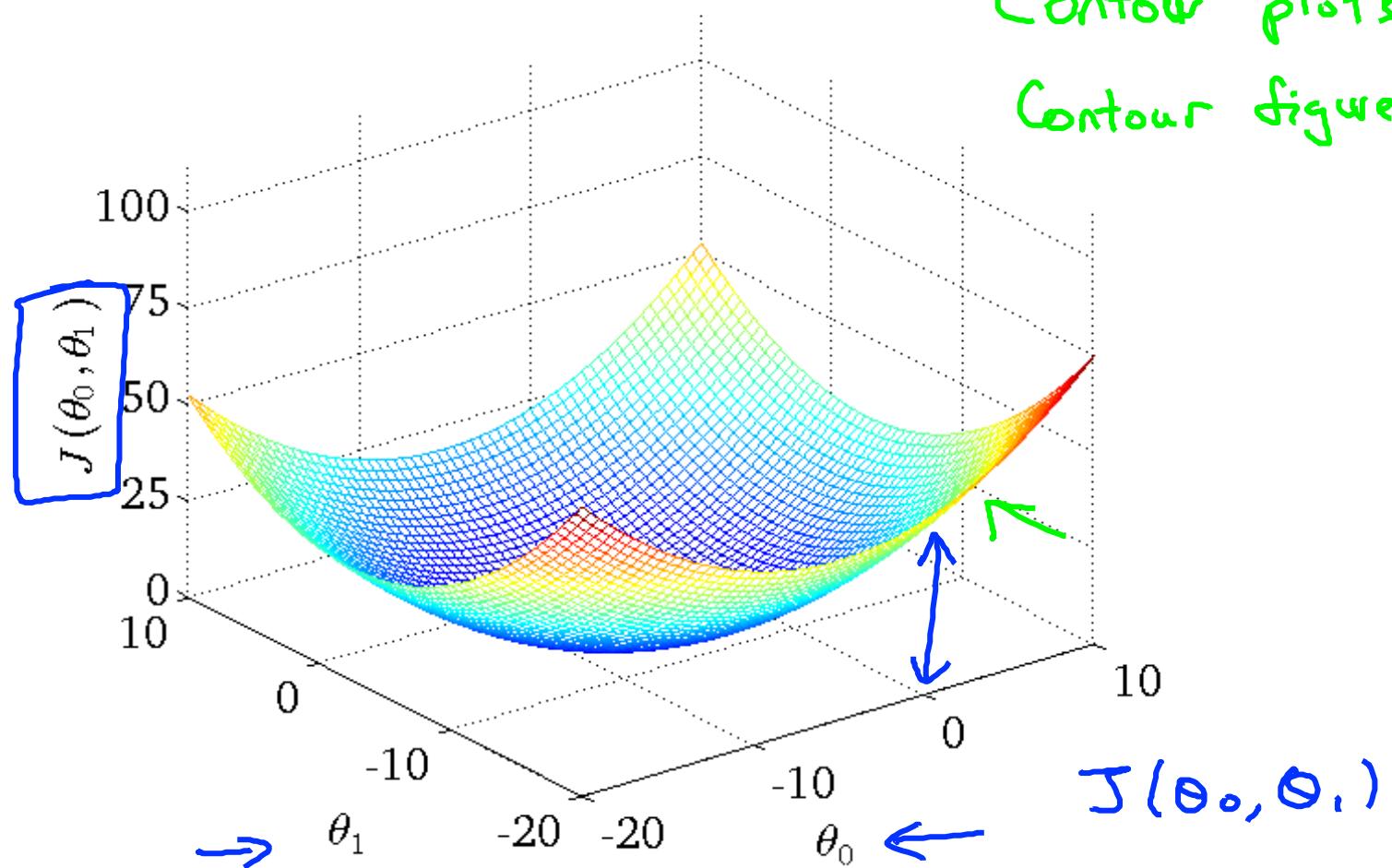


$$h_{\theta}(x) = 50 + 0.06x$$

$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)

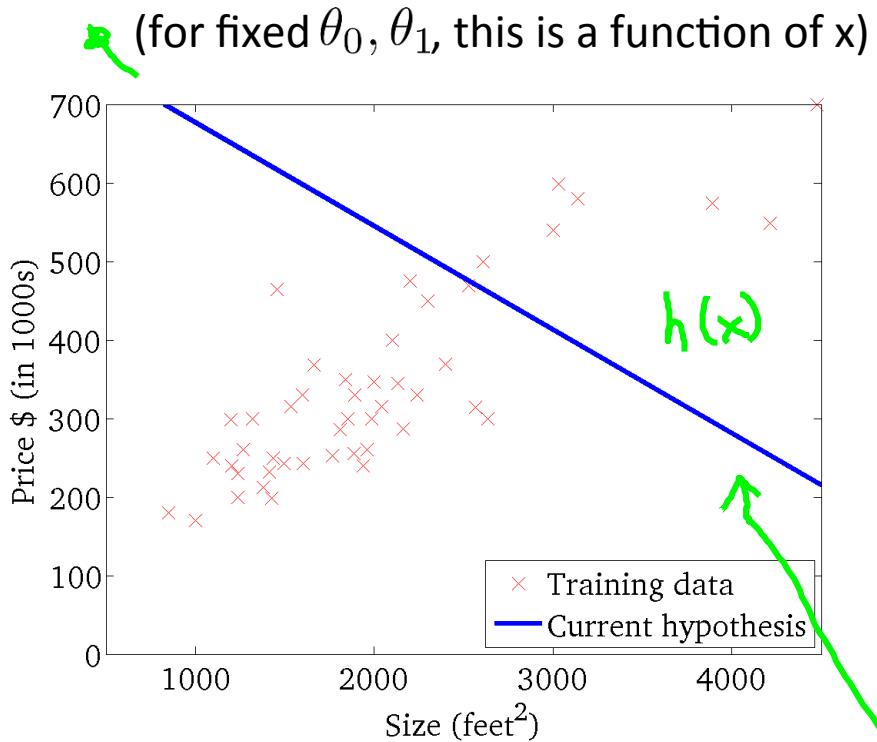


Contour plots
Contour figures -

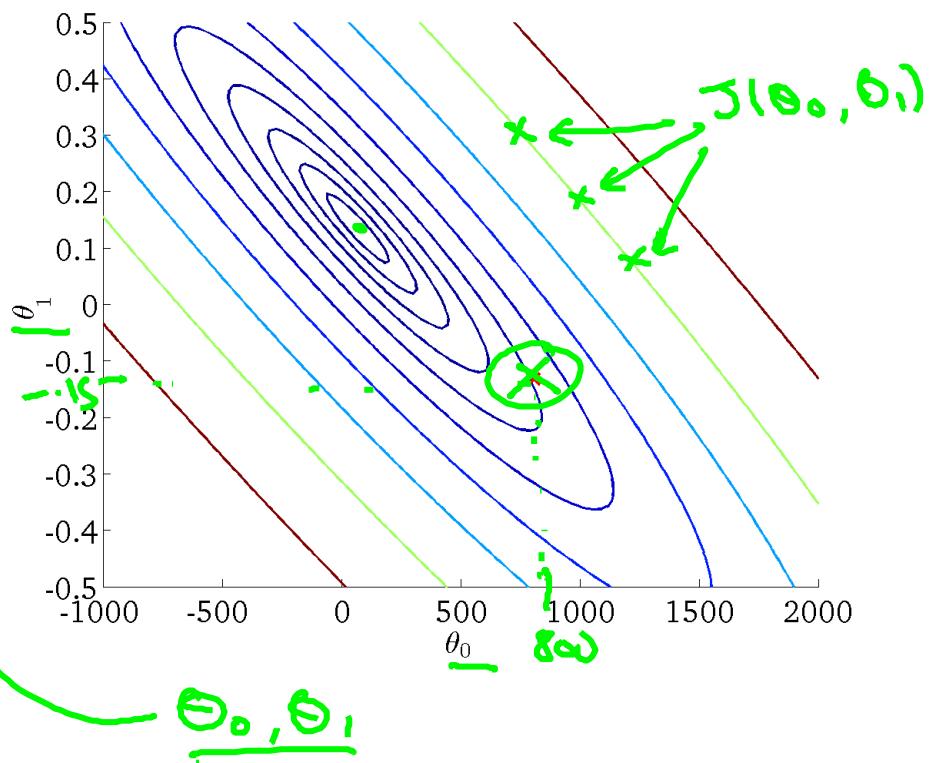


$$h_{\theta}(x)$$

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

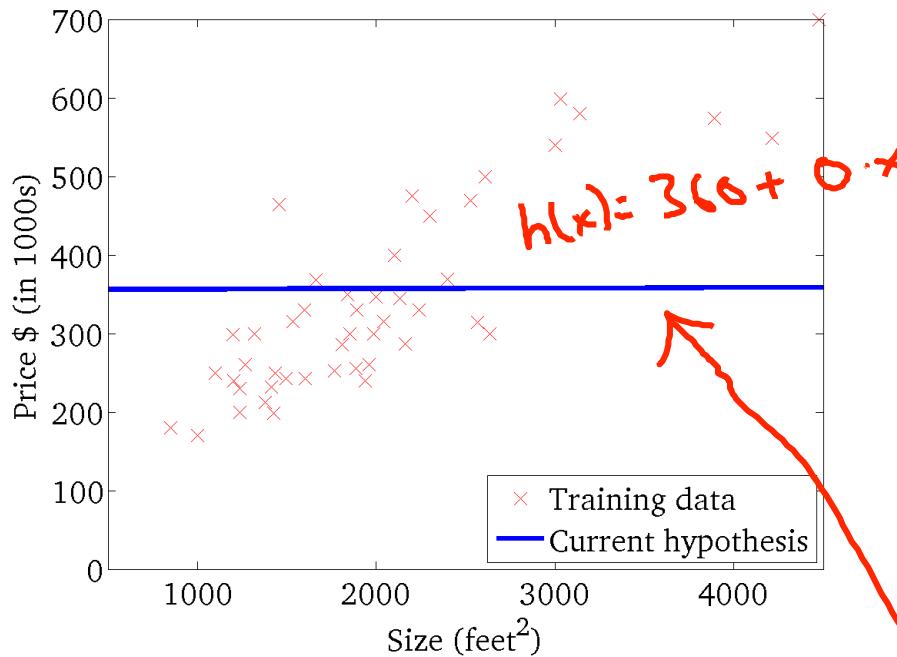


(function of the parameters θ_0, θ_1)



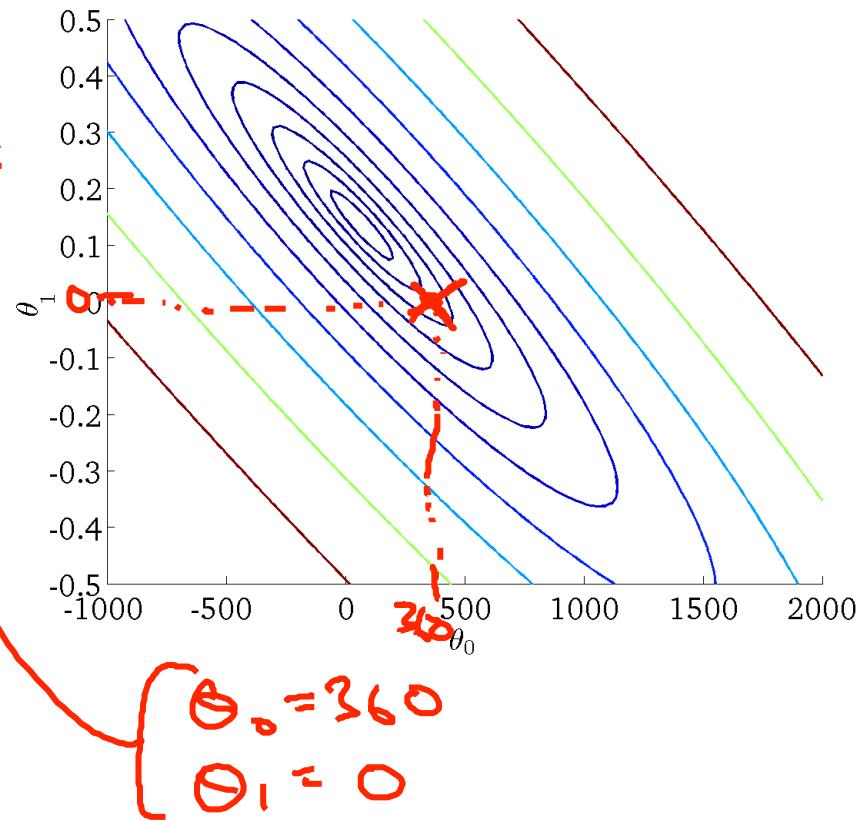
$$h_{\theta}(x)$$

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



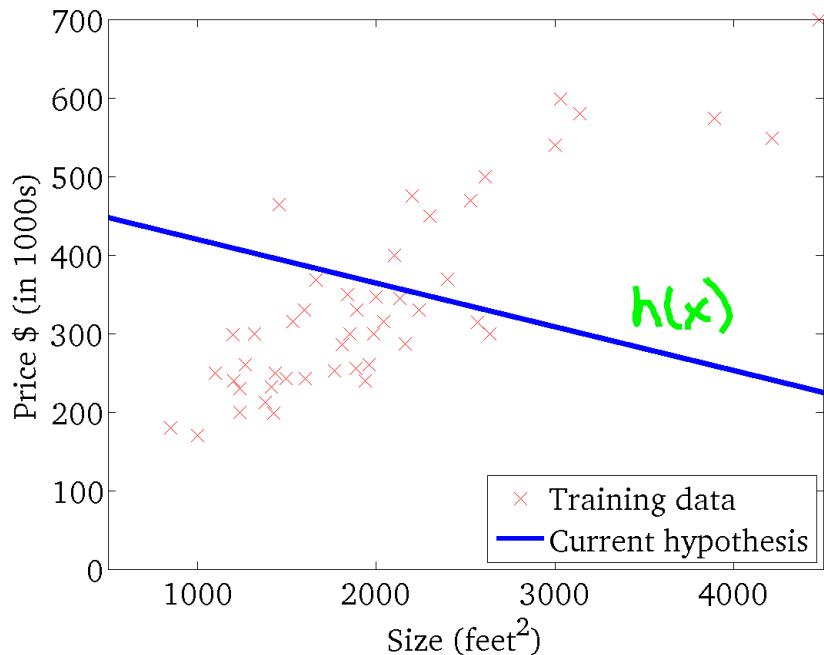
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



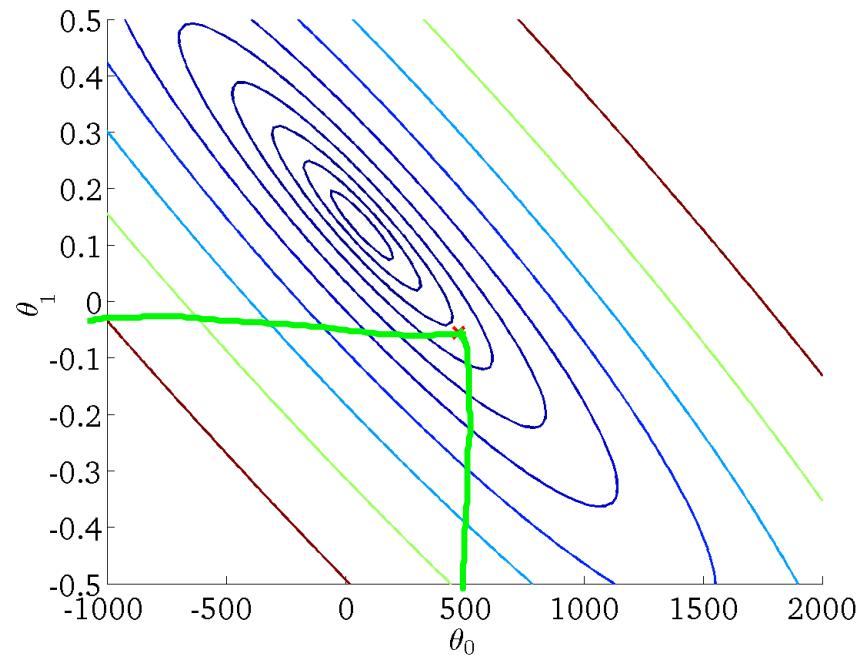
$$h_{\theta}(x)$$

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



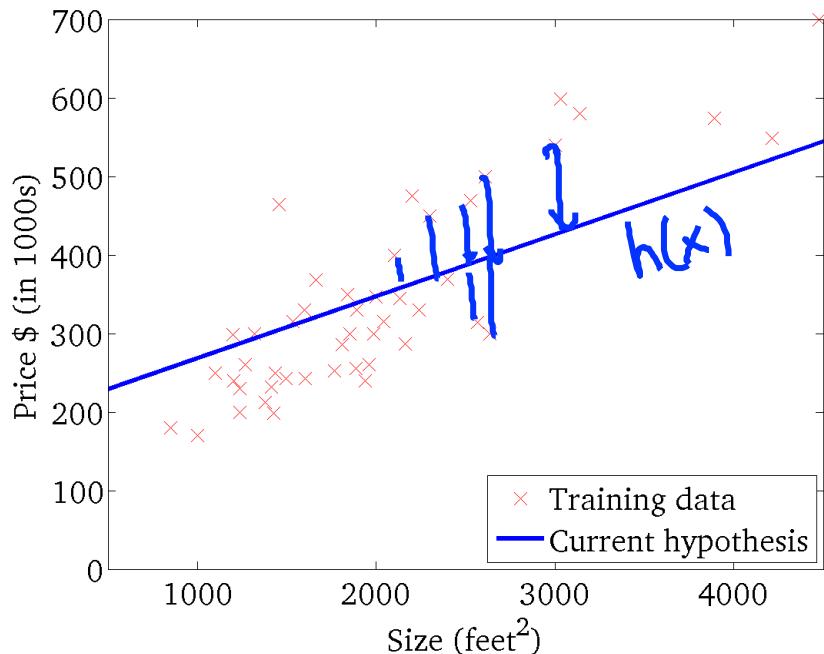
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



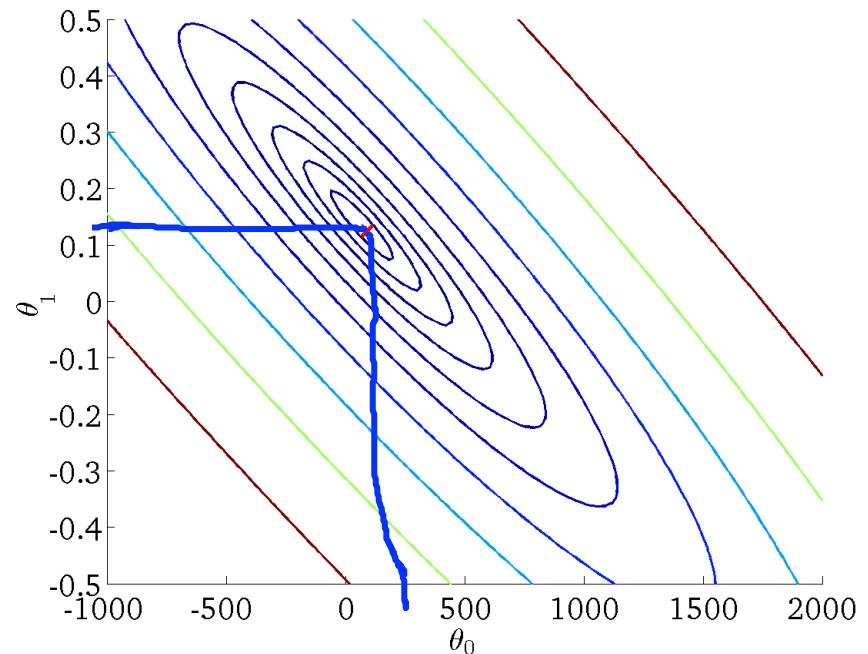
$$h_{\theta}(x)$$

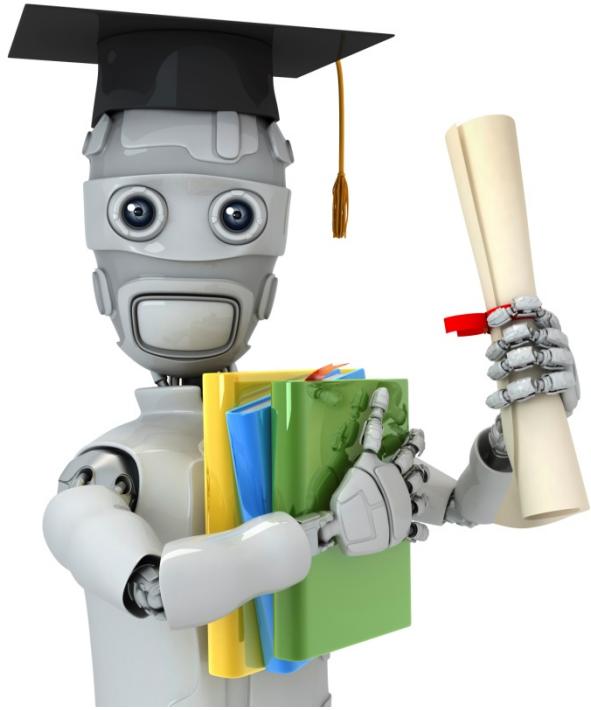
(for fixed θ_0, θ_1 , this is a function of x)



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

(function of the parameters θ_0, θ_1)





Machine Learning

Linear regression
with one variable

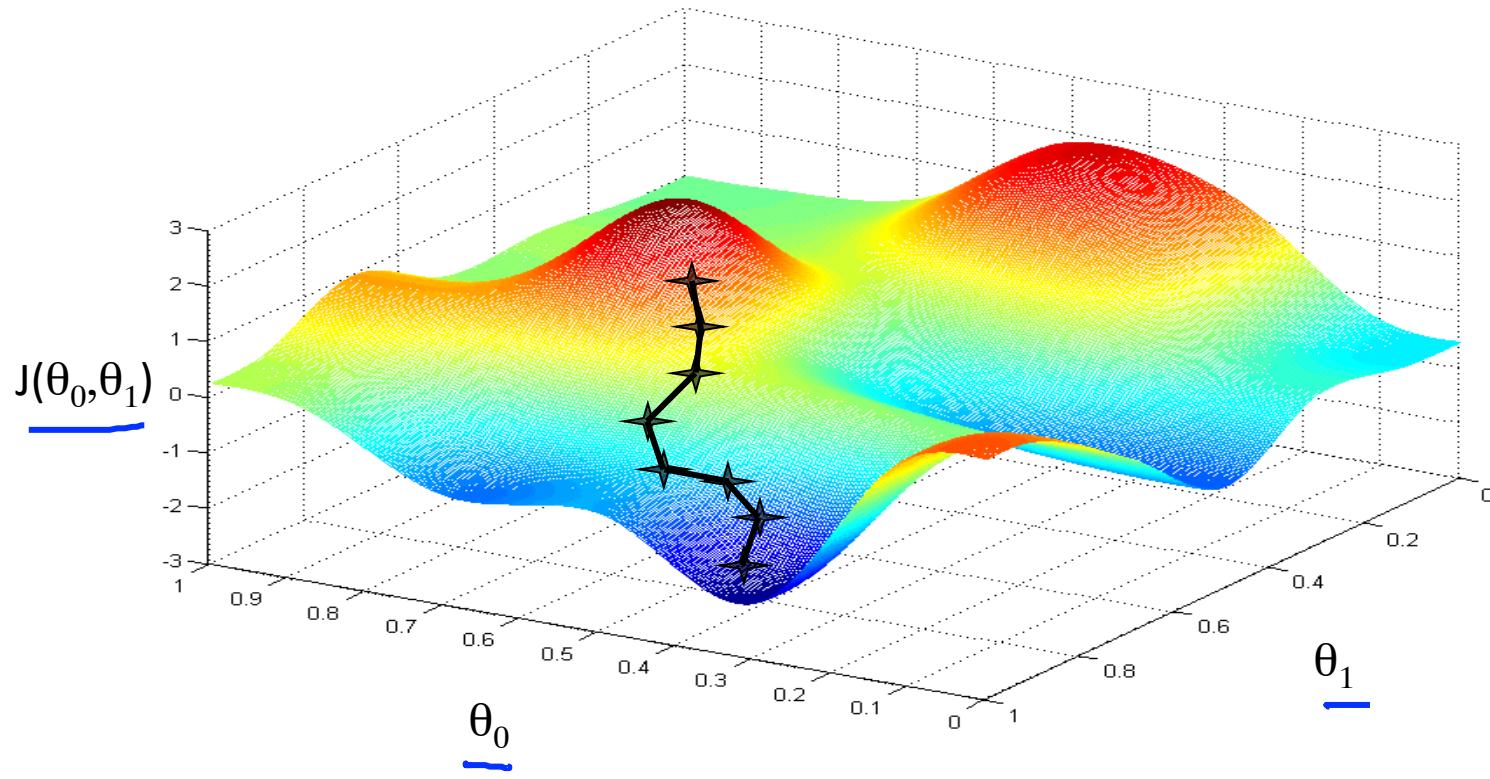
Gradient
descent

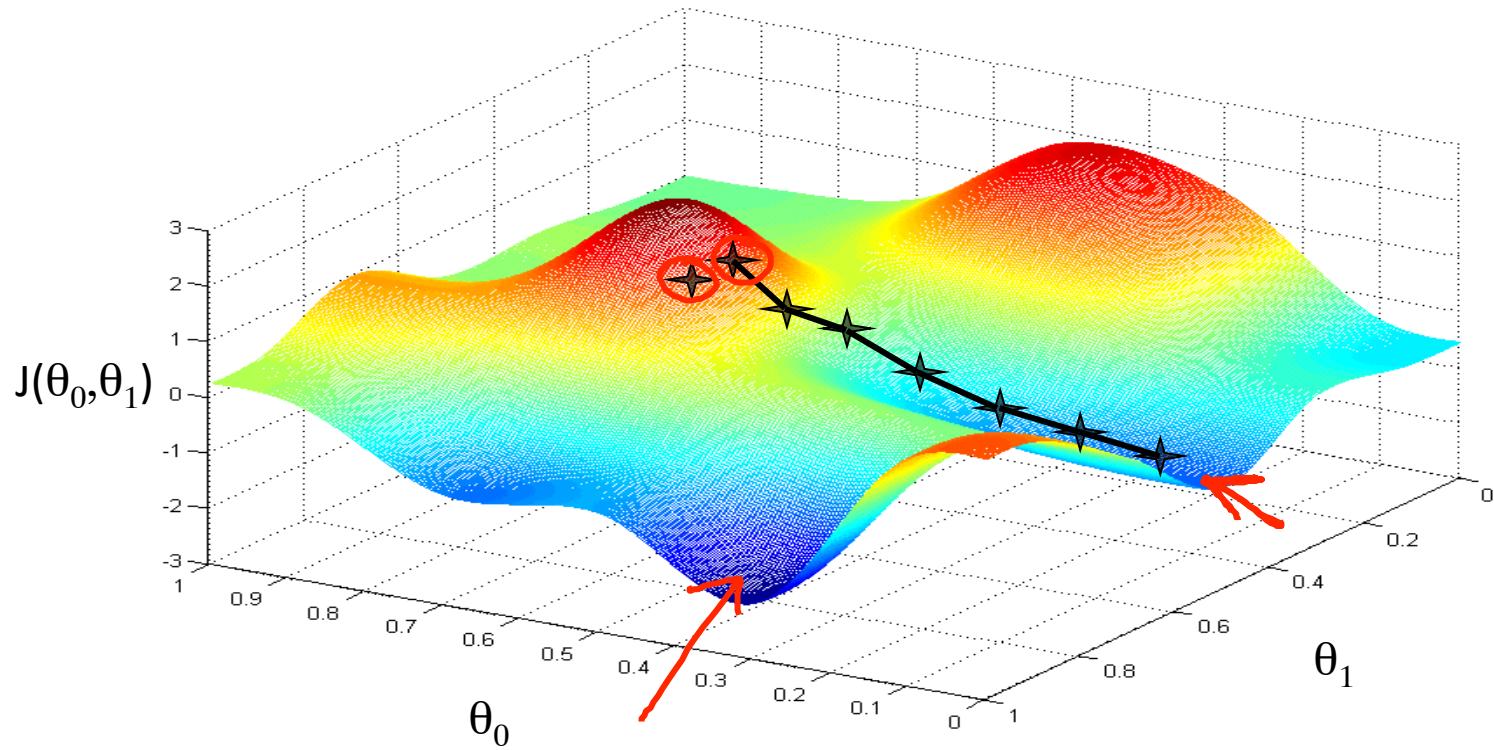
Have some function $\underline{J(\theta_0, \theta_1)}$ $J(\theta_0, \theta_1, \theta_2, \dots, \theta_n)$

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

Outline:

- Start with some $\underline{\theta_0, \theta_1}$ (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing $\underline{\theta_0, \theta_1}$ to reduce $\underline{J(\theta_0, \theta_1)}$
until we hopefully end up at a minimum





Andrew Ng

Gradient descent algorithm

repeat until convergence {

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

learning rate

θ_0, θ_1

Assignment

$$\begin{array}{l} a := b \\ \quad \uparrow \\ a := a + 1 \end{array}$$

Truth assertion

$$a = b \leftarrow$$

$$a = a + 1 \times$$

(for $j = 0$ and $j = 1$)

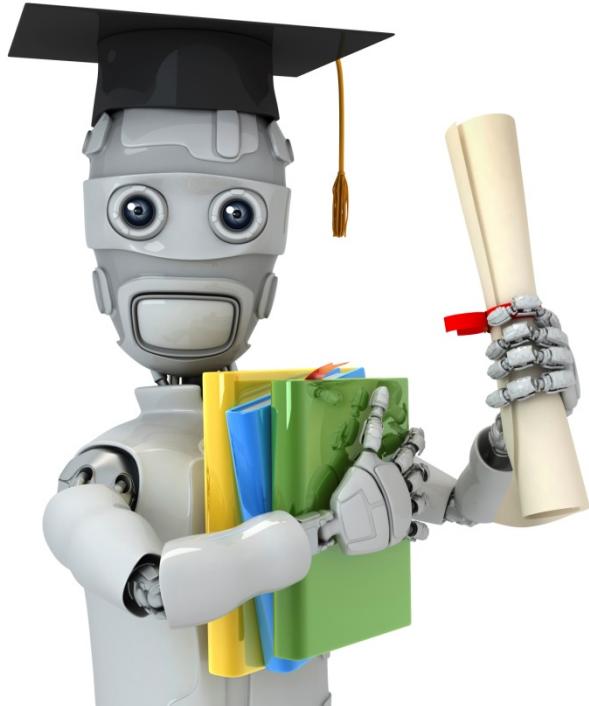
Simultaneously update
 θ_0 and θ_1

Correct: Simultaneous update

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

Incorrect:

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



Machine Learning

Linear regression with one variable

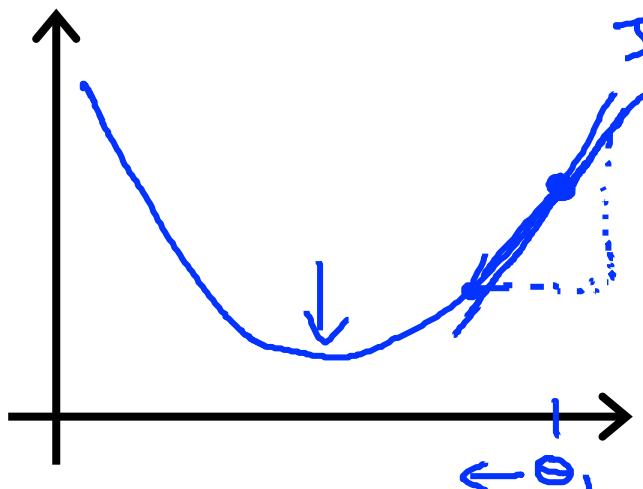
Gradient descent intuition

Gradient descent algorithm

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

learning rate derivative

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

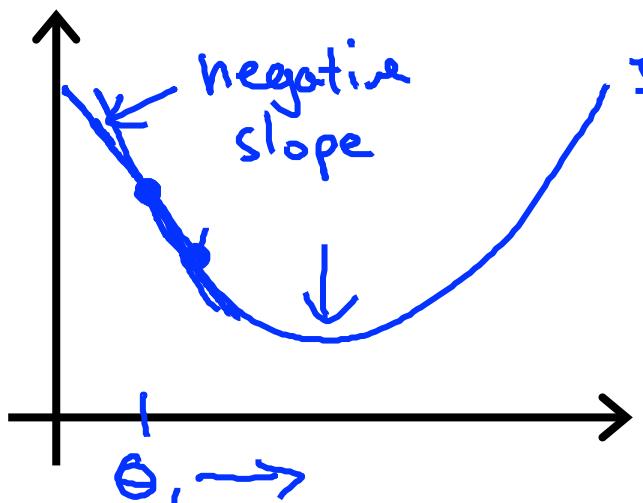


$J(\theta_1)$ ($\theta_1 \in \mathbb{R}$)

$$\theta_1 := \theta_1 - \frac{\alpha}{\frac{\partial}{\partial \theta_1} J(\theta_1)} \geq 0$$

Diagram illustrating the update rule: A blue box contains the formula $\theta_1 := \theta_1 - \frac{\alpha}{\frac{\partial}{\partial \theta_1} J(\theta_1)}$. Above the box, a small diagram shows a blue rectangle with a horizontal arrow labeled $\frac{\partial}{\partial \theta_1}$ pointing to its right side. A blue circle with a minus sign inside it is placed over the term $\frac{\alpha}{\frac{\partial}{\partial \theta_1}}$.

$$\theta_1 := \theta_1 - \frac{\alpha}{\frac{\partial}{\partial \theta_1} J(\theta_1)} \cdot (\text{positive number})$$

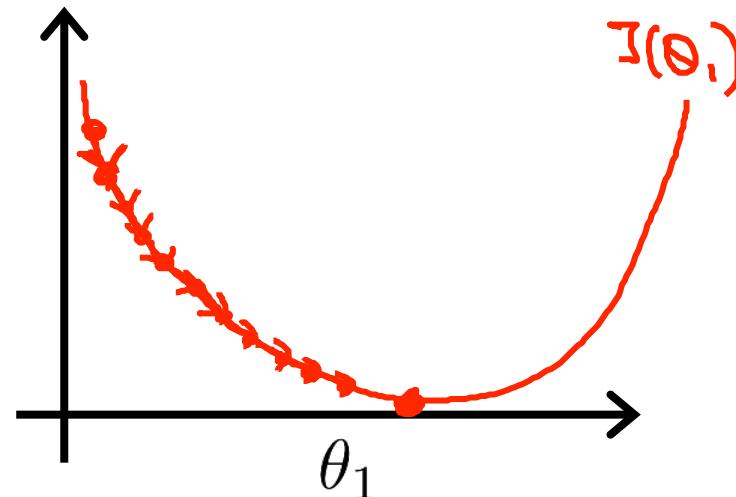


$$\frac{\frac{\partial}{\partial \theta_1} J(\theta_1)}{\leq 0}$$

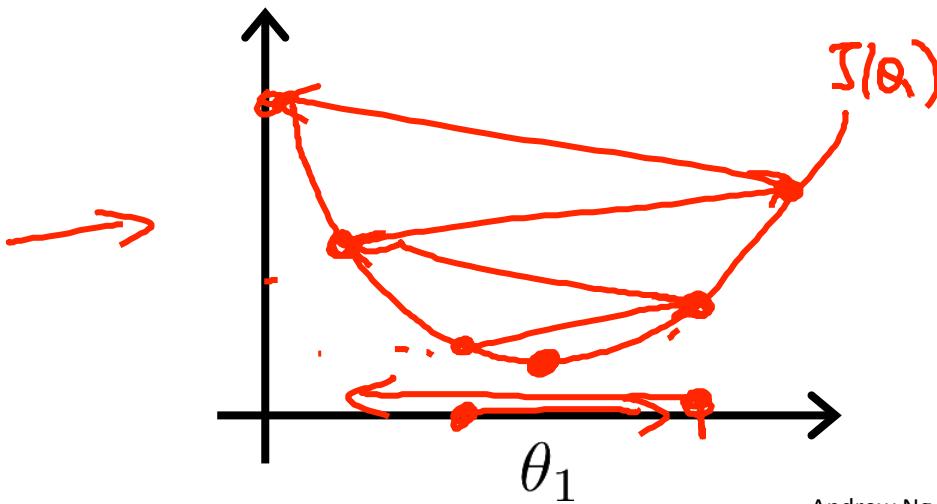
$$\theta_1 := \theta_1 - \frac{\alpha}{\frac{\partial}{\partial \theta_1} J(\theta_1)} \cdot (\text{negative number})$$

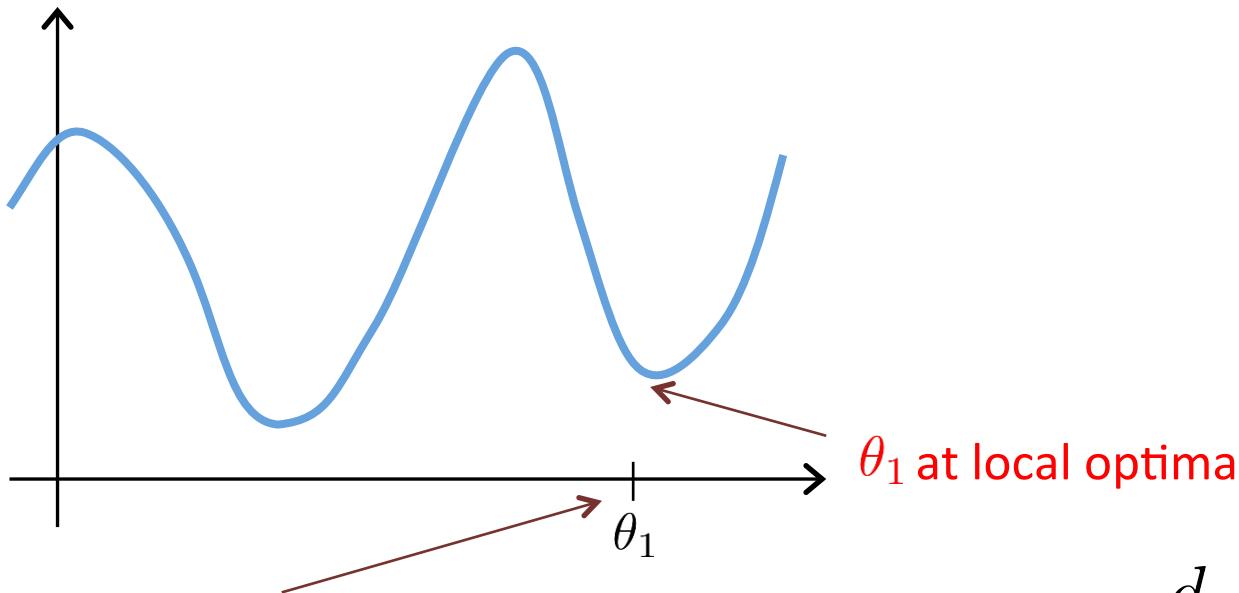
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



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



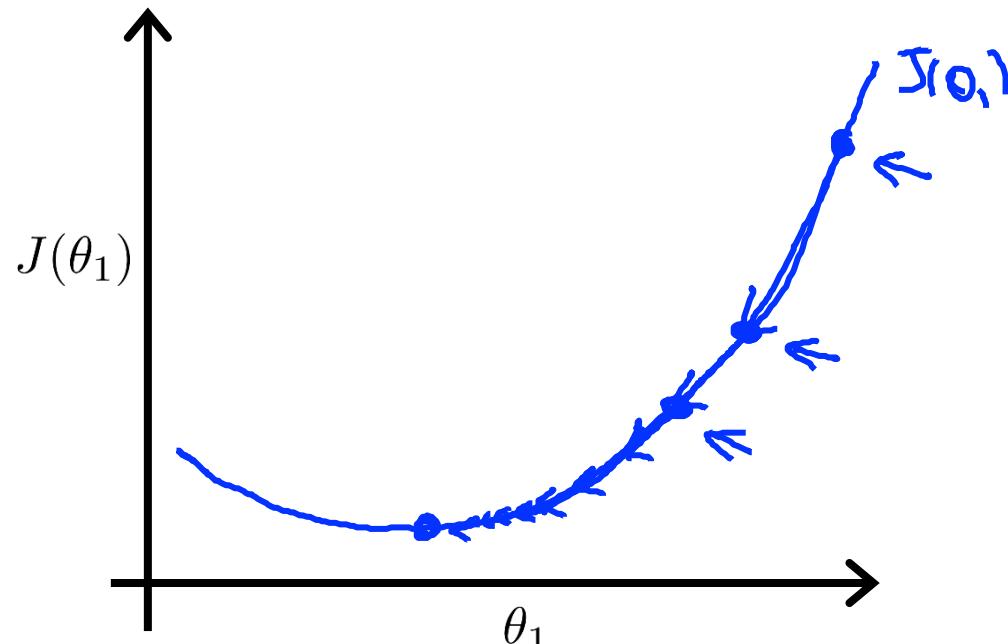


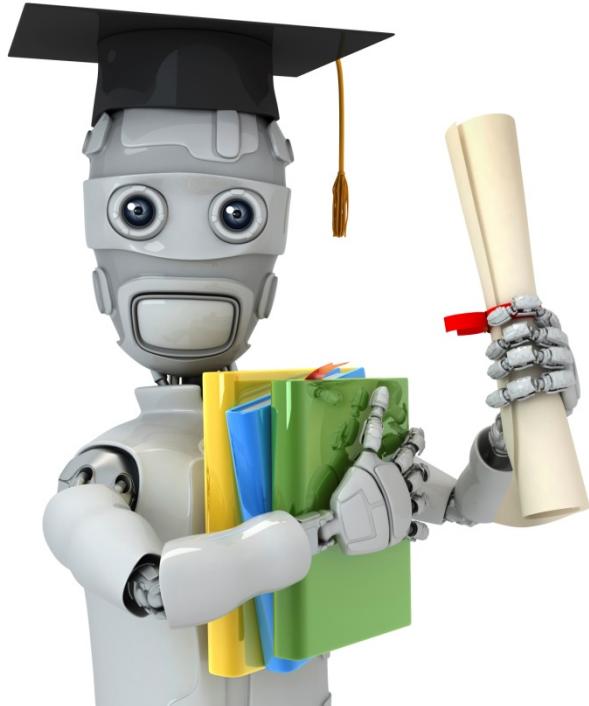
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

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

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

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





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

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

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

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

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

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

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

Gradient descent algorithm

repeat until convergence {

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

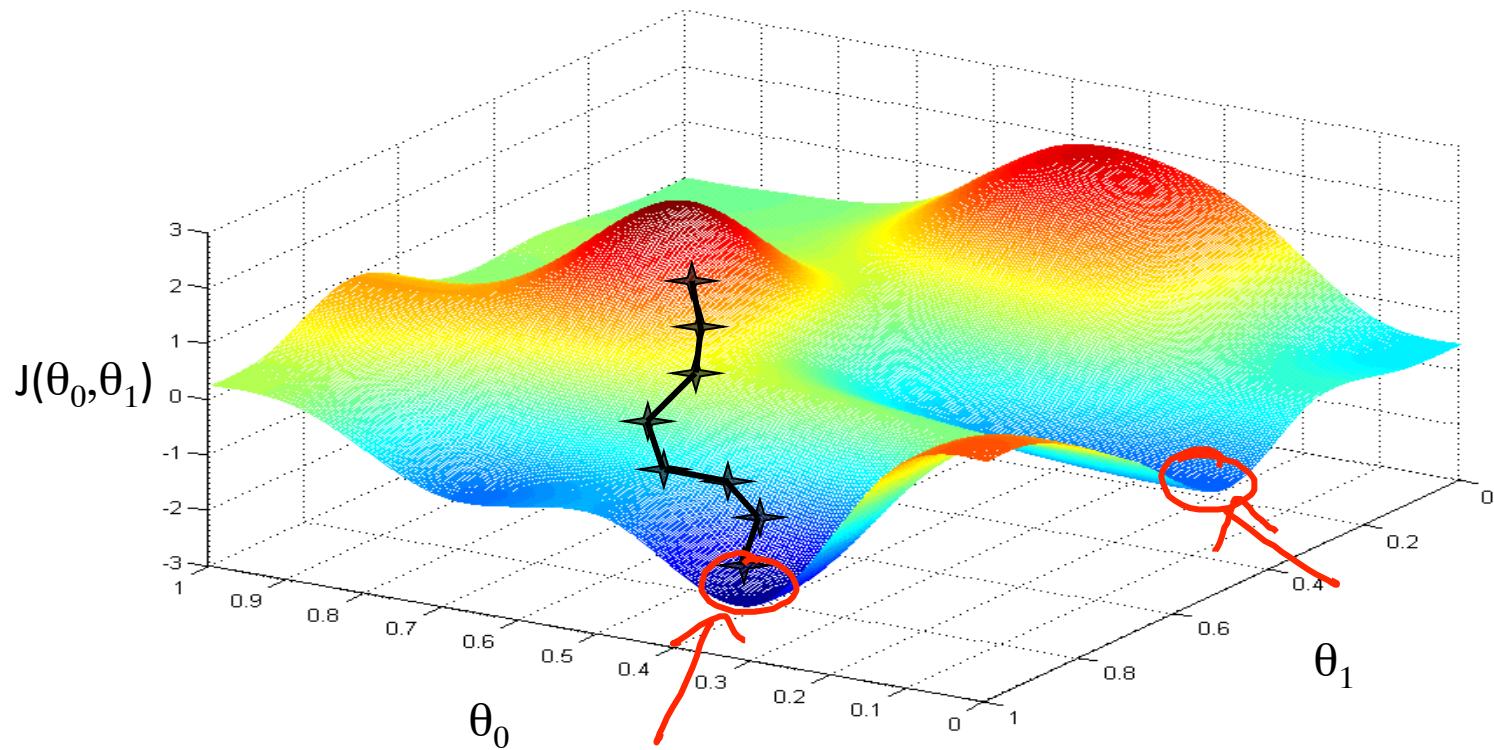
$$\theta_1 := \theta_1 - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \right]$$

}

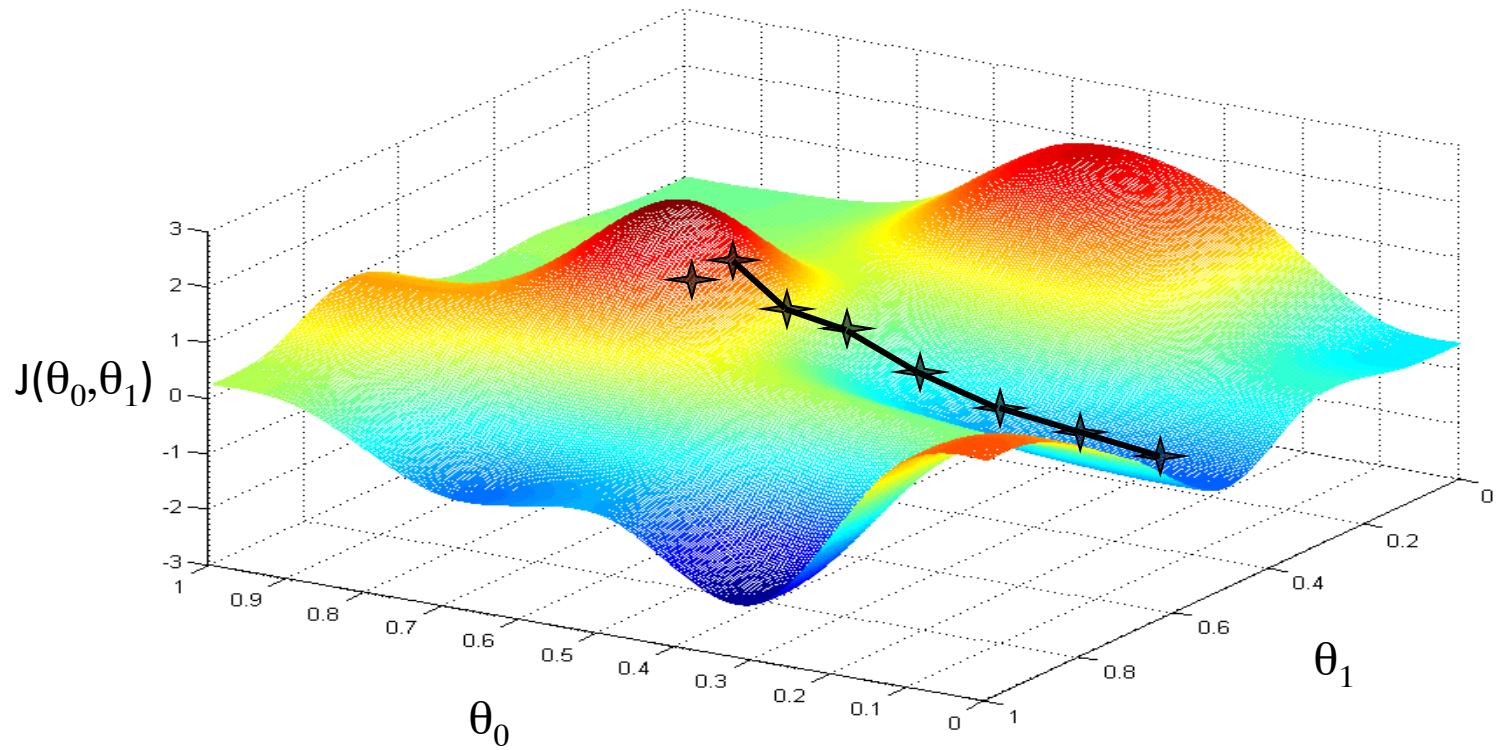
$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

update
 θ_0 and θ_1
simultaneously

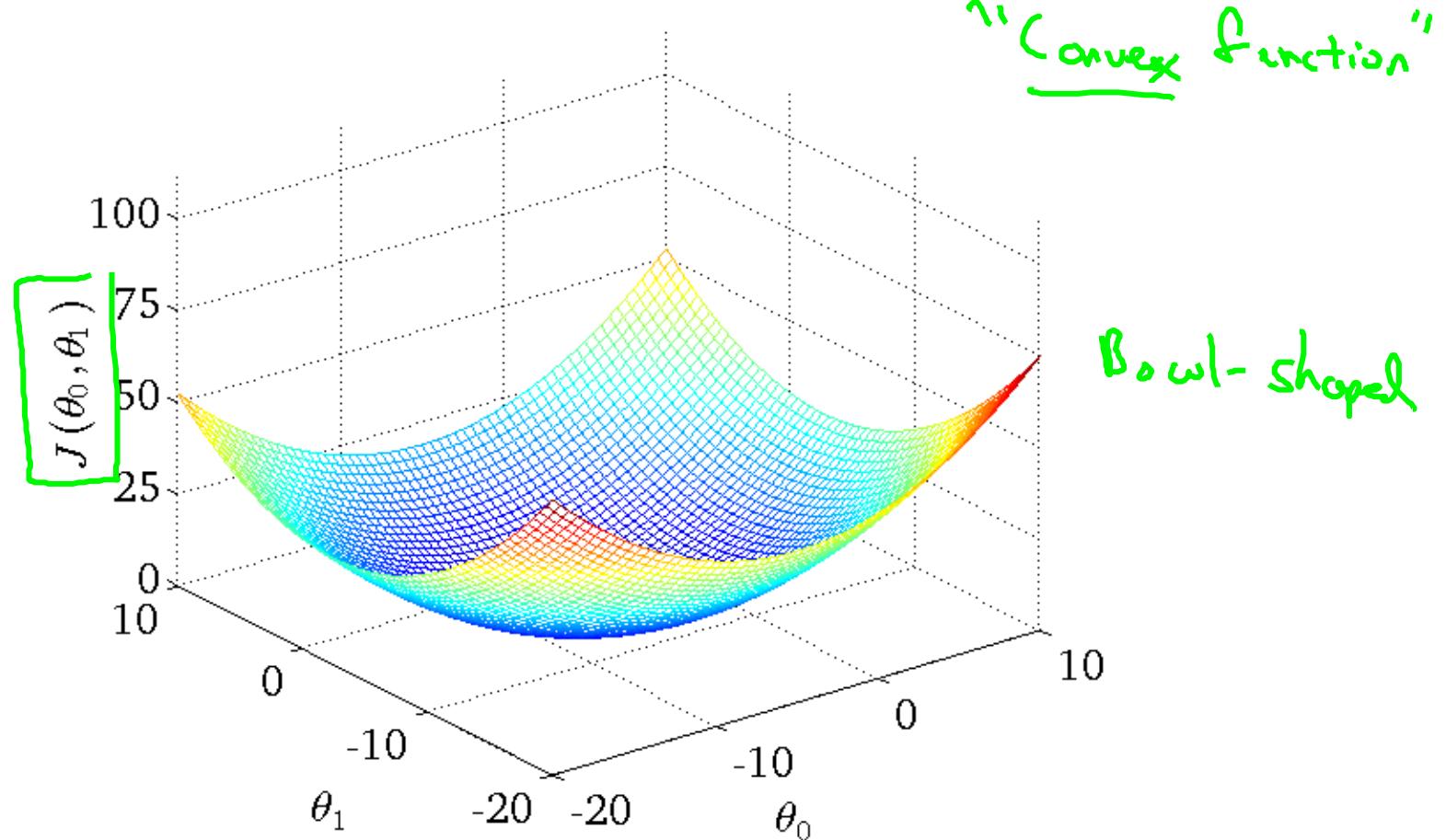
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$



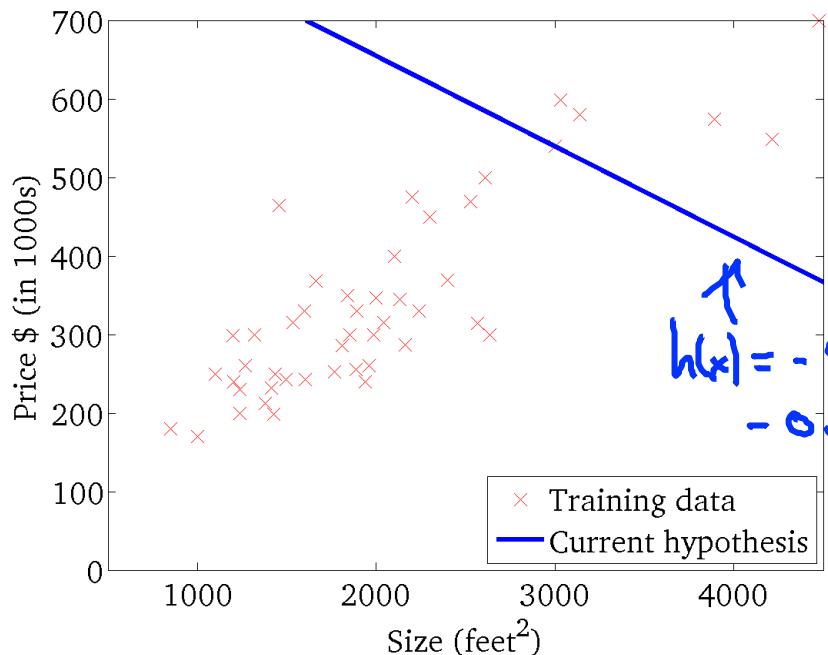
Andrew Ng



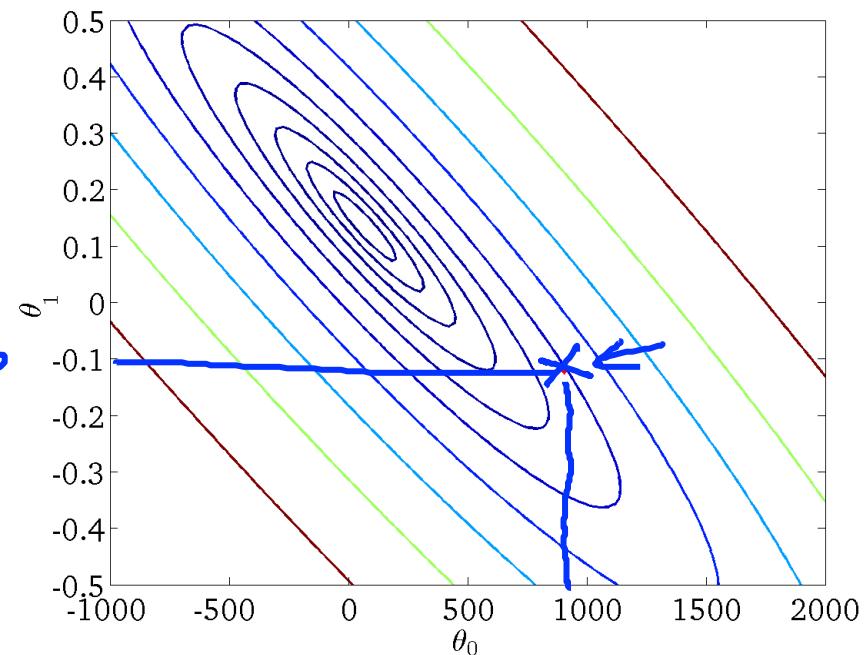
Andrew Ng



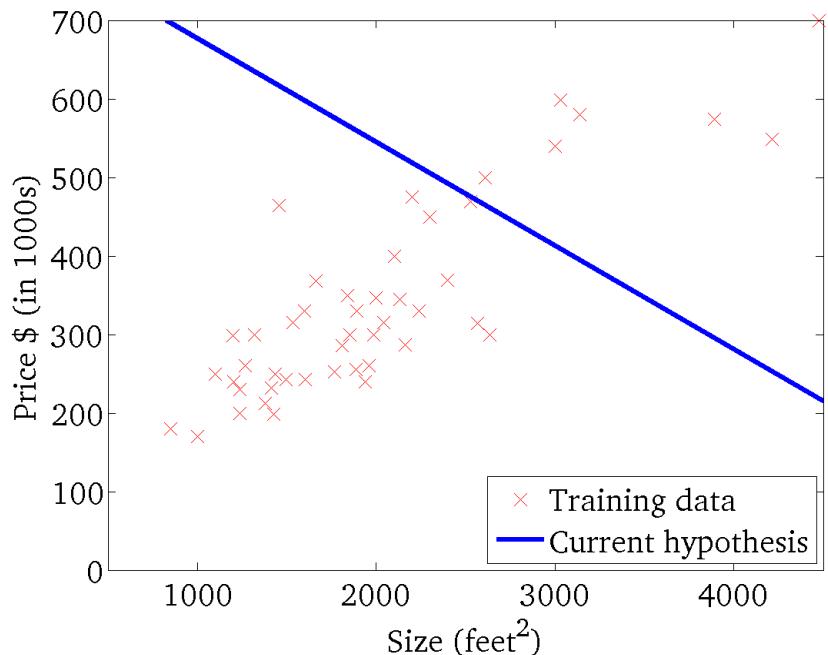
$\underline{h_{\theta}(x)}$
 (for fixed θ_0, θ_1 , this is a function of x)



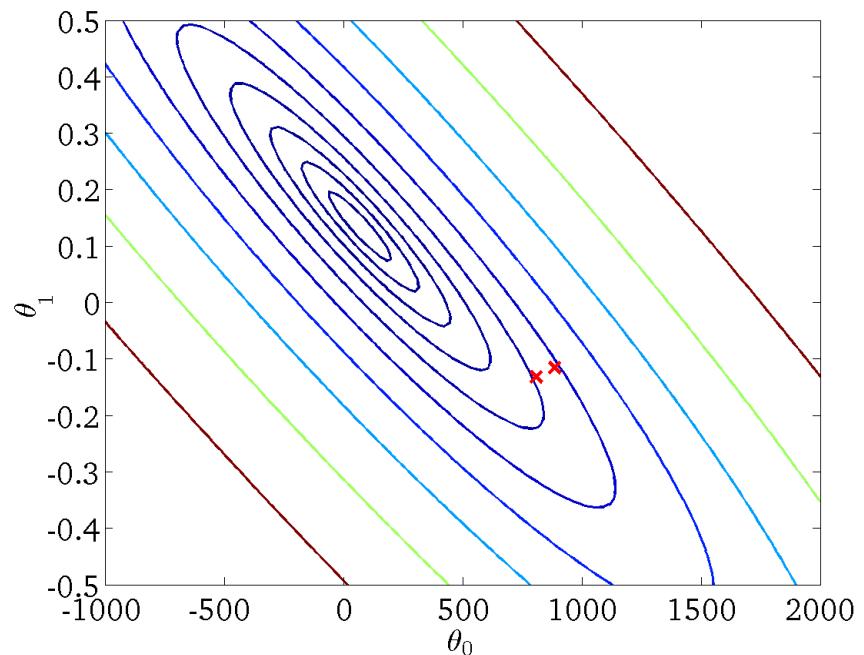
$\underline{J(\theta_0, \theta_1)}$
 (function of the parameters θ_0, θ_1)



$h_{\theta}(x)$
(for fixed θ_0, θ_1 , this is a function of x)

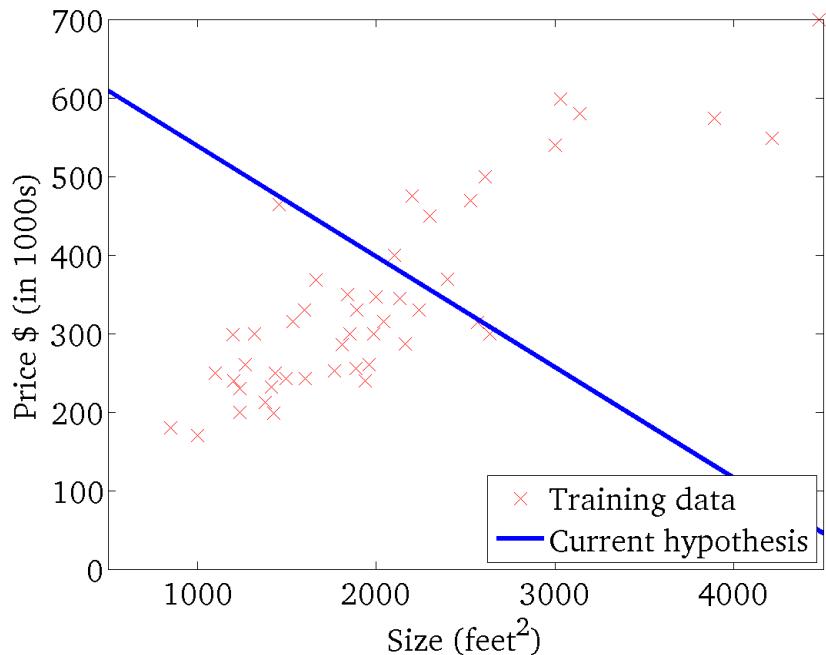


$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)



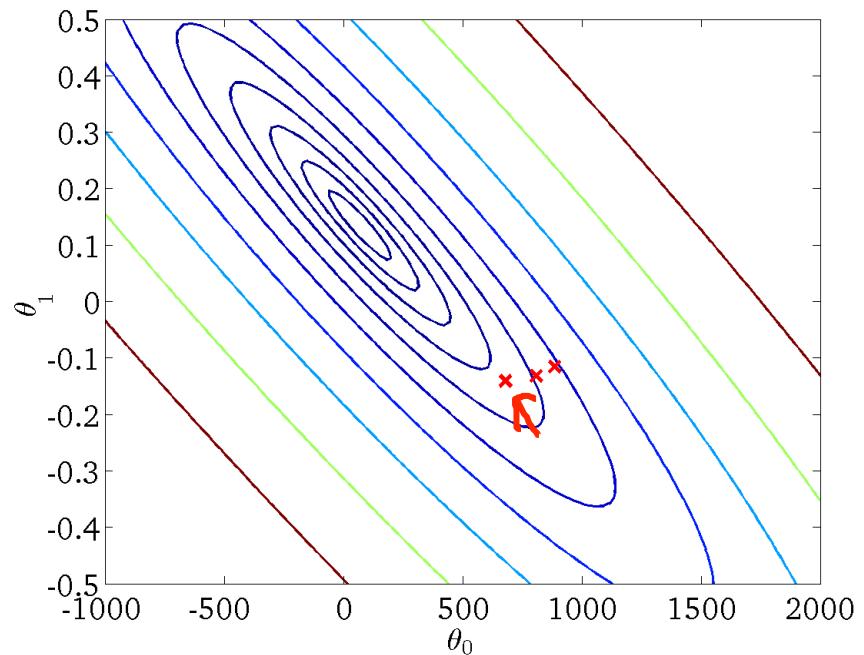
$$h_{\theta}(x)$$

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



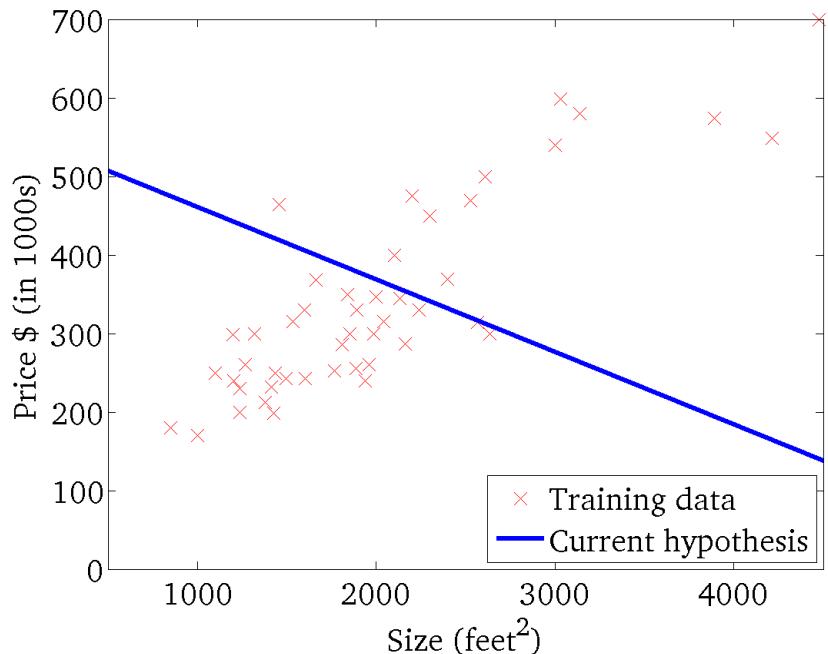
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



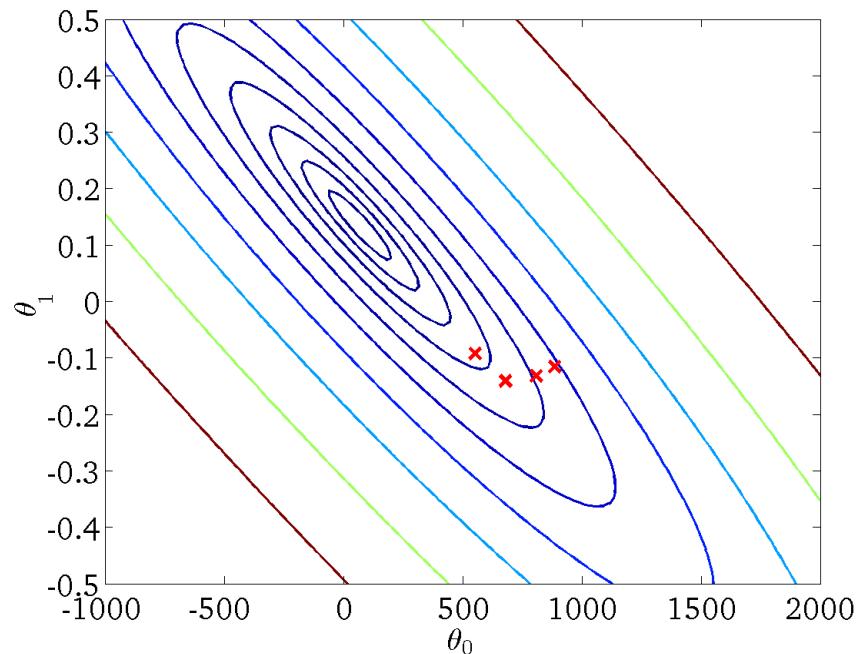
$$h_{\theta}(x)$$

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



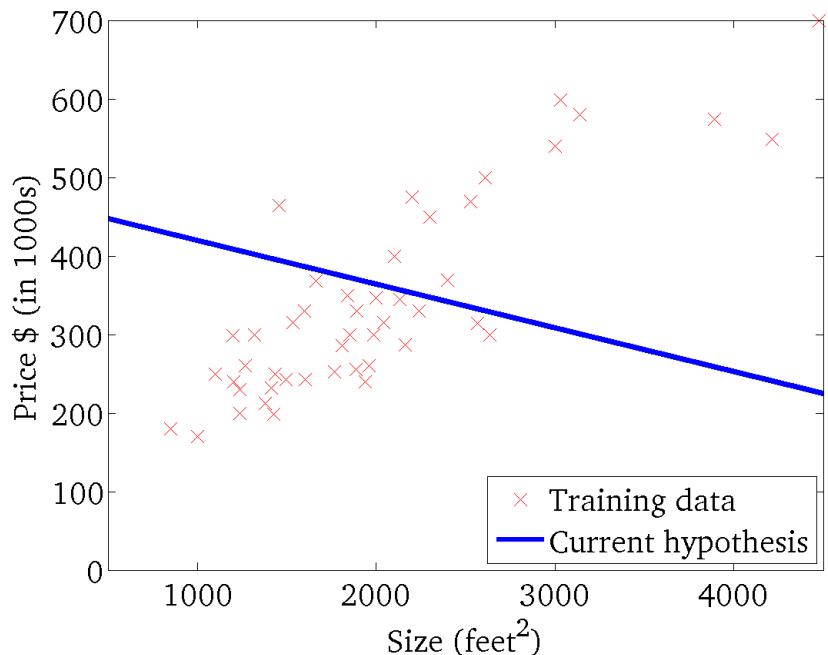
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



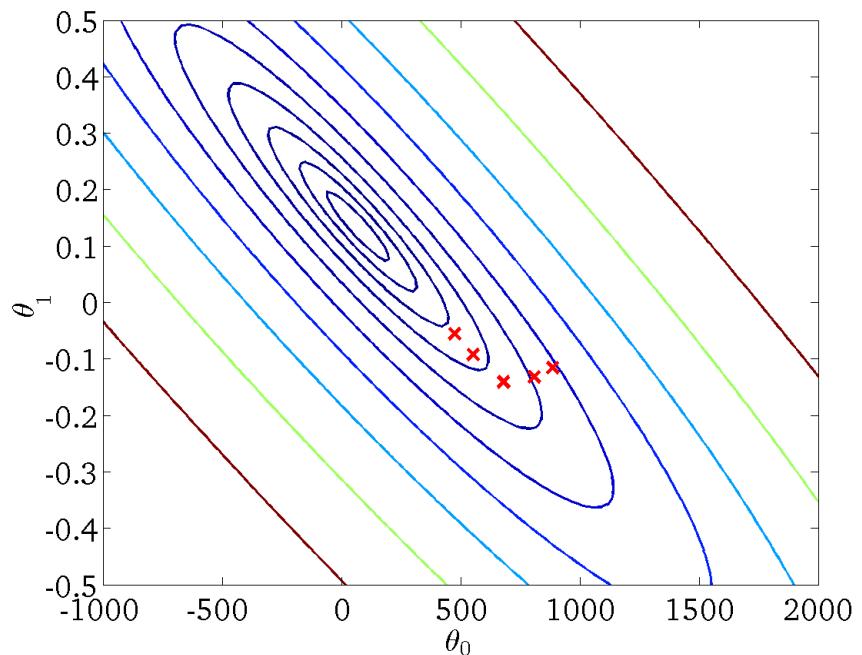
$$h_{\theta}(x)$$

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



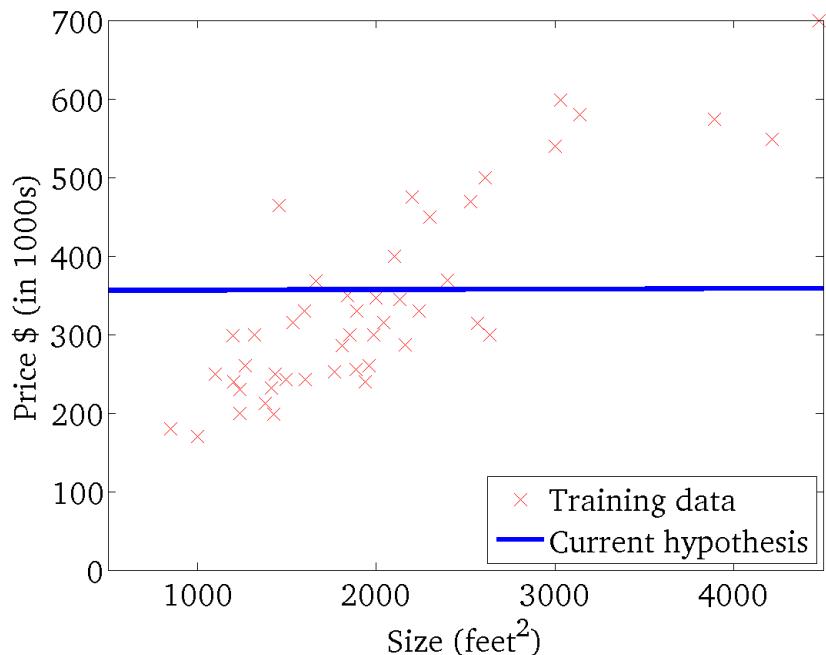
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



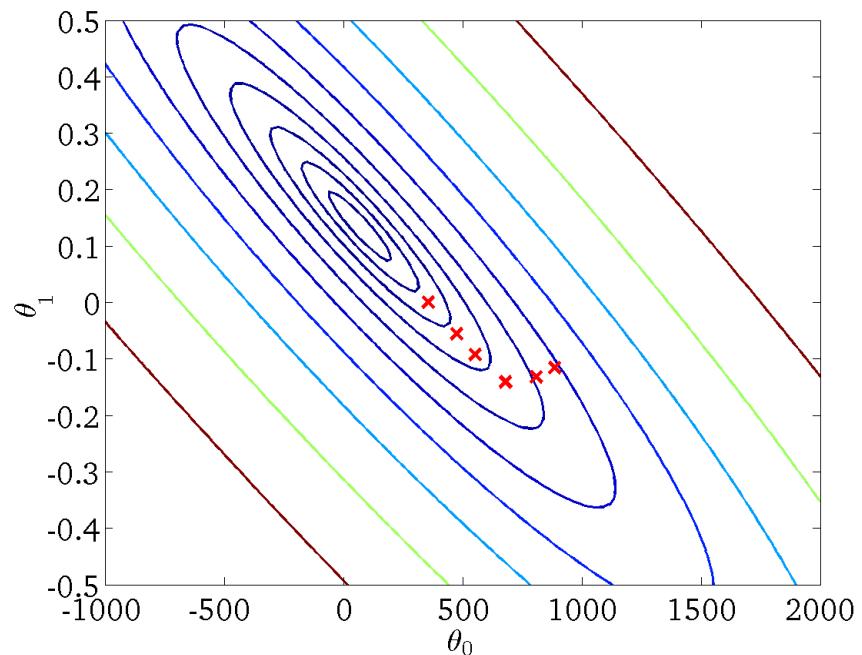
$$h_{\theta}(x)$$

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



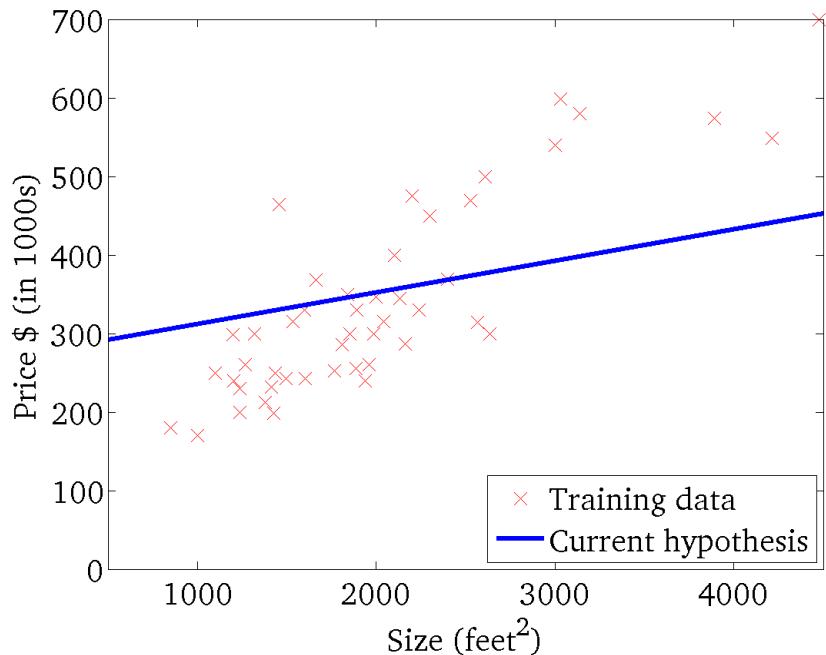
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



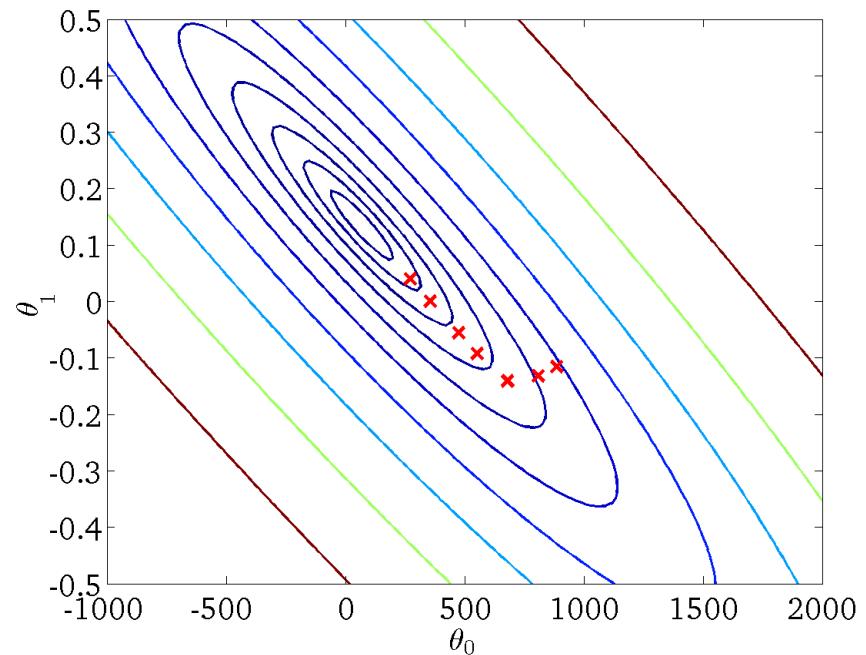
$$h_{\theta}(x)$$

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

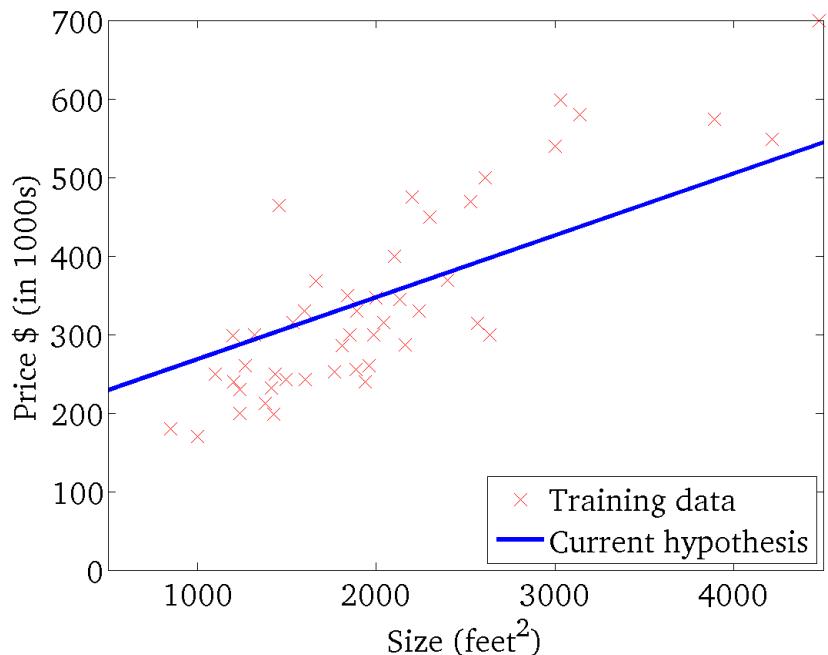


$$J(\theta_0, \theta_1)$$

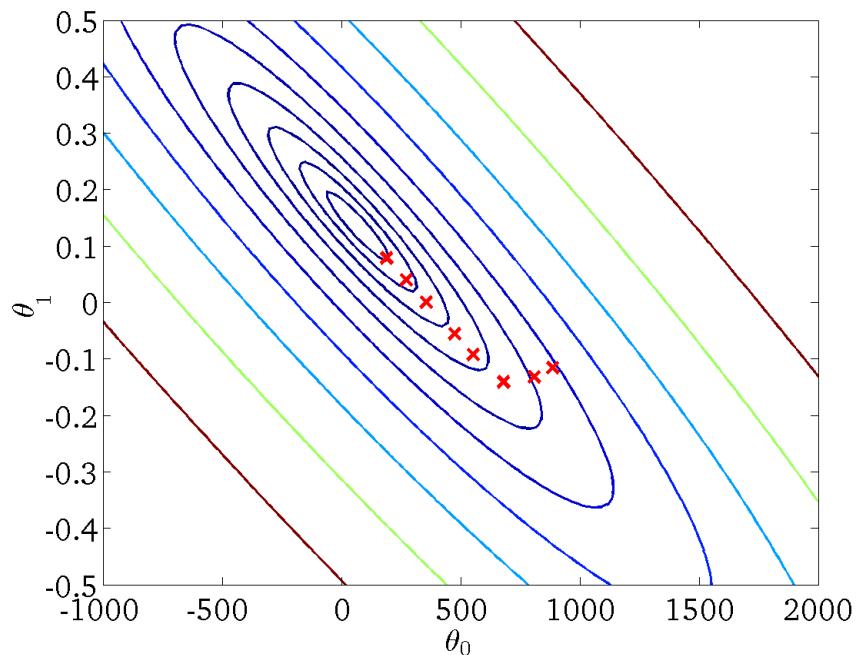
(function of the parameters θ_0, θ_1)



$h_{\theta}(x)$
(for fixed θ_0, θ_1 , this is a function of x)

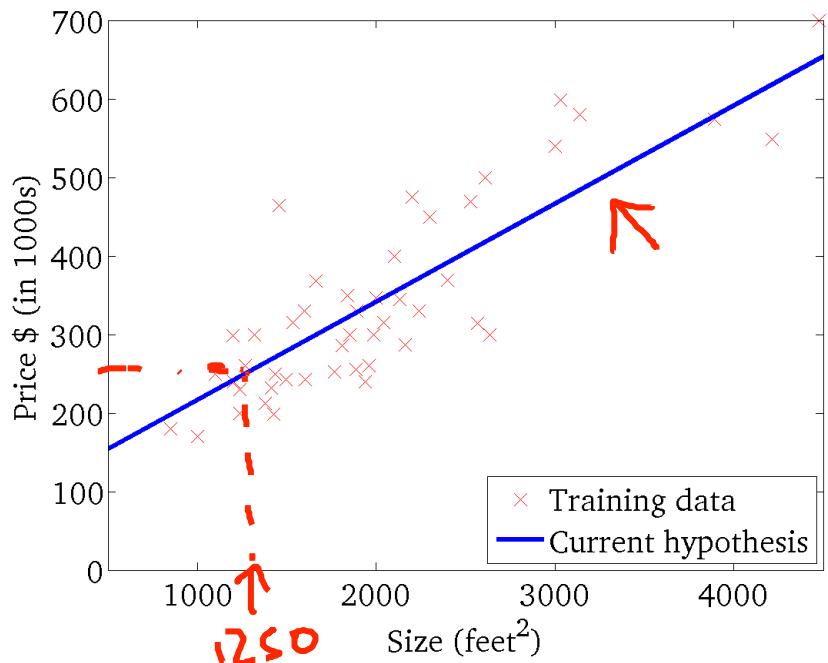


$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)



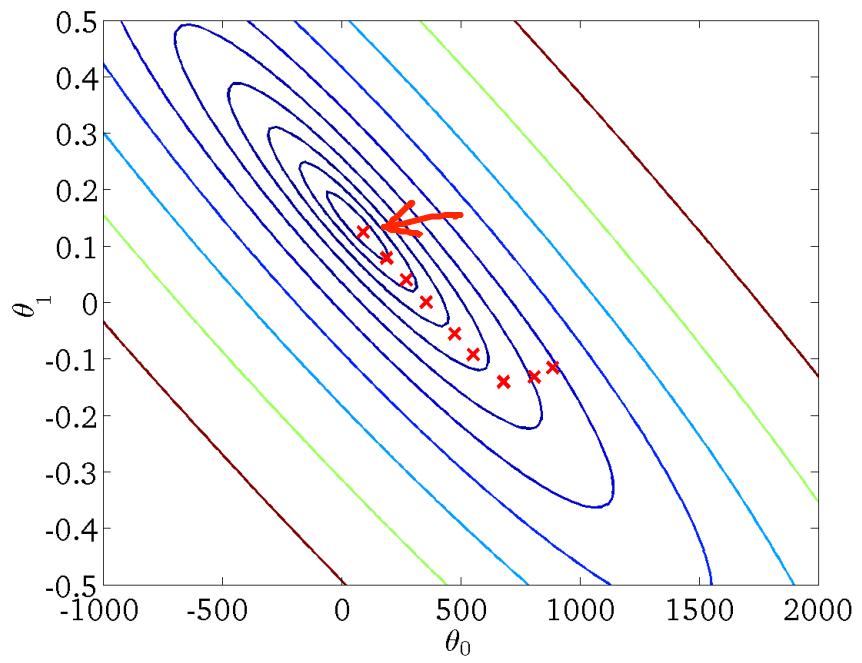
$$h_{\theta}(x)$$

(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)

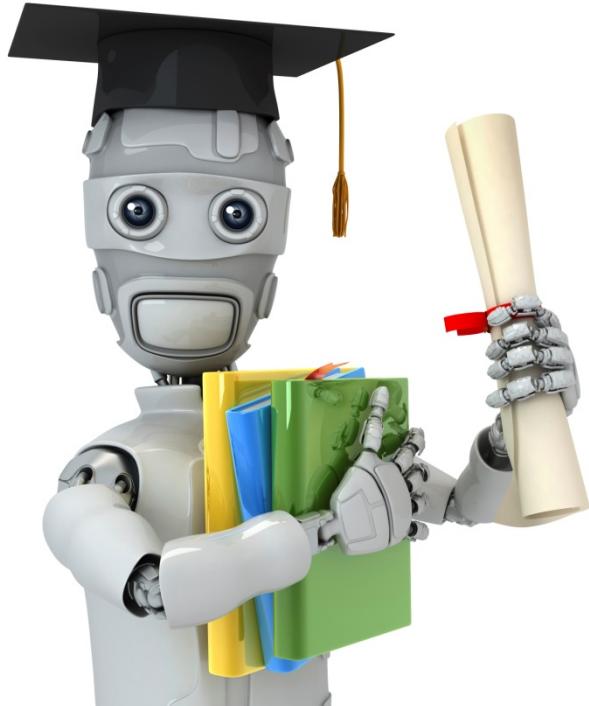


“Batch” Gradient Descent

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

$$\xrightarrow{\text{all}} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$$

1.3 Linear Algebra



Machine Learning

Linear Algebra review (optional)

Matrices and vectors

Matrix: Rectangular array of numbers:

$$\begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \\ \nearrow \quad \uparrow \quad \uparrow \end{array} \left[\begin{array}{cc} 1402 & 191 \\ 1371 & 821 \\ 949 & 1437 \\ 147 & 1448 \end{array} \right]$$

4×2 matrix

$$\rightarrow \boxed{\mathbb{R}^{4 \times 2}}$$

$$2 \rightarrow \left[\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \end{array} \right] \quad \begin{matrix} \uparrow & \uparrow & \uparrow \\ 3 & & \end{matrix}$$

2×3 matrix

$$\boxed{\mathbb{R}^{2 \times 3}}$$

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.

$$A_{11} = 1402$$

$$A_{12} = 191$$

$$A_{\underline{3}\underline{2}} = 1437$$

$$A_{41} = 147$$

~~A_{43}~~ = Undefined (error)

Vector: An $n \times 1$ matrix.

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \quad \begin{matrix} \nearrow & \searrow \\ n=4 \end{matrix} \quad \leftarrow \text{4-dimensional vector}$$

~~$\mathbb{R}^{3 \times 2}$~~

\mathbb{R}^4

$y_i = i^{th}$ element

$$y_1 = 460$$

$$y_2 = 232$$

$$y_3 = 315$$

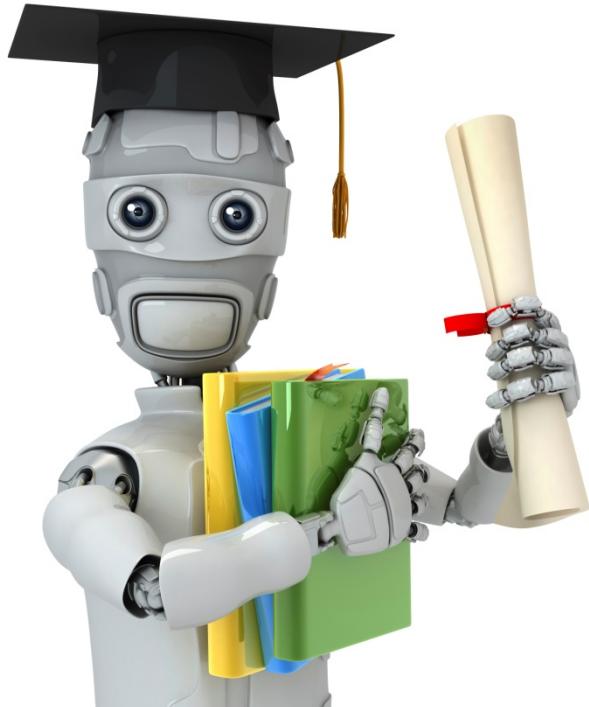
$\rightarrow [A, B, C, X]$

a, b, x, y

1-indexed vs 0-indexed:

$$y[1] \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \quad \begin{matrix} \leftarrow \\ \leftarrow \\ \leftarrow \\ \leftarrow \end{matrix} \quad \boxed{\text{1-indexed}}$$

$$y[0] \quad y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad \begin{matrix} \leftarrow \\ \leftarrow \\ \leftarrow \\ \leftarrow \end{matrix} \quad \boxed{\text{0-indexed}}$$



Machine Learning

Linear Algebra review (optional)

Addition and scalar
multiplication

Matrix Addition

$$\begin{array}{c}
 \downarrow \quad \downarrow \\
 \left[\begin{array}{cc} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{array} \right] + \left[\begin{array}{cc} 4 & 0.5 \\ 2 & 5 \\ 0 & 1 \end{array} \right] = \left[\begin{array}{cc} 5 & 0.5 \\ 4 & 10 \\ 3 & 2 \end{array} \right]
 \end{array}$$

3×2

3×2

3×2

A hand-drawn diagram illustrating matrix multiplication. It shows two matrices, A (3x2) and B (2x2), being multiplied to produce matrix C (3x2). Matrix A has columns [1, 0] and [2, 5]. Matrix B has columns [4, 0.5] and [2, 5]. The resulting matrix C has columns [1, 0] and [2, 5]. A plus sign is placed between the two matrices to indicate addition.

Scalar Multiplication

↙ real number

$$3 \times \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} = \boxed{\begin{bmatrix} 3 & 0 \\ 6 & 15 \\ 9 & 3 \end{bmatrix}} = \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} \times 3$$
$$\frac{1}{4} \begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} / 4 = \frac{1}{4} \begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{3}{2} & \frac{3}{4} \end{bmatrix}$$

Combination of Operands

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

Scalar multiplication

Scalar division

$$= \begin{bmatrix} 3 \\ 12 \\ 6 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ \frac{2}{3} \end{bmatrix}$$

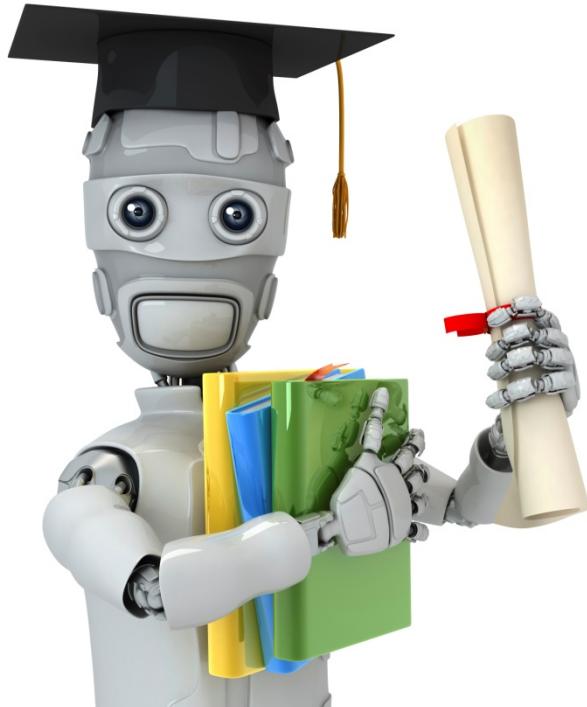
matrix subtraction / vector subtraction

matrix addition / vector addition

$$= \begin{bmatrix} 2 \\ 12 \\ 10 \frac{1}{3} \end{bmatrix}$$

3x1 matrix

3-dimensional vector



Machine Learning

Linear Algebra review (optional)

Matrix-vector multiplication

Example

$$\begin{matrix} & \begin{matrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{matrix} \\ \underbrace{\quad\quad}_{3 \times 2} & \end{matrix} \quad \begin{matrix} 1 \\ 5 \end{matrix} = \begin{bmatrix} 16 \\ 4 \\ 7 \end{bmatrix}$$

3x1 matrix

$$1 \times 1 + 3 \times 5 = 16$$

$$4 \times 1 + 0 \times 5 = 4$$

$$2 \times 1 + 1 \times 5 = 7$$

Details:

$$\underbrace{A}_{\substack{\text{m} \times \text{n} \text{ matrix} \\ (\text{m rows}, \\ \text{n columns})}} \times \underbrace{x}_{\substack{\text{n} \times 1 \text{ matrix} \\ (\text{n-dimensional} \\ \text{vector})}} = \underbrace{y}_{\substack{\text{m-dimensional} \\ \text{vector}}}$$

The diagram illustrates the multiplication of a matrix A by a vector x to produce a vector y . Matrix A is shown as a stack of n horizontal rows, each represented by a blue oval. Vector x is shown as a single vertical column with blue arrows pointing upwards. The resulting vector y is also a vertical column with blue arrows pointing upwards. Handwritten annotations in blue highlight the dimensions: "m x n matrix (m rows, n columns)" under matrix A , "n x 1 matrix (n-dimensional vector)" under vector x , and "m-dimensional vector" under vector y .

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

Example

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

3×4

$$\begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 14 \\ 13 \\ -7 \end{bmatrix} = \begin{bmatrix} 14 \\ 13 \\ -7 \end{bmatrix}$$

$$1 \times 1 + 2 \times 3 + 1 \times 2 + 5 \times 1 = 14$$
$$0 \times 1 + 3 \times 3 + 0 \times 2 + 4 \times 1 = 13$$
$$-1 \times 1 + (-2) \times 3 + 0 \times 2 + 0 \times 1 = -7$$

House sizes:

→ 2104

→ 1416

→ 1534

→ 852

Matrix x

1	2104
1	1416
1	1534
1	852

{

4x2

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

$h_{\theta}(x)$

2x1

Vector

-40
0.25

X

4x1 matrix

$-40 \times 1 + 0.25 \times 2104$
$-40 \times 1 + 0.25 \times 1416$

$h_{\theta}(2104)$

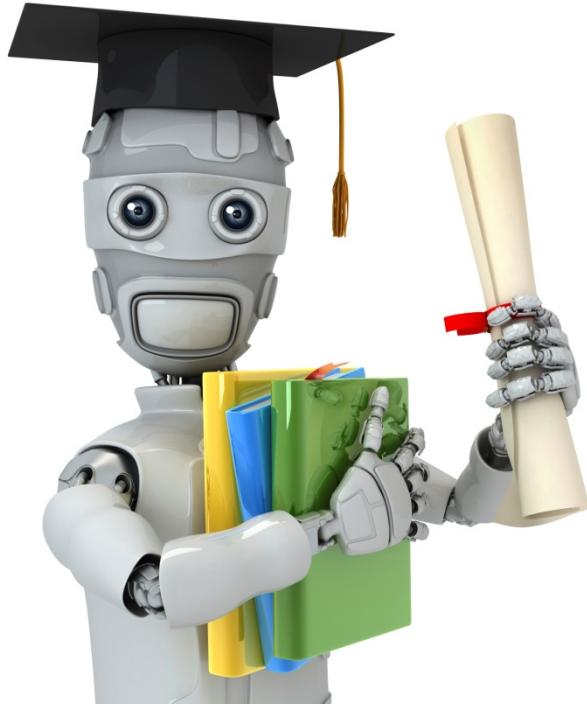
$h_{\theta}(1416)$

$\boxed{\text{prediction}} = \boxed{\text{Data Matrix}} \times \boxed{\text{Parameters}}$

4x1

n

for $i = 1: 1000$,
 $\text{prediction}(i) := \dots$



Machine Learning

Linear Algebra review (optional)

Matrix-matrix multiplication

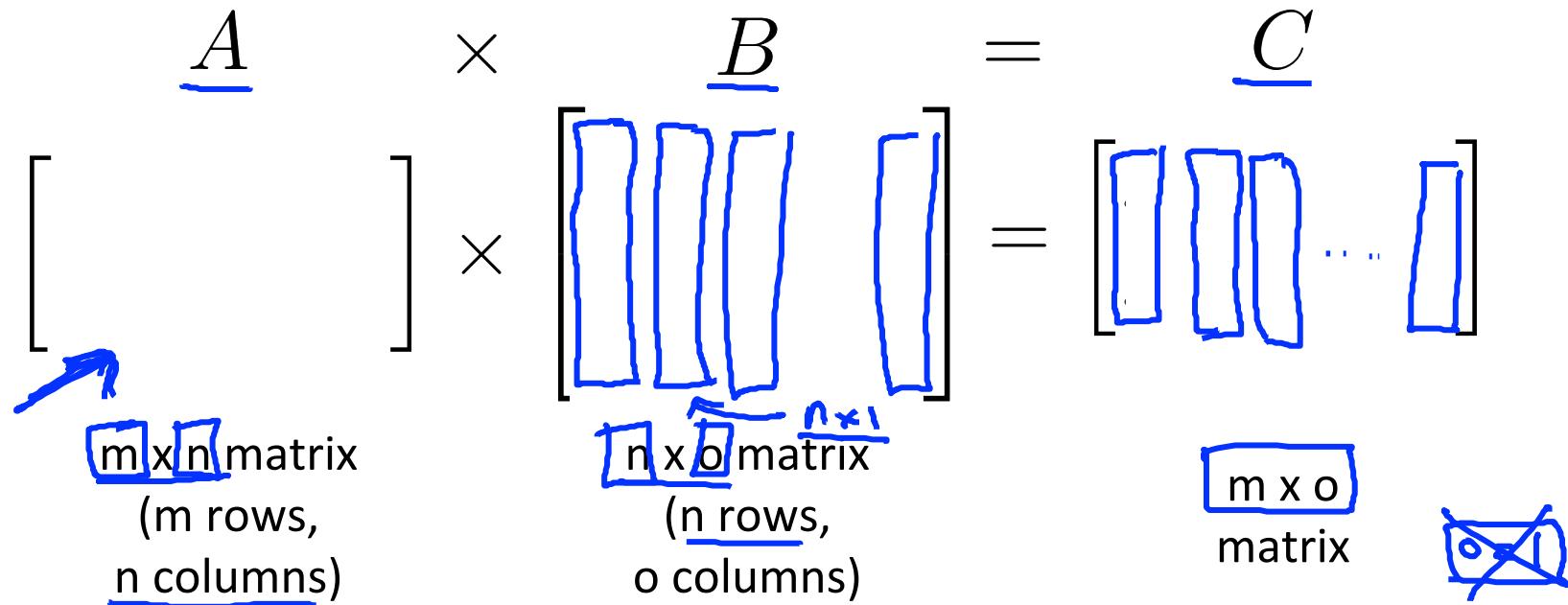
Example

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 11 \\ 9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 14 \end{bmatrix}$$

Details:



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

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 9 & 7 \\ 15 & 12 \end{bmatrix}$$

Diagram illustrating the multiplication of two matrices:

- The first matrix is 2×2 and the second is 2×2 .
- The result is a 2×2 matrix.
- Handwritten annotations show the calculation:
 - Top-left element: $1 \times 0 + 3 \times 3 = 9$
 - Top-right element: $2 \times 0 + 5 \times 3 = 15$
 - Bottom-left element: $1 \times 1 + 3 \times 2 = 7$
 - Bottom-right element: $2 \times 1 + 5 \times 2 = 12$
- Arrows point from the elements of the first matrix to the columns of the second matrix, and from the columns of the second matrix to the resulting elements.

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 1 \times 0 + 3 \times 3 \\ 2 \times 0 + 5 \times 3 \end{bmatrix} = \begin{bmatrix} 9 \\ 15 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \times 1 + 3 \times 2 \\ 2 \times 1 + 5 \times 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 12 \end{bmatrix}$$

House sizes:

$$\left\{ \begin{array}{r} 2104 \\ 1416 \\ 1534 \\ \hline 852 \end{array} \right.$$

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

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

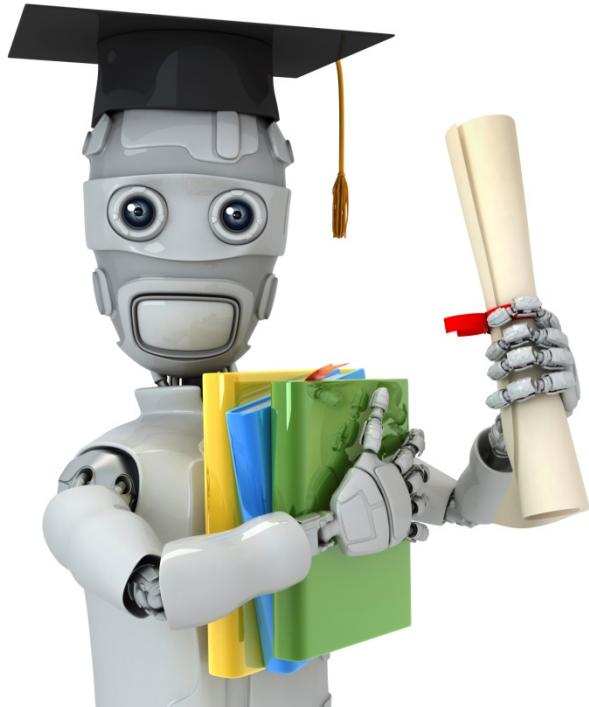
Matrix

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

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

↑
Prediction
of 1st
 h_{θ}

↑
Predictions
of 2nd
 h_{θ}



Machine Learning

Linear Algebra review (optional)

Matrix multiplication properties

$$\begin{matrix} 3 \times 5 \\ \curvearrowleft \end{matrix} = 5 \times 3$$

"Commutative"

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

E.g.

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

$$\begin{matrix} A \times B \\ m \times n \end{matrix}$$

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

$$\begin{matrix} A \times B \rightarrow m \times n \\ B \times A \rightarrow n \times m \end{matrix}$$



$$\underline{3 \times 5 \times 2}$$

$$3 \times 10 = 30 = 15 \times 2$$

$$3 \times (5+2) = (3 \times 5) + 2$$

"Associative"

$$\begin{array}{c} A \times (B \times C) \\ (A \times B) \times C \end{array}$$

$$A \times B \times C.$$

Let $D = B \times C$.

Compute $A \times D$.

Let $E = A \times B$.

Compute $E \times C$.

$\begin{array}{c} A \times (B \times C) \\ (A \times B) \times C \end{array}$

Some answer.

Identity Matrix

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

Examples of identity matrices:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad 1 \times 1$$
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad 2 \times 2$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 3 \times 3$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 4 \times 4$$

1 is identity

$$1 \times z = z \times 1 = z$$

for any z

Informally:

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & \dots \end{bmatrix} \leftarrow$$

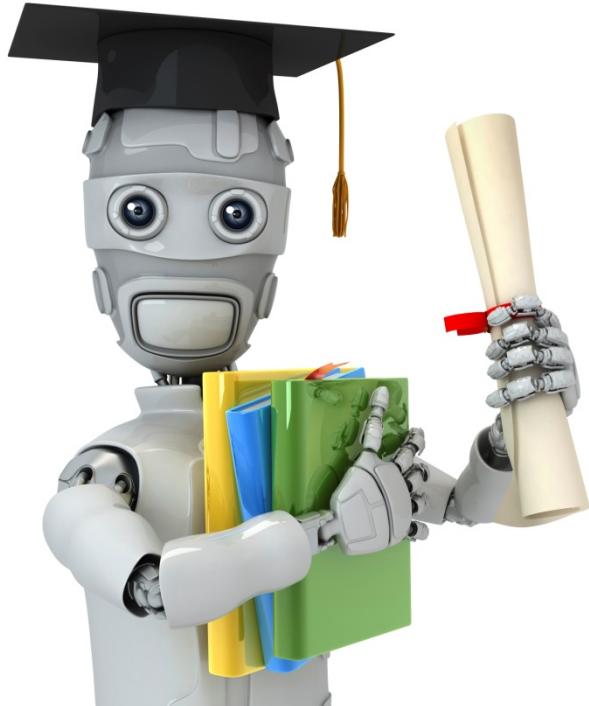
For any matrix A ,

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

$\begin{matrix} \nearrow mxn & \nearrow nxn & \nearrow mxm & \nearrow mxn & \nearrow mxn \end{matrix}$

$$I_{n \times n}$$

Note:
 $AB \neq BA$ in general
 $AI = IA$ ✓



Machine Learning

Linear Algebra review (optional)

Inverse and transpose

I = "identity."

$$3 \begin{pmatrix} 3^{-1} \\ \frac{1}{3} \end{pmatrix} = 1$$

$$\frac{12 \times (12^{-1})}{\frac{1}{12}} = 1$$

$$0 \begin{pmatrix} 0^{-1} \\ \underline{\quad} \end{pmatrix}$$

undefined

Not all numbers have an inverse.

Matrix inverse:

If A is an m x m matrix, and if it has an inverse,

$$\rightarrow A(A^{-1}) = A^{-1}A = I.$$

Square matrix
(#rows = #columns)

A^{-1}

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

E.g.

$$\begin{bmatrix} 3 & 4 \\ 2 & 16 \end{bmatrix} \underbrace{\qquad}_{2 \times 2} \qquad A$$

$$\begin{bmatrix} 0.4 & -0.1 \\ -0.05 & 0.075 \end{bmatrix} \underbrace{\qquad}_{A^{-1}}$$

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

$A^{-1}A$

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

Matrix Transpose

Example:

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

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

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

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

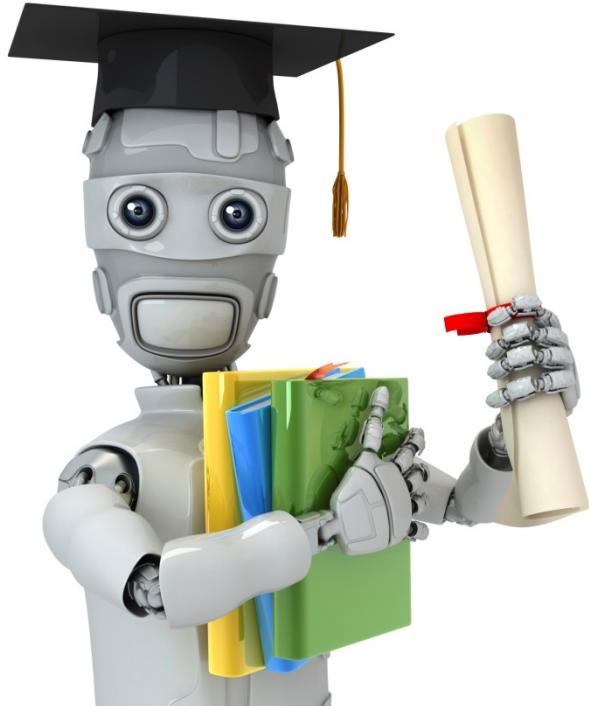
$$B_{ij} = A_{ji}.$$

$$B_{12} = A_{21} = 2$$

$$B_{32} = 9 \quad A_{23} = 9.$$

Chapter 2 Week2

2.1 Multiple Features



Machine Learning

Linear Regression with multiple variables

Multiple features

Multiple features (variables).

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

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

Multiple features (variables).

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

Notation:

- $\rightarrow n = \text{number of features}$ $n=4$
- $\rightarrow x^{(i)} = \text{input (features) of } i^{\text{th}} \text{ training example.}$
- $\rightarrow x_j^{(i)} = \text{value of feature } j \text{ in } i^{\text{th}} \text{ training example.}$

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

$x_3^{(2)} = 2$

$m = 4$

Hypothesis:

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

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

E.g. $\underline{h_{\theta}(x)} = \underline{\underline{\theta_0}} + \underline{\underline{\theta_1}x_1} + \underline{\underline{\theta_2}x_2} + \underline{\underline{\theta_3}x_3} - \underline{\underline{\theta_4}x_4}$

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

For convenience of notation, define $x_0 = 1.$ ($x_0^{(i)} = 1$)

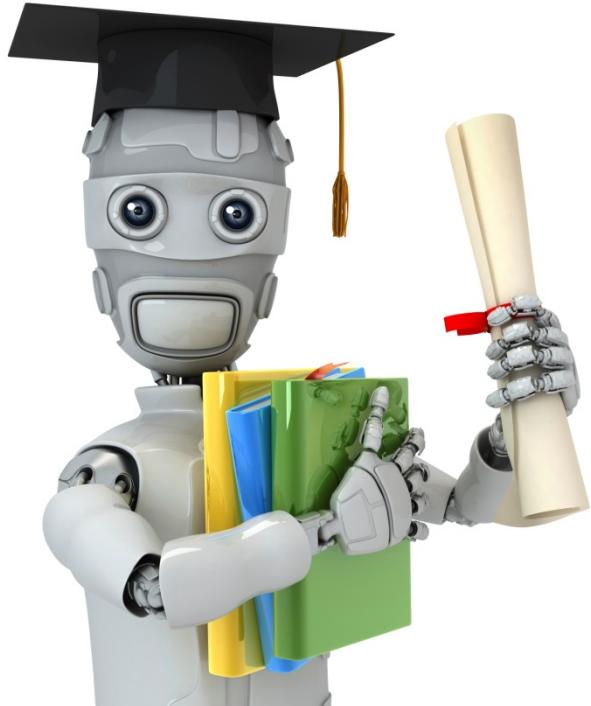
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\begin{aligned} h_{\theta}(x) &= \underline{\theta_0x_0 + \theta_1x_1 + \cdots + \theta_nx_n} \\ &= \boxed{\Theta^T x} \end{aligned}$$

$$\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix} \Theta^T \quad (n+1) \times 1 \text{ matrix} \quad \Theta^T x$$

Multivariate linear regression. 



Machine Learning

Linear Regression with multiple variables

Gradient descent for multiple variables

Hypothesis: $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

$\xrightarrow{x_0 = 1}$

Parameters: $\theta_0, \theta_1, \dots, \theta_n$ Θ $n+1$ -dimensional vector

Cost function:

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

Gradient descent:

Repeat {

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

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

Gradient Descent

Previously (n=1):

Repeat {

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

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

(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 $j = 0, \dots, n$)

}

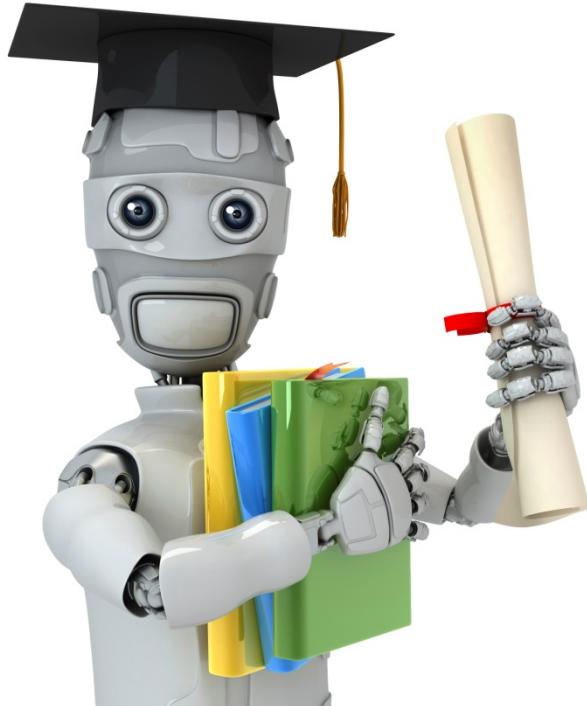
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

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

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

...

Andrew Ng



Machine Learning

Linear Regression with multiple variables

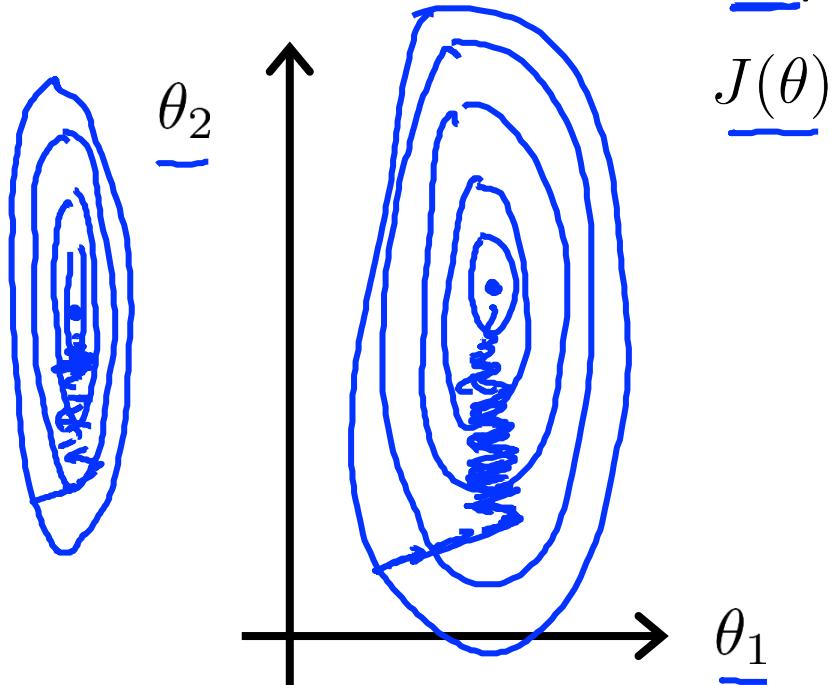
Gradient descent in practice I: Feature Scaling

Feature Scaling

Idea: Make sure features are on a similar scale.

E.g. $x_1 = \text{size } (\underline{0-2000} \text{ feet}^2)$ ←

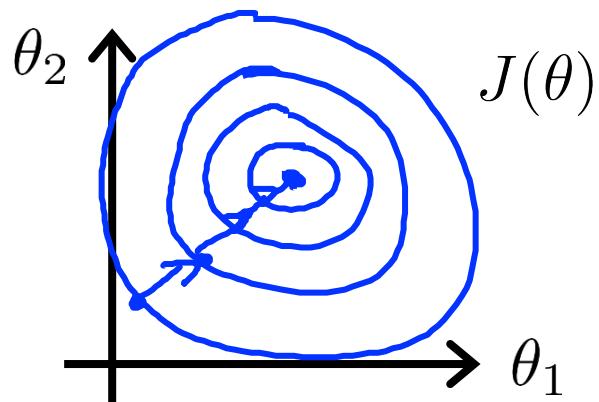
$x_2 = \text{number of bedrooms } (\underline{1-5})$ ←



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

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

$$0 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq 1$$



Feature Scaling

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

$$x_0 = 1$$

$$6 \leq x_1 \leq 3 \quad \checkmark$$

$$-2 \leq x_2 \leq 0.5 \quad \checkmark$$

$$-100 \leq x_3 \leq 100 \quad \times$$

$$-0.0001 \leq x_4 \leq 0.0001 \quad \times$$

$$\boxed{-1 \leq x_i \leq 1}$$

$$-3 \text{ to } 3 \quad \checkmark$$

$$-\frac{1}{3} \text{ to } \frac{1}{3} \quad \checkmark$$

Mean normalization

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

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

Average size = 100

$$x_2 = \frac{\#\text{bedrooms} - 2}{5 - 4}$$

1 - 5 bedrooms

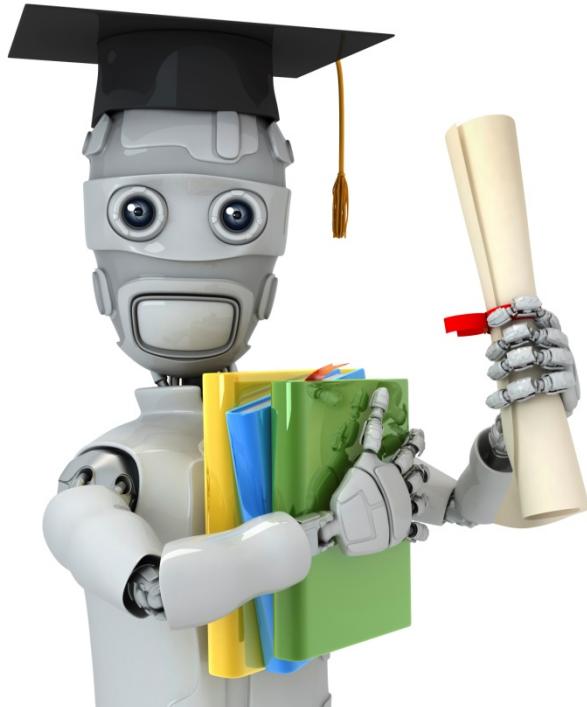
$$\rightarrow [-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5]$$

$$x_1 \leftarrow \frac{x_1 - \mu_1}{\sigma_1}$$

avg value of x_1 in training set

range ($\frac{\max - \min}{\sigma_1}$)
(or standard deviation)

$$x_2 \leftarrow \frac{x_2 - \mu_2}{\sigma_2}$$



Machine Learning

Linear Regression with multiple variables

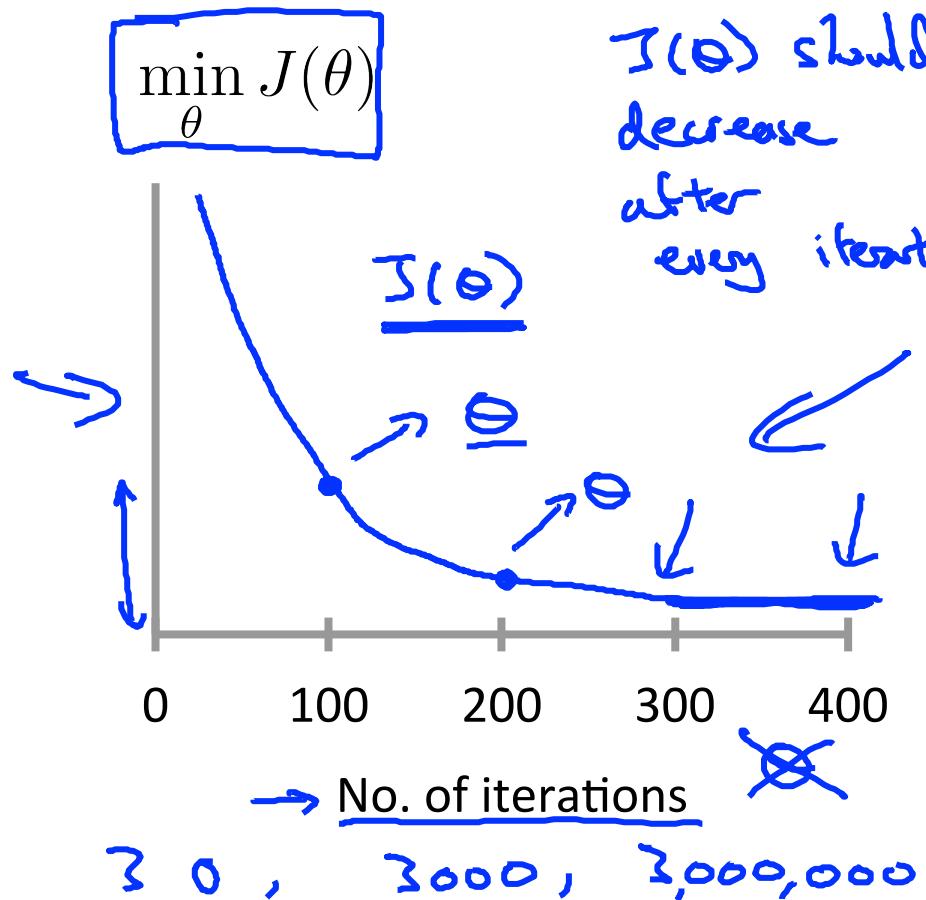
Gradient descent in
practice II: Learning rate

Gradient descent

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

- “Debugging”: How to make sure gradient descent is working correctly.
- How to choose learning rate α .

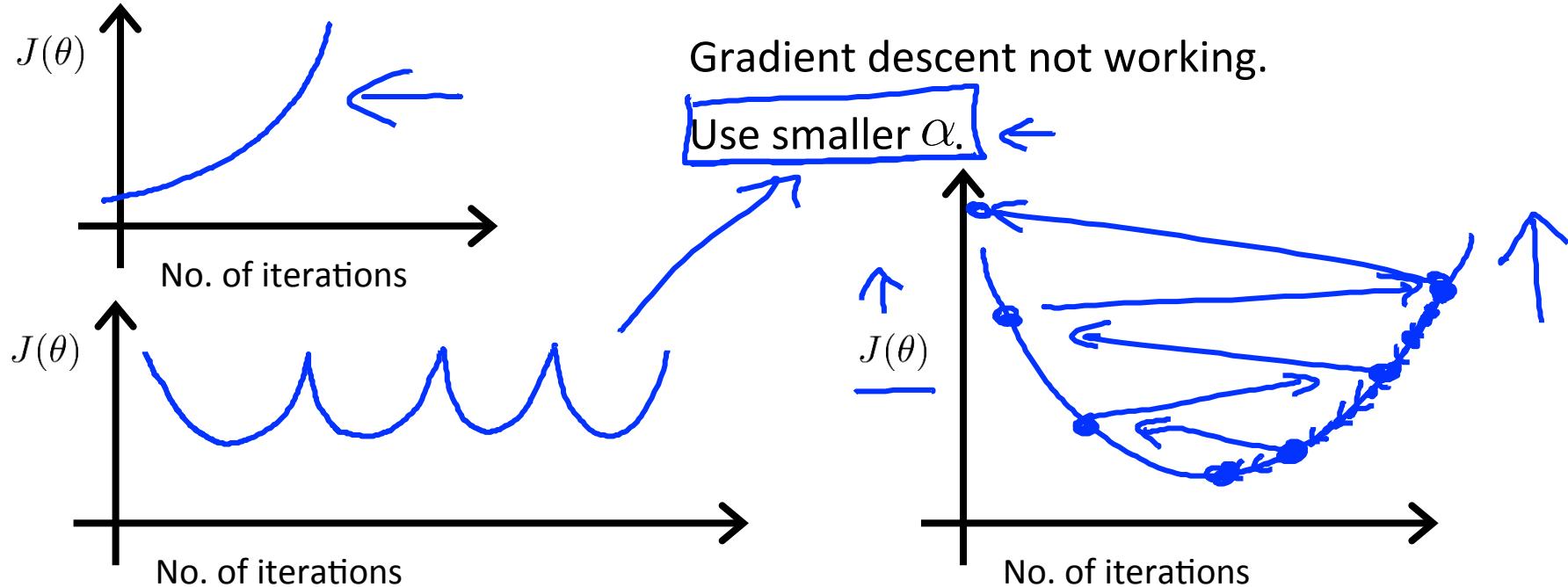
Making sure gradient descent is working correctly.



→ Example automatic convergence test:

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

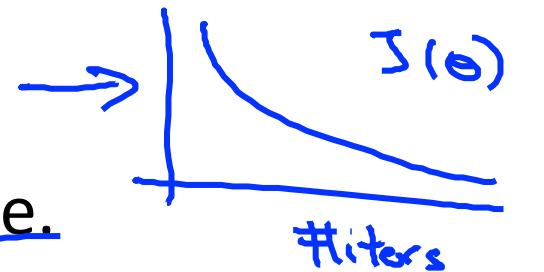
Making sure gradient descent is working correctly.



- 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:

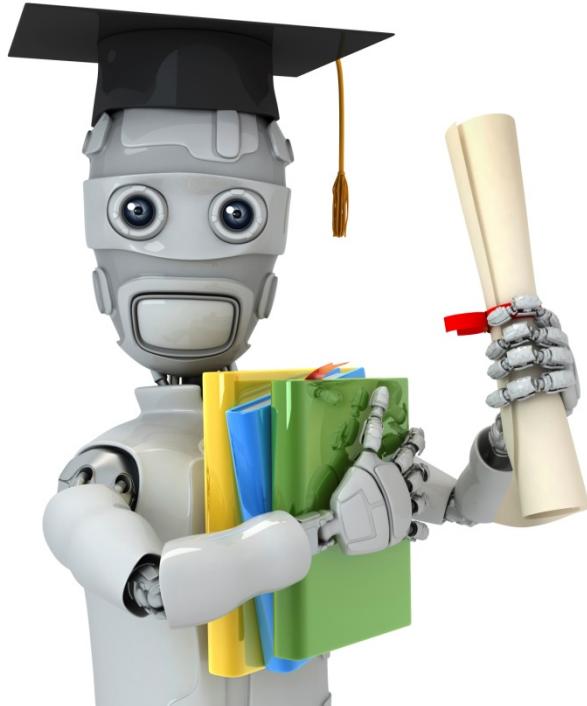
- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge. (Slow converge also possible)



To choose α , try

$$\dots, \underline{0.001}, \underline{0.003}, \underline{0.01}, \underline{0.03}, \underline{0.1}, \underline{0.3}, \underline{1}, \dots$$

$\nearrow 2x$ $\nwarrow 2x$ $\nearrow 3x$ $\nwarrow 3x$ \nearrow



Machine Learning

Linear Regression with multiple variables

Features and
polynomial regression

Housing prices prediction

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

x_1 x_2



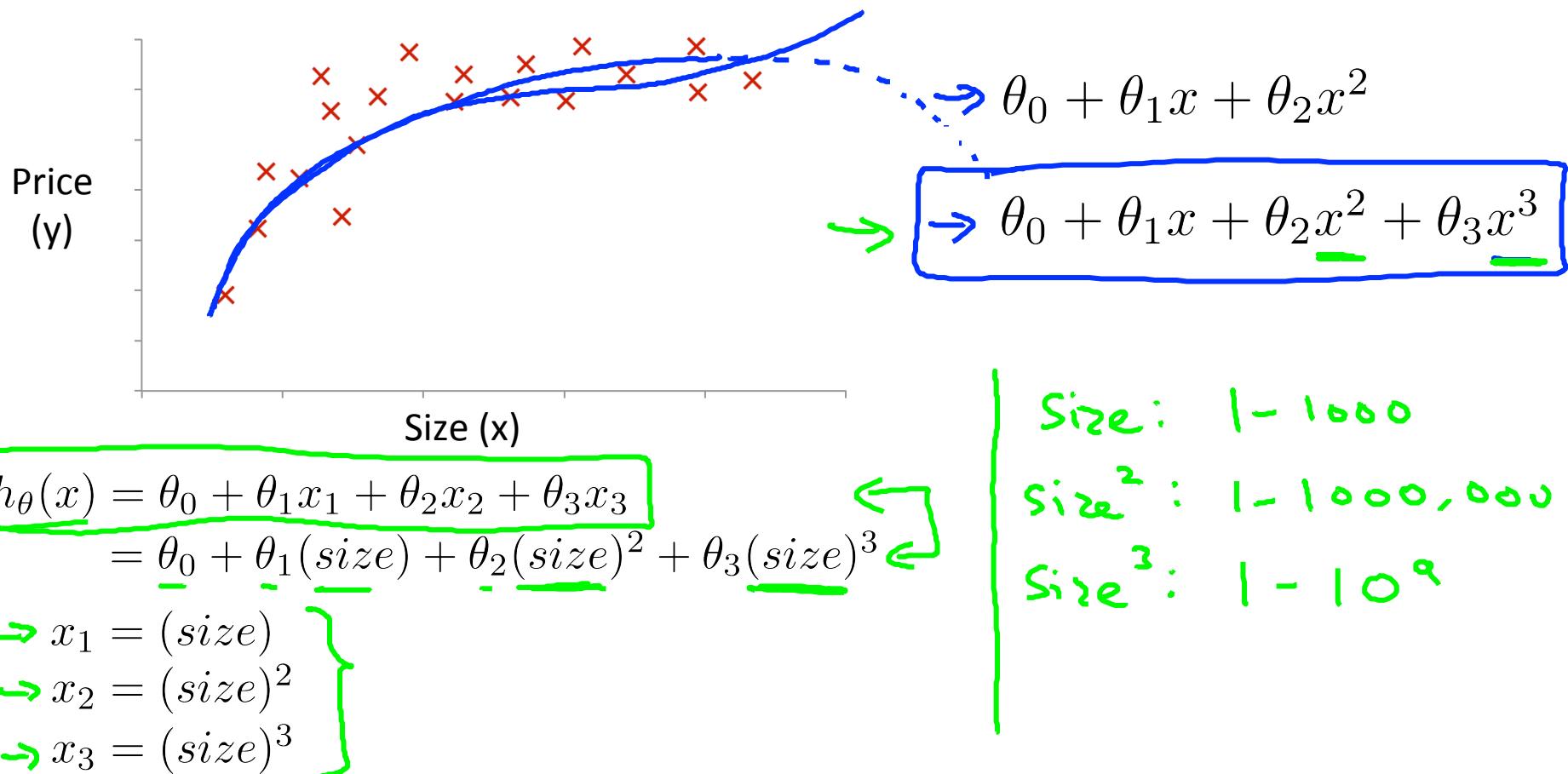
Area

$$x = \underline{\text{frontage} \times \text{depth}}$$

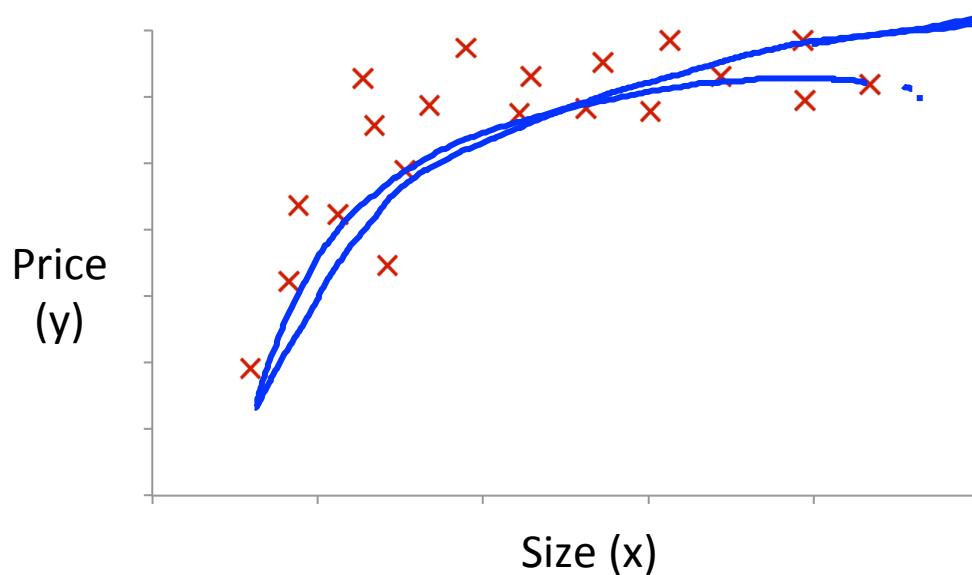
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

↑ land area

Polynomial regression

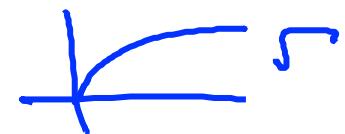


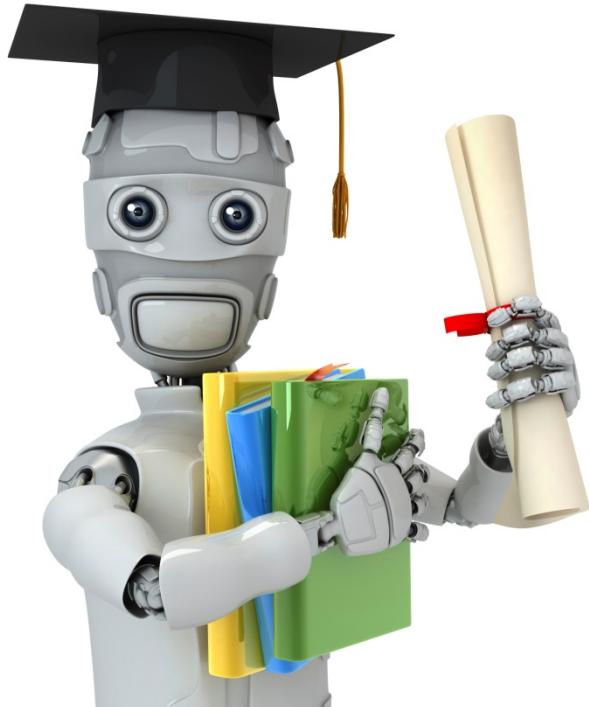
Choice of features



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

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



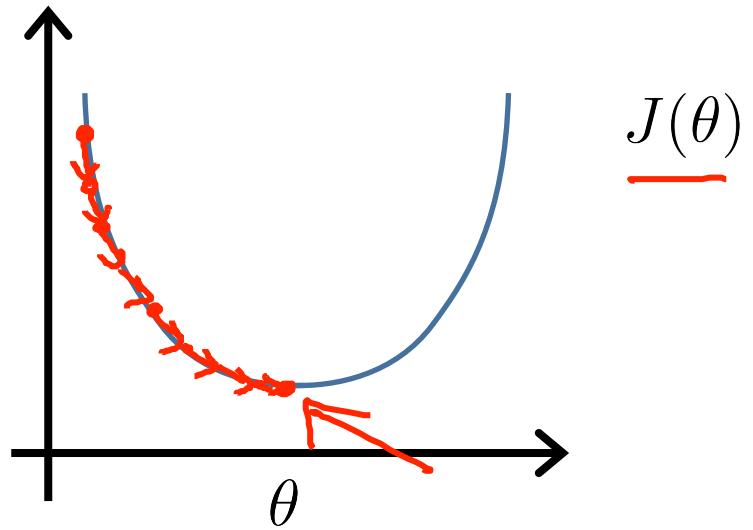


Machine Learning

Linear Regression with multiple variables

Normal equation

Gradient Descent



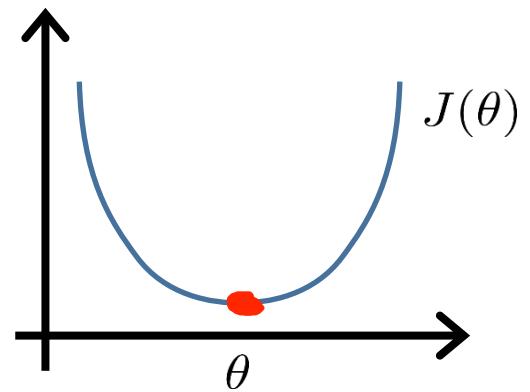
Normal equation: Method to solve for θ
analytically.

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

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

$$\frac{\partial}{\partial \theta} J(\theta) = \dots \stackrel{\text{set}}{=} 0$$

Solve for θ



$$\theta \in \mathbb{R}^{n+1}$$

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

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

Solve for $\underline{\theta_0, \theta_1, \dots, \theta_n}$

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

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

$m \times (n+1)$

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

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

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

\times (design matrix)

$$X = \begin{bmatrix} \cdots & (x^{(1)})^\top & \cdots \\ \cdots & (x^{(1)})^\top & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & (x^{(m)})^\top & \cdots \end{bmatrix}$$

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

$$\mathcal{O} = (X^\top X)^{-1} X^\top y$$

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ \vdots & \vdots \\ 1 & x_1^{(m)} \end{bmatrix}_{m \times 2}$$

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}_{m \times 1}$$

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

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

Set $A = X^T X$

$$\boxed{(X^T X)^{-1}} = A^{-1}$$

Octave: $\text{pinv}(X' * X) * X' * y$

$$\boxed{\text{pinv}(X^T * X) * X^T * y}$$

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

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

$$\begin{array}{l} X' \quad X^T \\ \cancel{\text{Feature Scaling}} \\ 0 \leq x_1 \leq 1 \\ 0 \leq x_2 \leq 1000 \\ 0 \leq x_3 \leq 10^{-5} \end{array} \checkmark$$

m training examples, n features.

Gradient Descent

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

$$\overbrace{n = 10^6}^{}$$

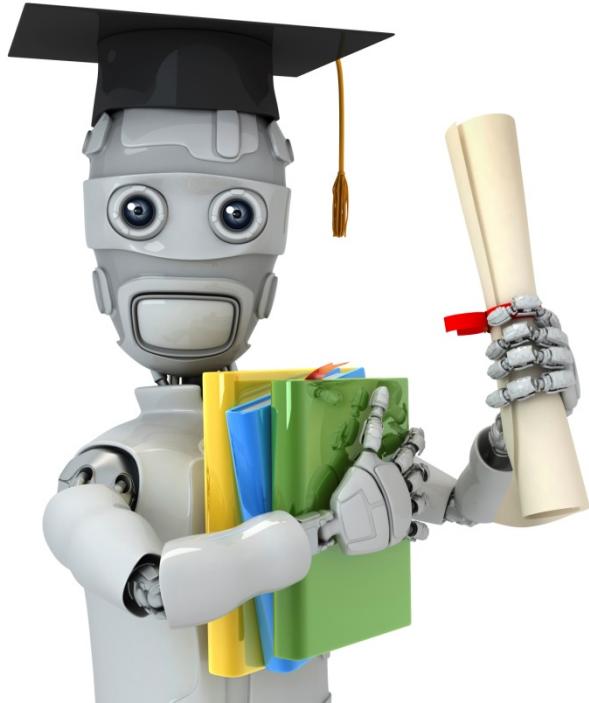
Normal Equation

- • No need to choose α .
- • Don't need to iterate.
- Need to compute $(X^T X)^{-1}$ $n \times n$ $O(n^3)$
- Slow if n is very large.

$$n = 100$$

$$n = 1000$$

$$\dots - n = 10000$$



Machine Learning

Linear Regression with multiple variables

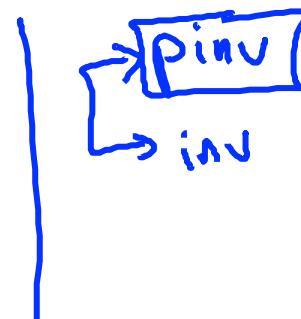
Normal equation
and non-invertibility
(optional)

Normal equation

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

$$\underline{X^T X}$$

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



What if $\boxed{X^T X}$ is non-invertible?



- Redundant features (linearly dependent).

E.g.
$$\begin{bmatrix} \underline{x_1} = \text{size in feet}^2 \\ \underline{x_2} = \text{size in m}^2 \\ \underline{x_1} = (3.28)^2 \underline{x_2} \end{bmatrix}$$

$$l_m = 3.28 \text{ feet}$$

$$\rightarrow \underline{m = 10} \leftarrow$$

$$\rightarrow \underline{n = 100} \leftarrow$$

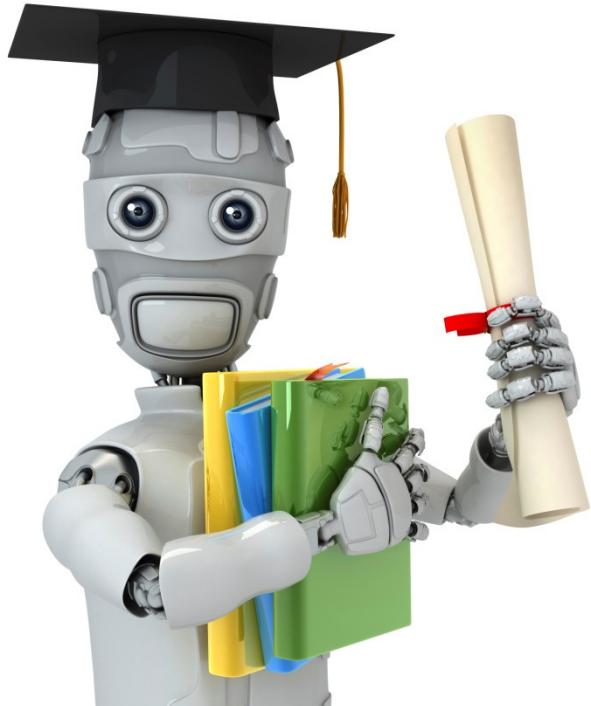
$$\mathbf{O} \in \mathbb{R}^{101}$$

- Too many features (e.g. $m \leq n$).

- Delete some features, or use regularization.

↓
later

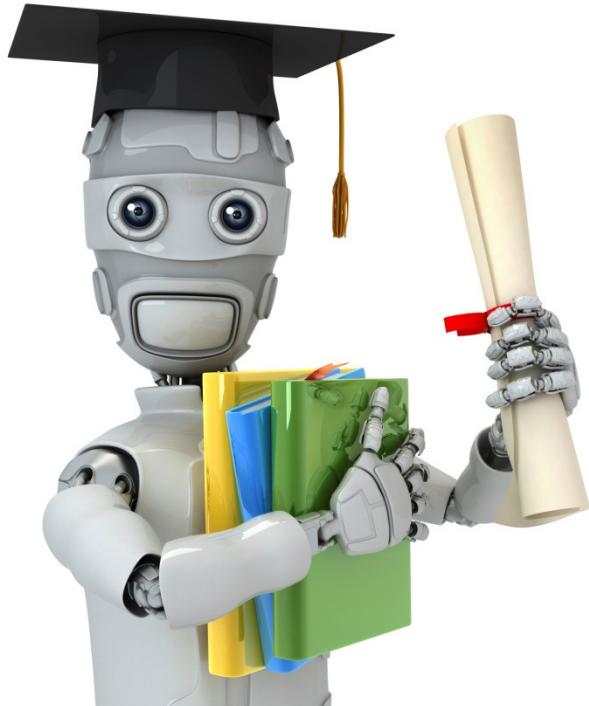
2.2 Octave



Machine Learning

Octave Tutorial

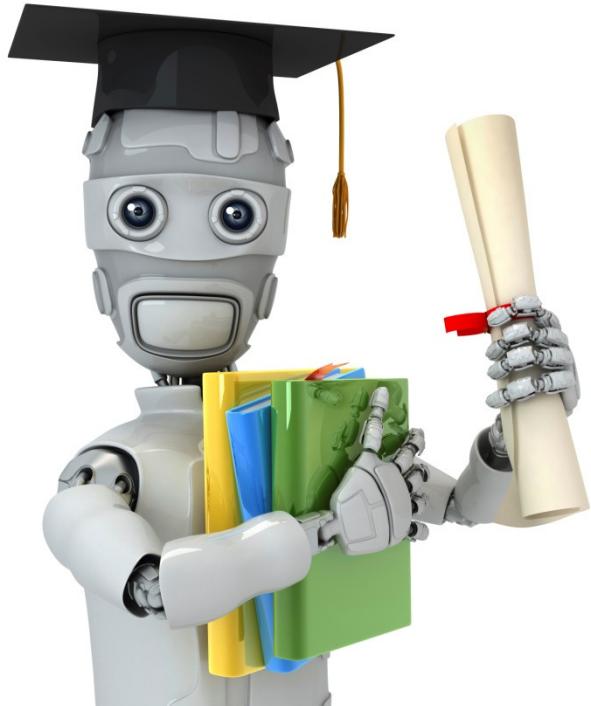
Basic operations



Machine Learning

Octave Tutorial

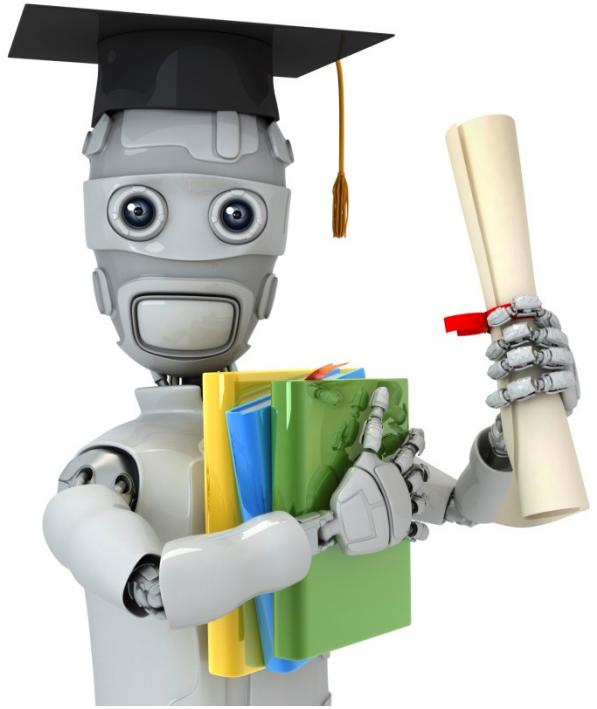
Moving data around



Machine Learning

Octave Tutorial

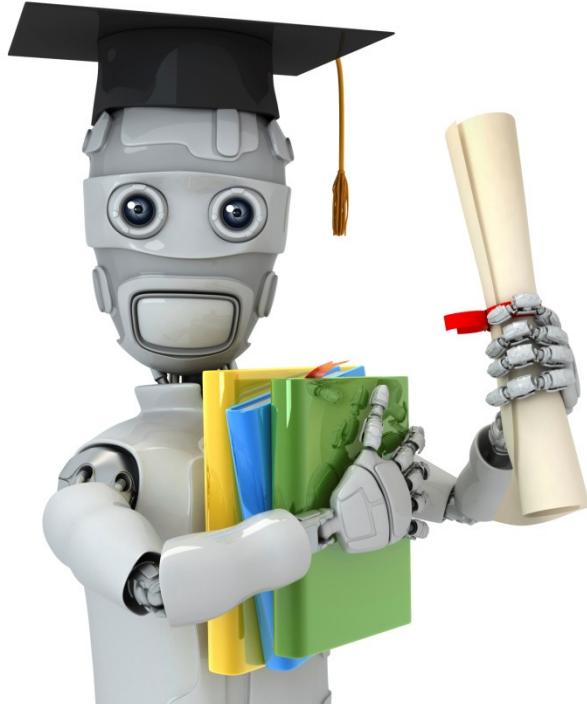
Computing on data



Machine Learning

Octave Tutorial

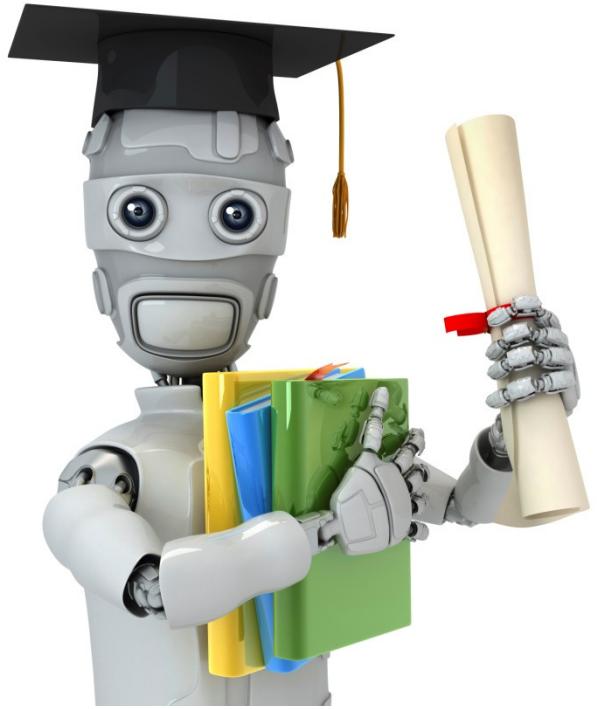
Plotting data



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

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

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

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

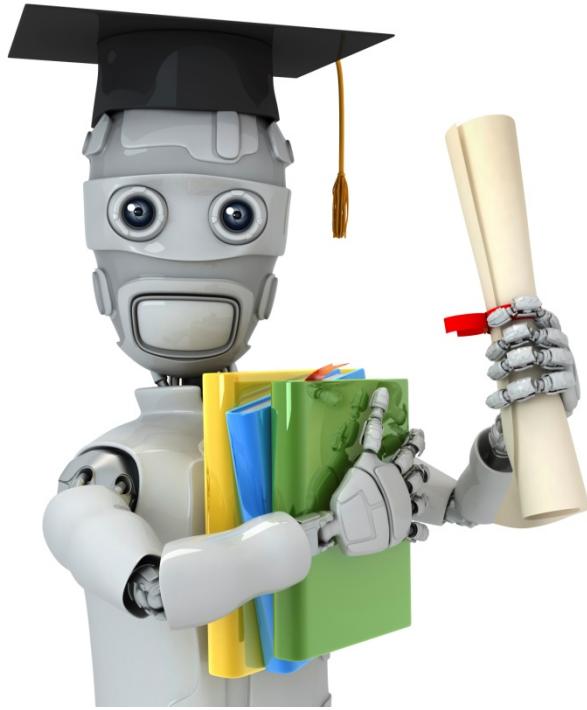
$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

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

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

Chapter 3 Week3

3.1 Logistic Reggression



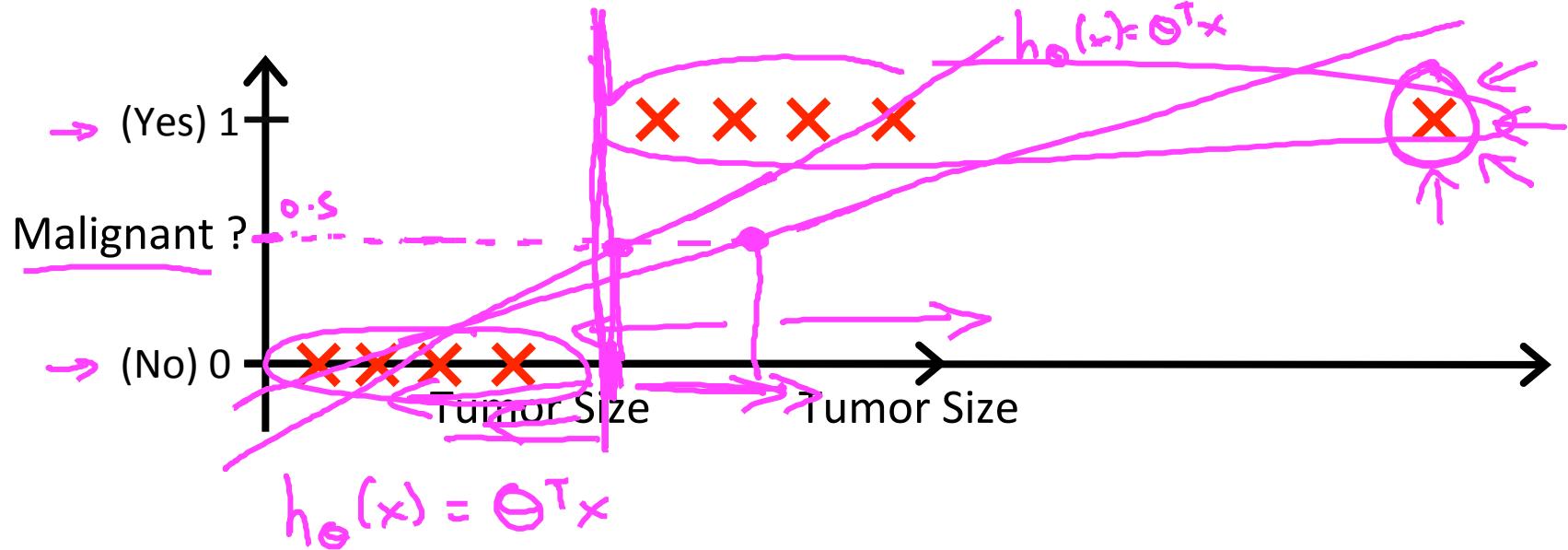
Machine Learning

Logistic Regression

Classification

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



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

→ If $h_\theta(x) \geq 0.5$, predict "y = 1"

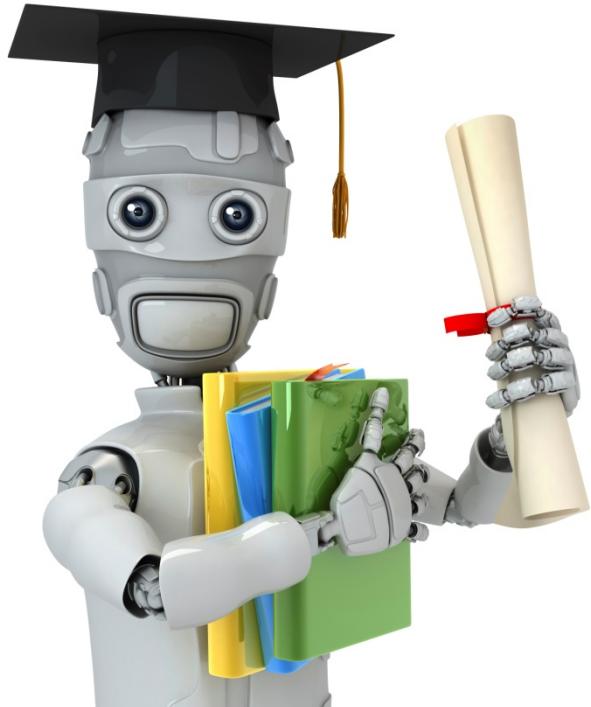
If $h_\theta(x) < 0.5$, predict "y = 0"

Classification: $y = 0 \text{ or } 1$

$h_\theta(x)$ can be $\underline{\quad}$ or $\underline{\quad}$

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

Classification



Machine Learning

Logistic Regression

Hypothesis Representation

Logistic Regression Model

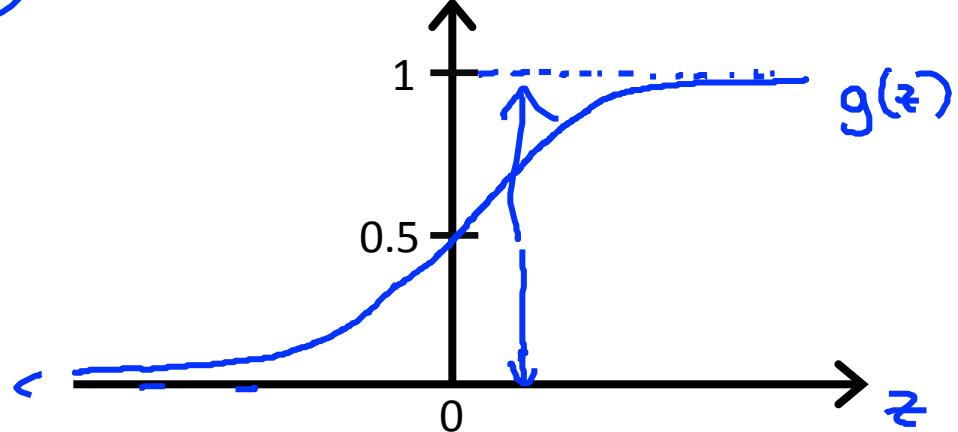
Want $0 \leq h_\theta(x) \leq 1$

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

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

$\theta^T x$

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



- Sigmoid function
- Logistic function

Parameters $\underline{\theta}$.

Interpretation of Hypothesis Output

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

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

Example: If $\underline{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 & \leftarrow \\ \text{tumorSize} & \leftarrow \end{bmatrix}$

$$\underline{h_{\theta}(x) = 0.7} \quad y=1$$

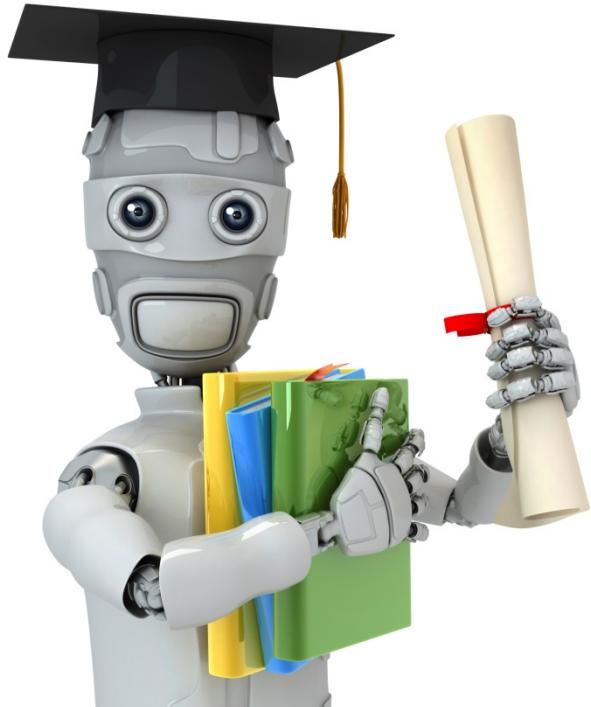
Tell patient that 70% chance of tumor being malignant

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

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

$$\underline{y = 0 \text{ or } 1}$$

$$\begin{aligned} \rightarrow P(y = 0|x; \theta) + P(y = 1|x; \theta) &= 1 \\ \rightarrow P(y = 0|x; \theta) &= 1 - P(y = 1|x; \theta) \end{aligned}$$



Machine Learning

Logistic Regression

Decision boundary

Logistic regression

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

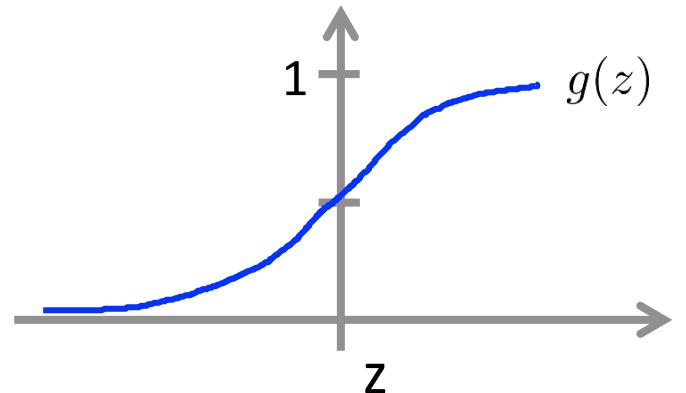
$$g(z) = \frac{1}{1+e^{-z}}$$

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

$$\theta^T x \geq 0$$

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

$$\theta^T x < 0$$



$$g(z) \geq 0.5$$

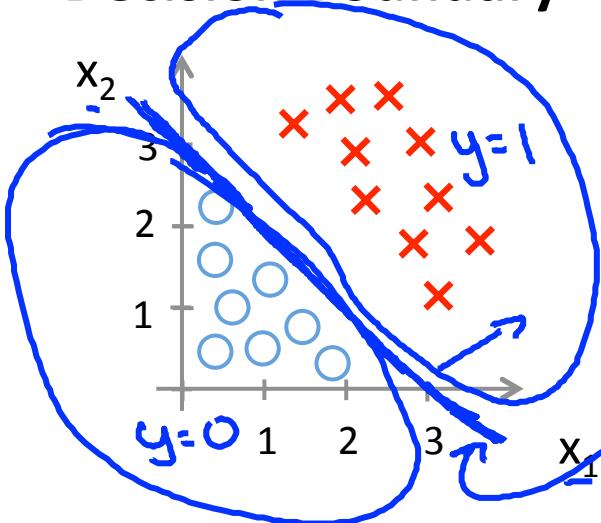
when $z \geq 0$

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

$$g(z) < 0.5$$

when $z < 0$

Decision Boundary



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

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

Decision boundary

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

$$\underline{\Theta^T x}$$

$$\underline{x_1 + x_2 \geq 3}$$

$$x_1, x_2$$

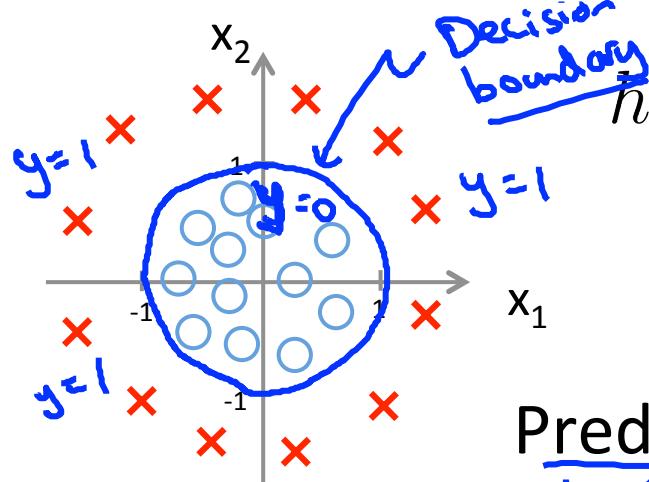
$$h_{\theta}(x) = 0.5$$

$$x_1 + x_2 = 3$$

$$x_1 + x_2 < 3$$

$$y = 0$$

Non-linear decision boundaries

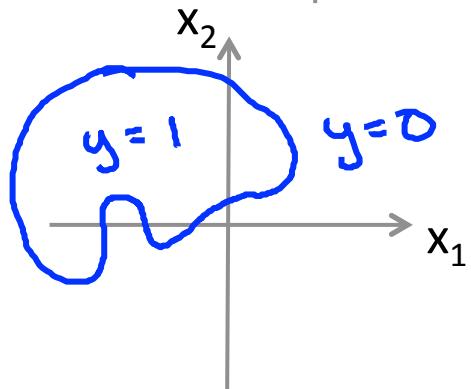


$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

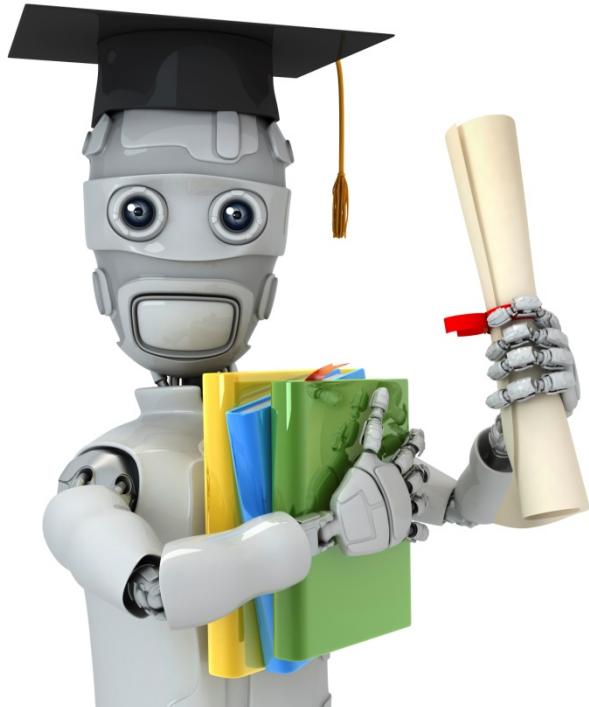
" " " "

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

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



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



Machine Learning

Logistic Regression

Cost function

Training
set:

m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad \mathbb{R}^{n+1}$$

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

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

How to choose parameters θ ?

Cost function

→ Linear regression: $J(\theta)$

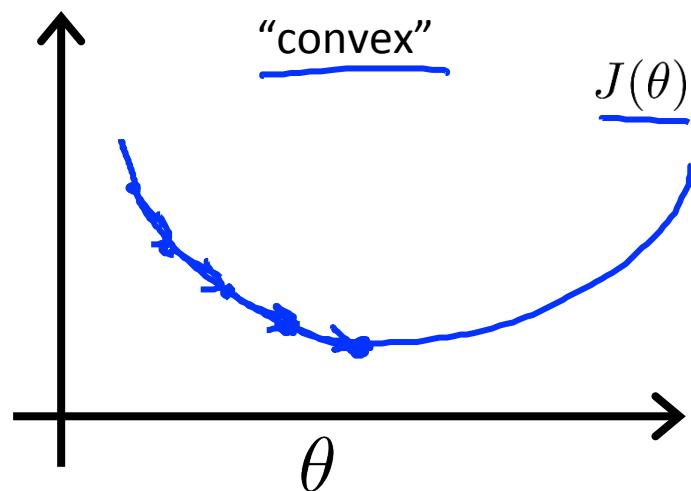
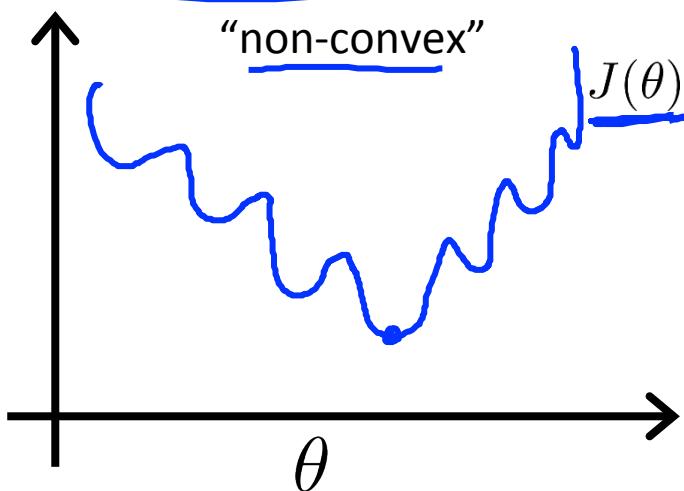
logistic

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

cost($h_\theta(x^{(i)})$, $y^{(i)}$)

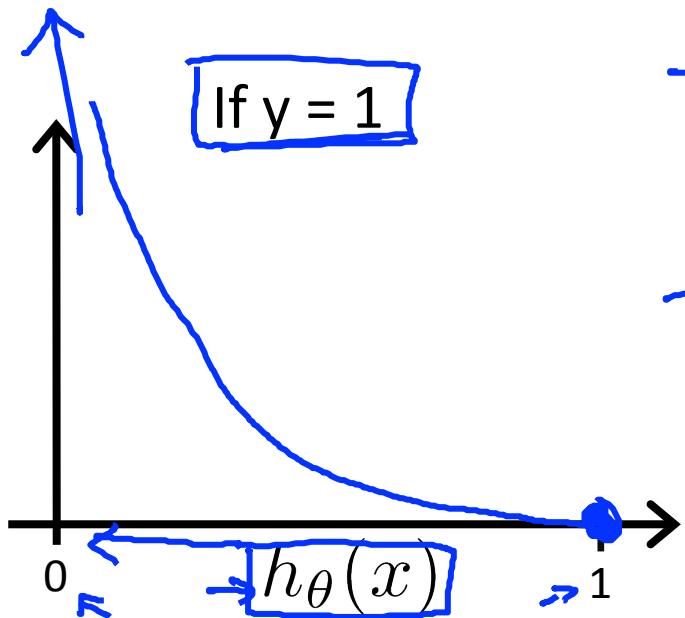
\rightarrow Cost($h_\theta(x^{(i)})$, $y^{(i)}$) = $\frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

$\frac{1}{2} e^{-\theta^T y}$



Logistic regression cost function

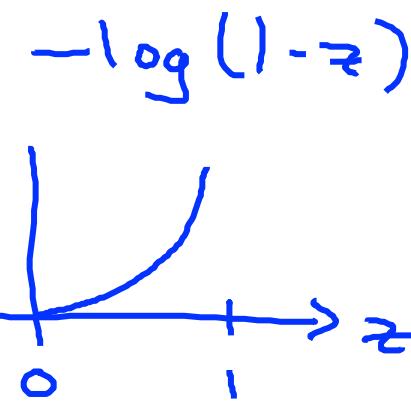
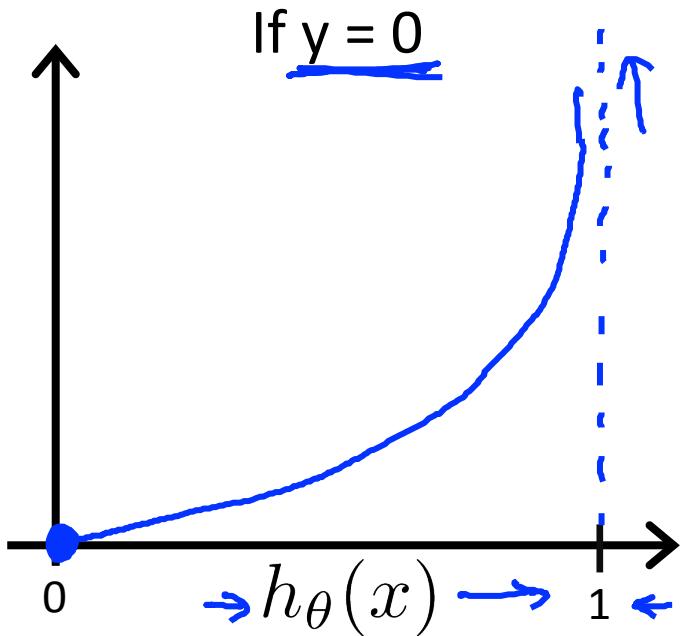
$$\text{Cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

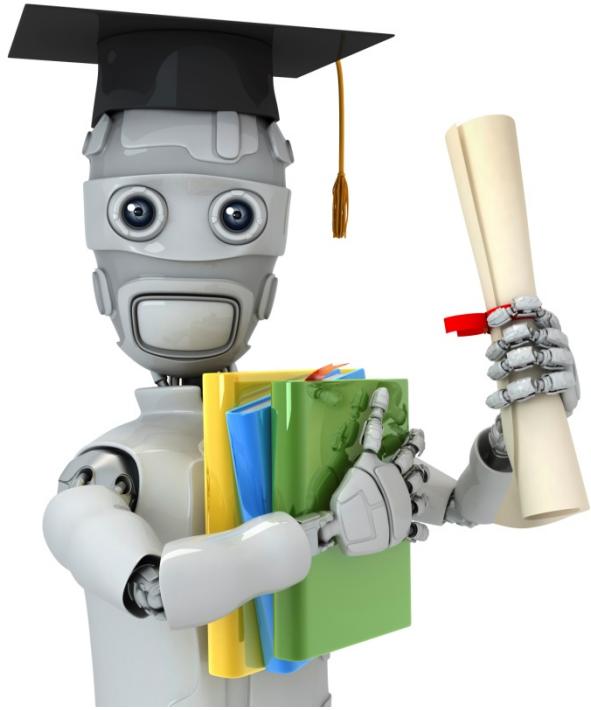


- Cost = 0 if $y = 1, h_{\theta}(x) = 1$
 - But as $h_{\theta}(x) \rightarrow 0$
 - $\underline{\text{Cost} \rightarrow \infty}$
- Captures intuition that if $h_{\theta}(x) = 0$,
(predict $P(y = 1|x; \theta) = 0$), but $y = 1$,
we'll penalize learning algorithm by a very
large cost.

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

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

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

Note: $y = 0$ or 1 always

$$\rightarrow \text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$$

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

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

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 :

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

Get $\underline{\theta}$

To make a prediction given new \underline{x} :

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

$$\underline{p(y=1 | x; \theta)}$$

Gradient Descent

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

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

Repeat {

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

}

(simultaneously update all θ_j)

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

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 {

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

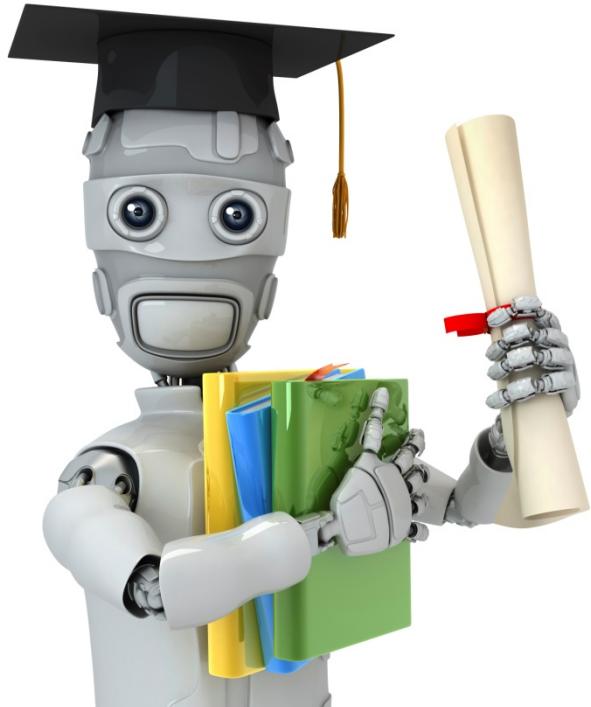
(simultaneously update all θ_j)

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \quad \text{for } i = 0 \rightarrow n$$

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

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

Algorithm looks identical to linear regression!



Machine Learning

Logistic Regression

Advanced optimization

Optimization algorithm

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

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 {

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

}

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:

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

$$\rightarrow \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \quad \theta_1 = 5, \theta_2 = 5.$$

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

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

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

\rightarrow options = optimset('GradObj', 'on', 'MaxIter', '100');

\rightarrow initialTheta = zeros(2,1);

$\left[\begin{array}{l} \text{optTheta}, \text{functionVal}, \text{exitFlag} \end{array} \right] \dots$

$= \text{fminunc}(@\text{costFunction}, \text{initialTheta}, \text{options});$

↑ ↑

$\theta \in \mathbb{R}^d \quad d \geq 2.$

```
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);
```

$$\underline{\text{theta}} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad \begin{array}{l} \xrightarrow{\text{theta}(1)} \\ \xrightarrow{\text{theta}(2)} \\ \vdots \\ \xrightarrow{\text{theta}(n+1)} \end{array}$$

function [jVal, gradient] = costFunction(theta)

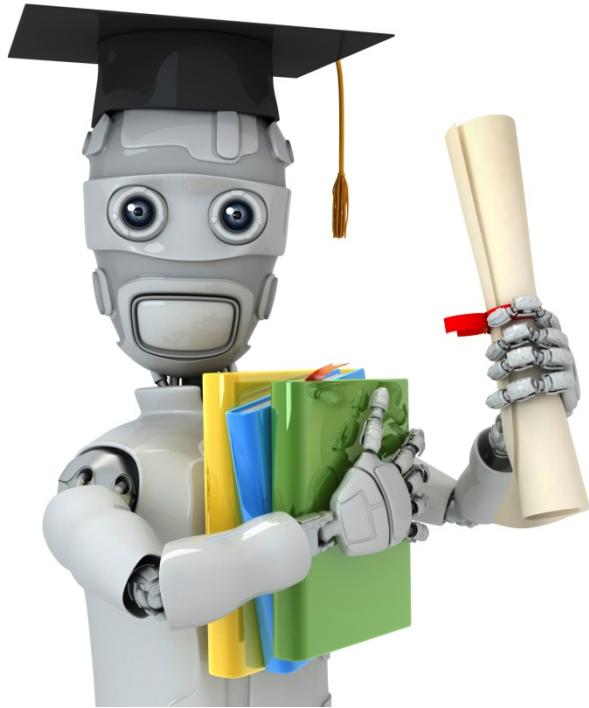
jVal = [code to compute $J(\theta)$];

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

$$y=1 \quad \uparrow \quad y=2 \quad \uparrow \quad y=3 \quad \uparrow \quad y=4$$

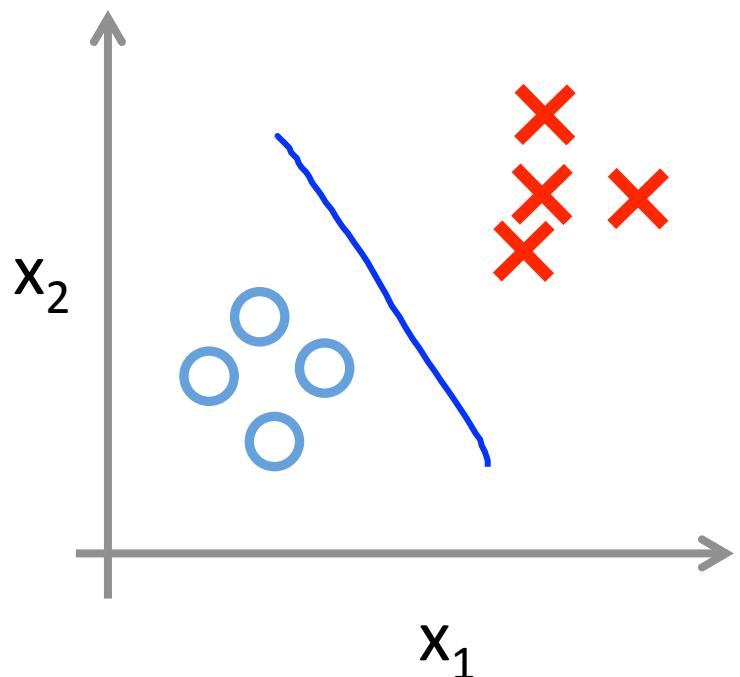
Medical diagrams: Not ill, Cold, Flu

$$y=1 \quad 2 \quad 3$$

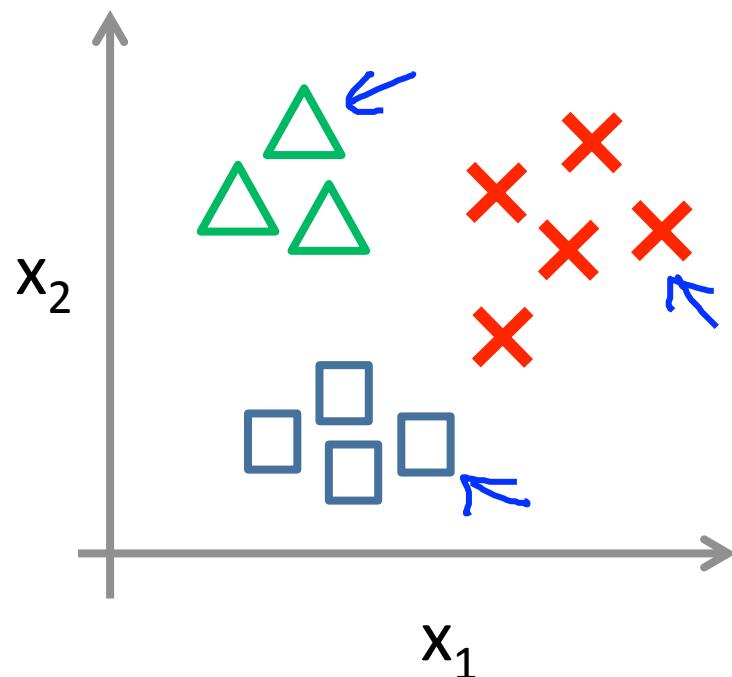
Weather: Sunny, Cloudy, Rain, Snow

$$y=1 \quad 2 \quad 3 \quad 4 \quad \leftarrow$$

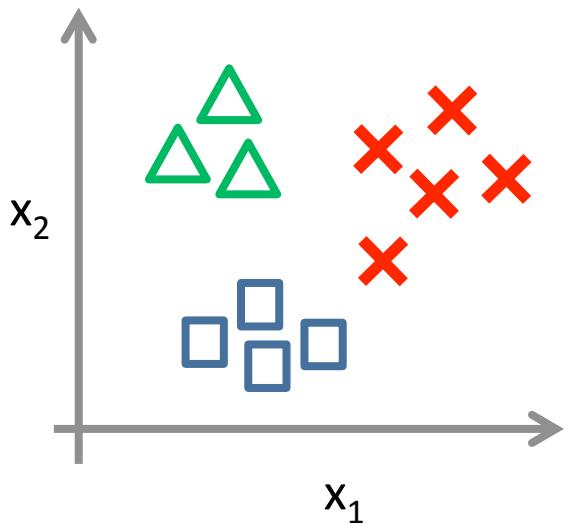

Binary classification:



Multi-class classification:

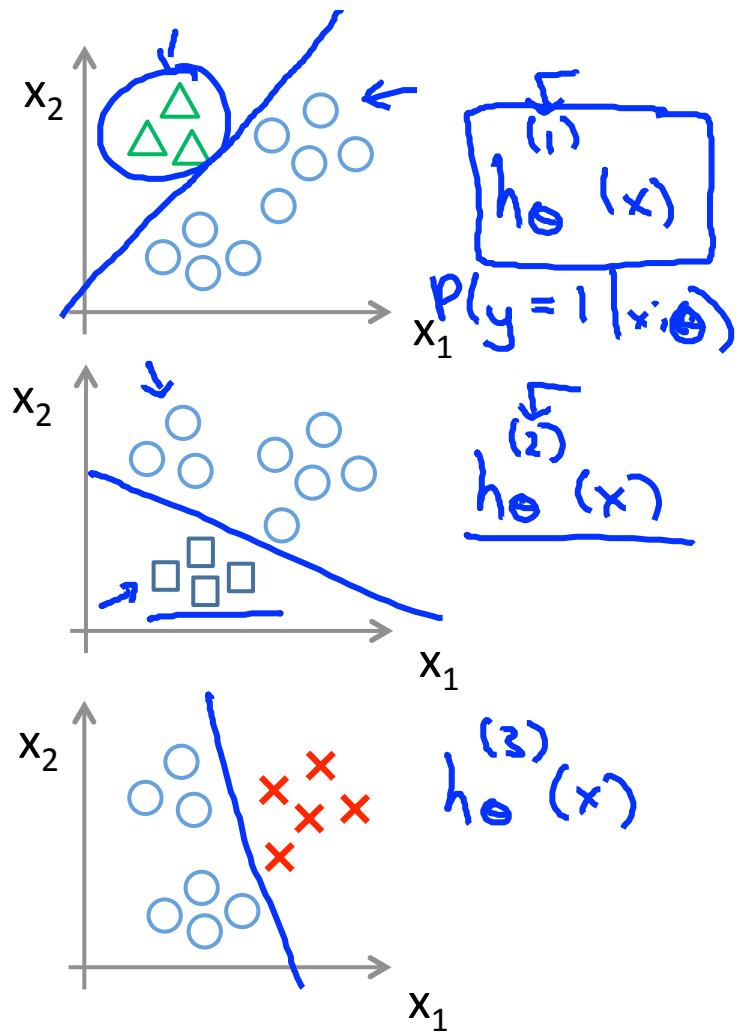


One-vs-all (one-vs-rest):



Class 1: ←
Class 2: ←
Class 3: ←

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



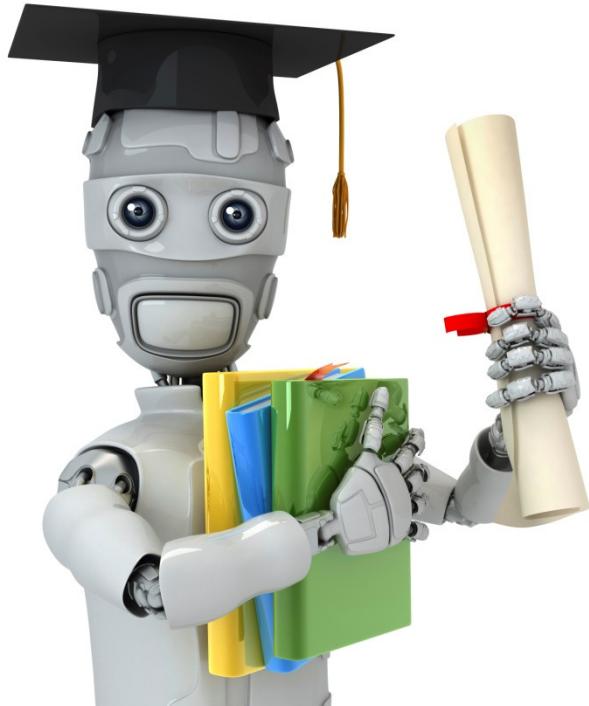
One-vs-all

Train a logistic regression classifier $\underline{h_{\theta}^{(i)}(x)}$ for each class \underline{i} to predict the probability that $\underline{y = i}$.

On a new input \underline{x} , to make a prediction, pick the class i that maximizes

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

3.2 Overfitting

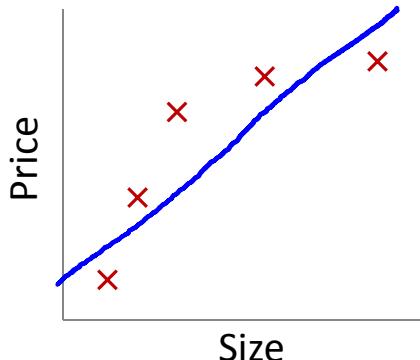


Machine Learning

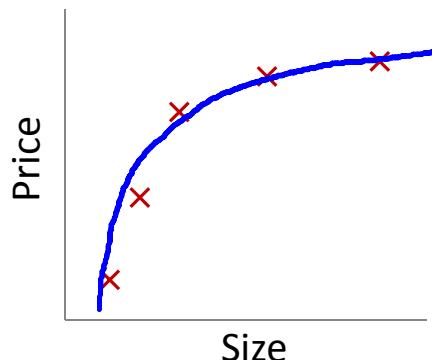
Regularization

The problem of overfitting

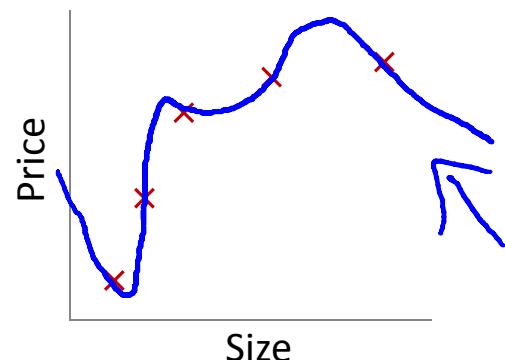
Example: Linear regression (housing prices)



$\rightarrow \theta_0 + \theta_1 x$
"Underfit" "High bias"



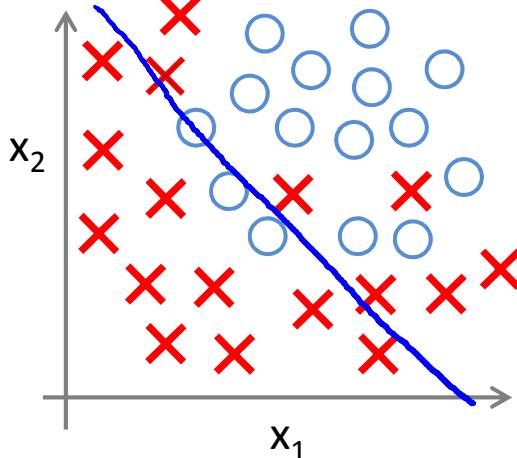
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$
"Just right"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
"Overfit" "High variance"

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).

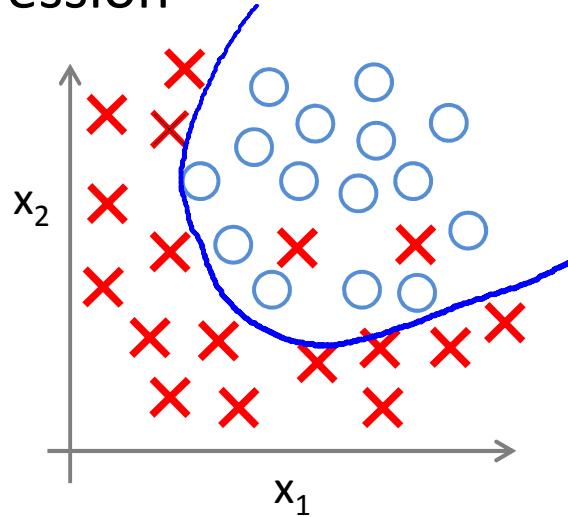
Example: Logistic regression



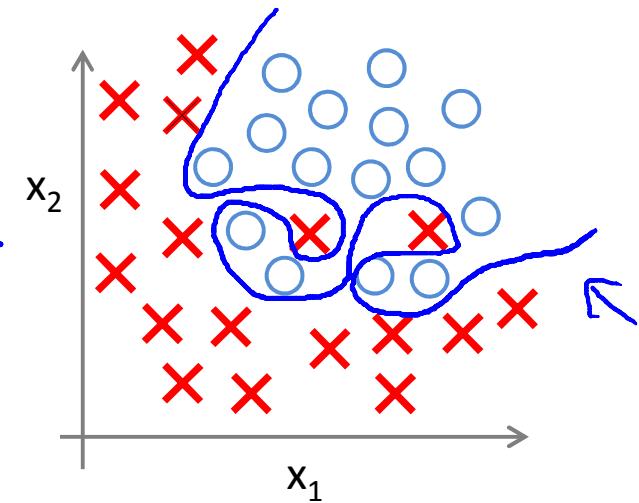
$$\rightarrow h_{\theta}(x) = g(\underline{\theta_0 + \theta_1 x_1 + \theta_2 x_2})$$

(g = sigmoid function)

\nwarrow
"Underfit"



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

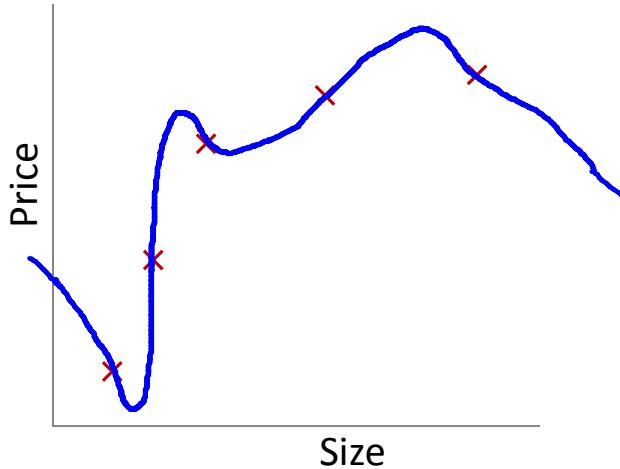


$$g(\underline{\theta_0 + \theta_1 x_1 + \theta_2 x_1^2} \\ + \underline{\theta_3 x_1^2 x_2} + \underline{\theta_4 x_1^2 x_2^2} \\ + \underline{\theta_5 x_1^2 x_2^3} + \underline{\theta_6 x_1^3 x_2} + \dots)$$

\nwarrow
"Overfit"

Addressing overfitting:

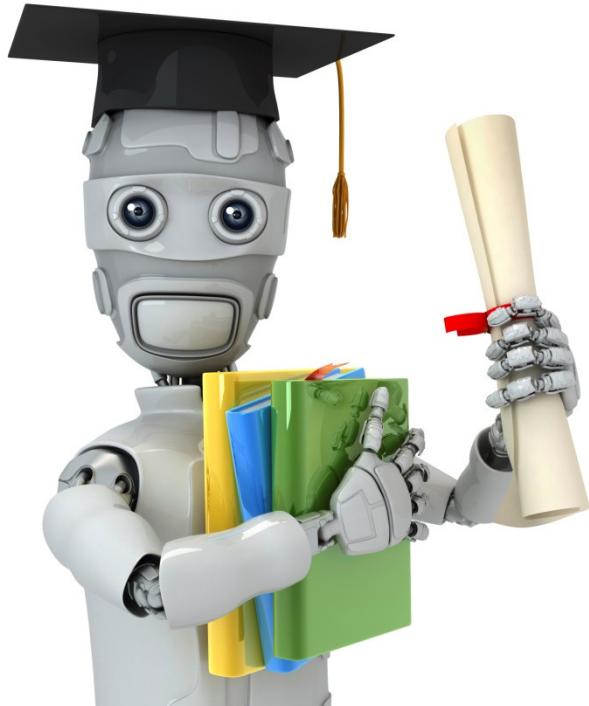
- x_1 = size of house
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of house
- x_5 = average income in neighborhood
- x_6 = kitchen size
- :
- x_{100}



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 .

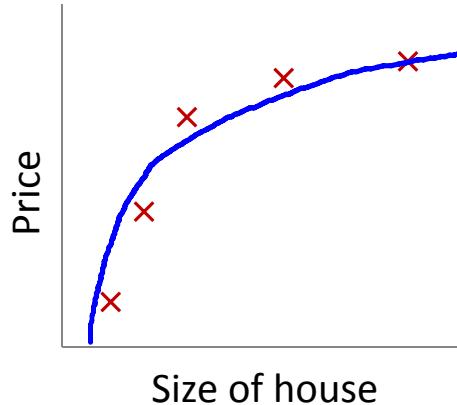


Machine Learning

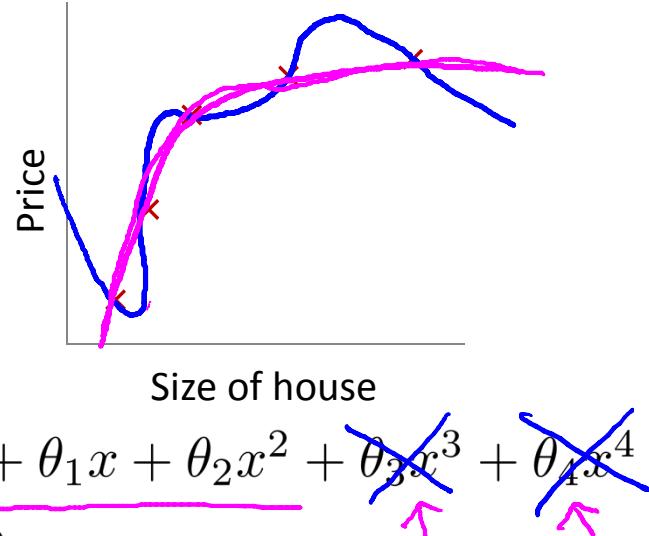
Regularization

Cost function

Intuition



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



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

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + 1000 \underline{\theta_3^2} + 1000 \underline{\theta_4^2}$$

$\underline{\theta_3 \approx 0}$ $\underline{\theta_4 \approx 0}$

Regularization.

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

- “Simpler” hypothesis
- Less prone to overfitting

$$\theta_3, \theta_4 \approx 0$$

Housing:

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

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

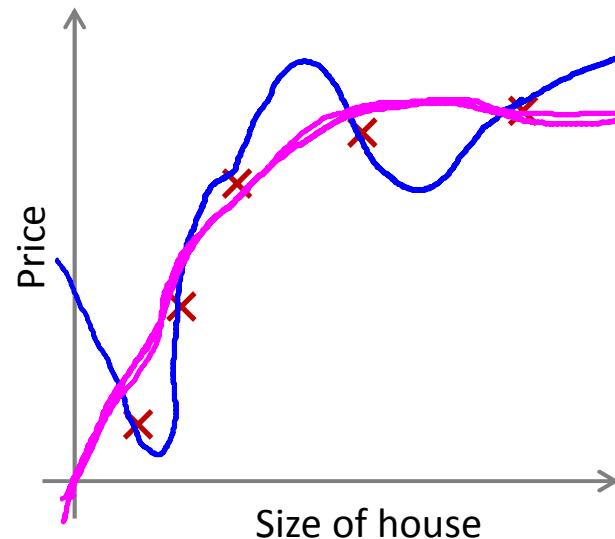
~~$\theta_1, \theta_2, \theta_3, \dots, \theta_{100}$~~

Regularization.

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

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

regularization parameter



In regularized linear regression, we choose θ to minimize

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

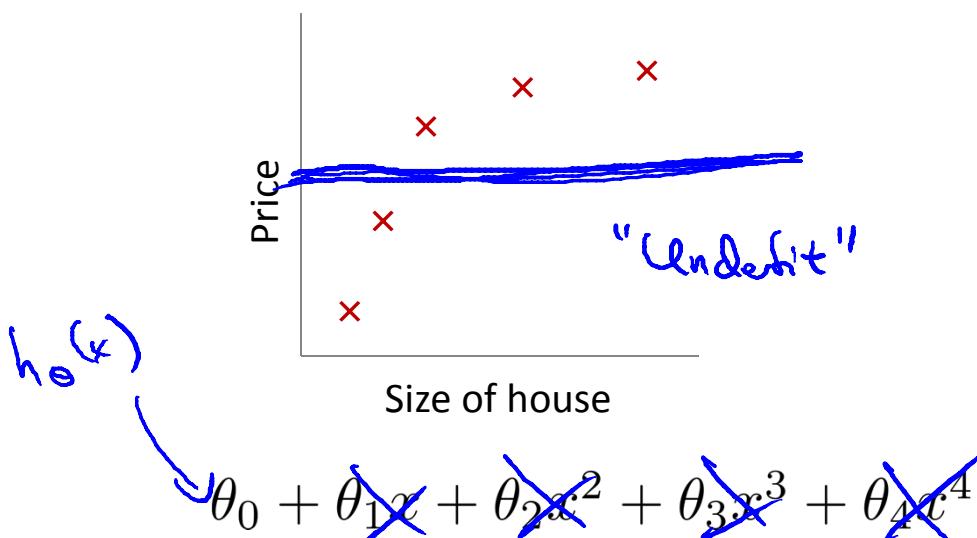
What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

- Algorithm works fine; setting λ to be very large can't hurt it
- Algorithm fails to eliminate overfitting.
- Algorithm results in underfitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.

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 $\lambda = 10^{10}$)?

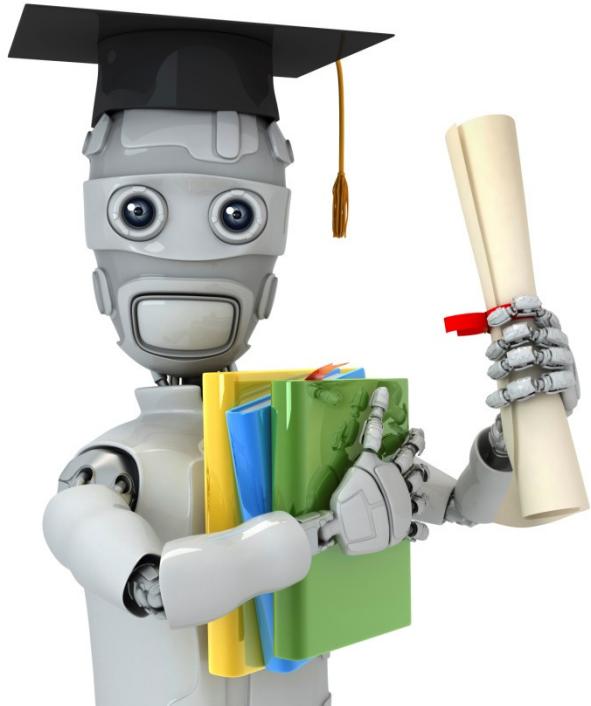


$$\underline{\theta}_1, \underline{\theta}_2, \underline{\theta}_3, \underline{\theta}_4$$

$$\theta_1 \approx 0, \theta_2 \approx 0$$

$$\theta_3 \approx 0, \theta_4 \approx 0$$

$$h_\theta(x) = \underline{\theta}_0$$



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 {

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

$$\frac{\partial}{\partial \theta_0}$$

$$\theta_1, \theta_2, \dots, \theta_n$$

$$\frac{\partial}{\partial \theta_0} J(\theta)$$

$$\begin{aligned} \theta_j &:= \theta_j - \boxed{\alpha} \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} - \frac{\lambda}{m} \theta_j \right] \\ &\quad (j = \cancel{0}, 1, 2, 3, \dots, n) \\ \theta_j &:= \boxed{\theta_j (1 - \alpha \frac{\lambda}{m})} - \boxed{\alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}} \rightarrow J(\theta) \end{aligned}$$

$$\boxed{\theta_j}$$

$$1 - \alpha \frac{\lambda}{m} < 1$$

$$\underline{0.99}$$

$$\theta_j \times 0.99$$

Normal equation

$$\underline{X} = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \leftarrow \text{m} \times (n+1)$$

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \uparrow \mathbb{R}^m$$

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

$$\frac{\partial}{\partial \theta_j} J(\theta) \stackrel{\text{set}}{=} 0 \quad \rightsquigarrow$$

$$\Rightarrow \theta = (X^T X + \lambda \underbrace{\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}}_{(n+1) \times (n+1)})^{-1} X^T y$$

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

Non-invertibility (optional/advanced).

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

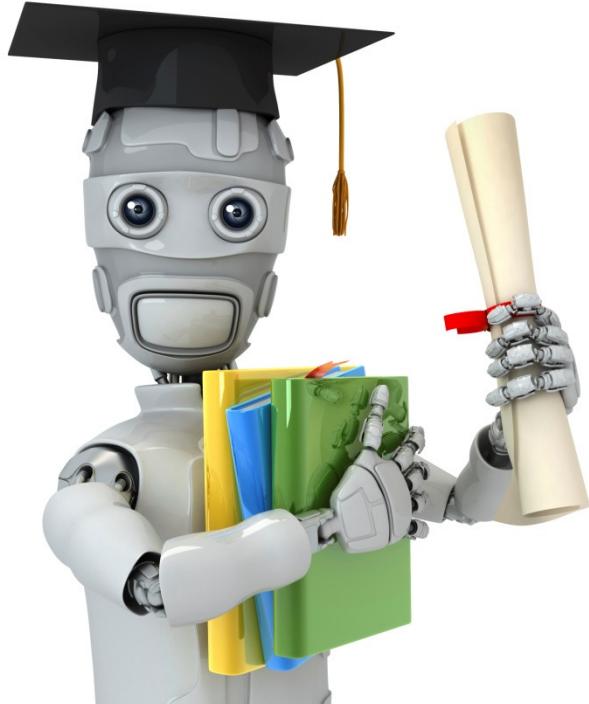
$$\theta = \underbrace{(X^T X)^{-1}}_{\text{non-invertible / singular}} X^T y$$

pinv $\frac{\text{inv}}{\kappa}$

If $\lambda > 0$,

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

invertible .

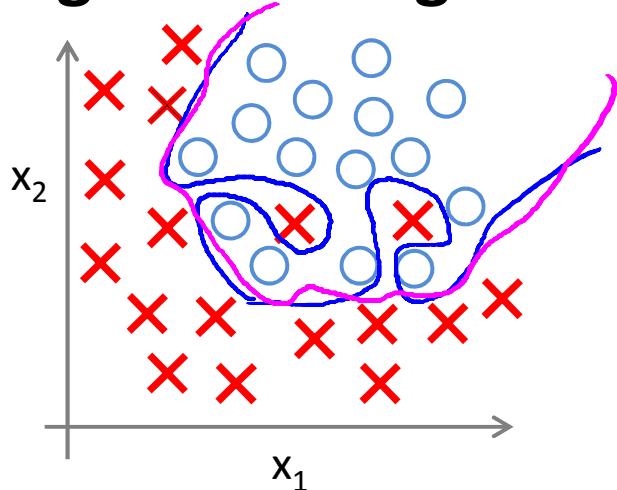


Machine Learning

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:

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

$\boxed{\theta_0, \theta_1, \dots, \theta_n}$

Gradient descent

Repeat {

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

$$\rightarrow \theta_j := \theta_j - \alpha \left[\underbrace{\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{(j = \cancel{x}, 1, 2, 3, \dots, n)} - \frac{1}{m} \theta_j \right] \leftarrow$$

$\theta_1, \dots, \theta_n$

}

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

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

Advanced optimization

f_{minimise} (a cost function) $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ $\theta_0 \leftarrow \text{theta}(1)$
 $\theta_1 \leftarrow \text{theta}(2)$
 $\theta_{n+1} \leftarrow \text{theta}(n+1)$

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

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

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

→ gradient (1) = [code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$];
 $\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$

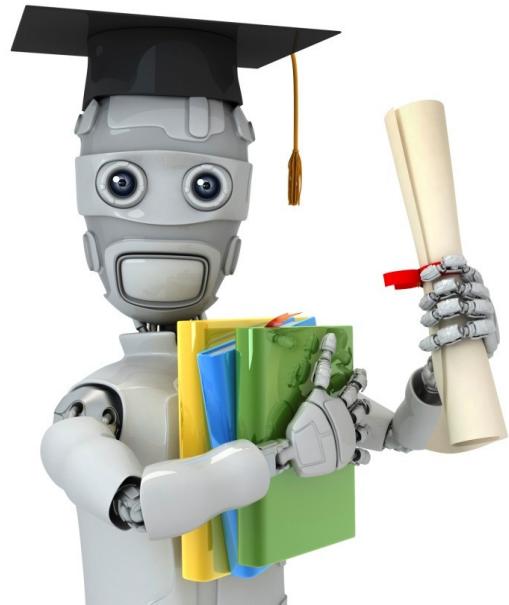
→ gradient (2) = [code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$];
 $\left(\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \right) - \frac{\lambda}{m} \theta_1$ $J(\theta)$

→ gradient (3) = [code to compute $\frac{\partial}{\partial \theta_2} J(\theta)$];
 $\vdots \quad \left(\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \right) - \frac{\lambda}{m} \theta_2$

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

Chapter 4 Neural Networks

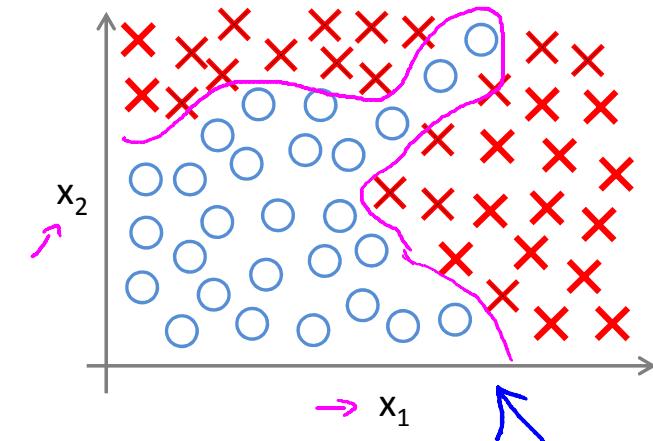
4.1 Non-lineay Hypotheses



Machine Learning

Neural Networks: Representation Non-linear hypotheses

Non-linear Classification



$\rightarrow \underline{x_1} = \text{size}$
 $\underline{x_2} = \# \text{bedrooms}$
 $\underline{x_3} = \# \text{floors}$
 $x_4 = \text{age}$
 \dots
 $x_{100} -$

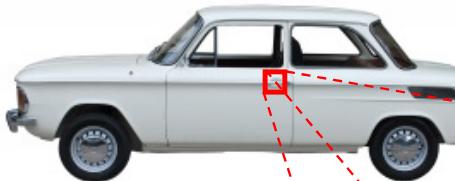
$\left\{ h=100 \right.$

$$\begin{aligned}
 & \text{Handwritten formula: } g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 \\
 & \quad + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 \\
 & \quad + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots) \\
 \\
 & \rightarrow \underline{x_1^2}, \underline{x_1 x_2}, \underline{x_1 x_3}, \underline{x_1 x_4} \dots \underline{x_1 x_{100}} \\
 & \quad \underline{x_2^2}, \underline{x_1 x_3} \dots \\
 & \quad \text{~5000 feature} \quad O(n^2) \\
 \\
 & \rightarrow \underline{x_1^2}, \underline{x_2^2}, \underline{x_3^2}, \dots, \underline{x_{100}^2} \\
 \\
 & \rightarrow \underline{x_1 x_2 x_3}, \underline{x_1^2 x_2}, \underline{x_{10} x_{11} x_{12}}, \dots \\
 & \quad O(n^3) \quad \frac{n^2}{2} \quad 170,000
 \end{aligned}$$

Andrew Ng

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



Andrew Ng

Computer Vision: Car detection



Cars



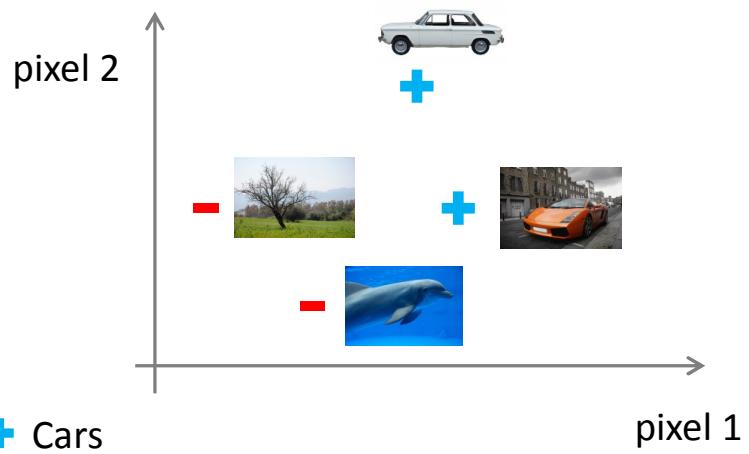
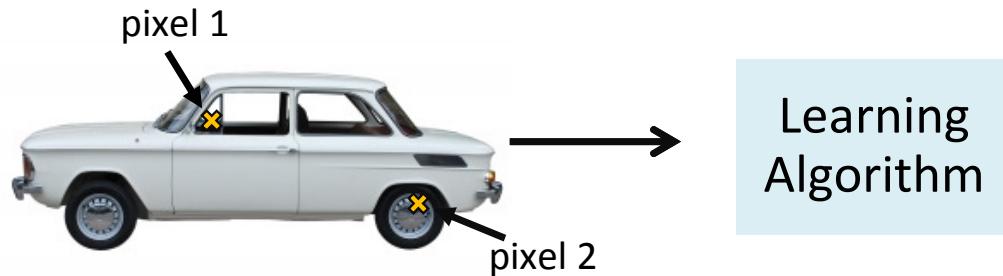
Not a car

Testing:

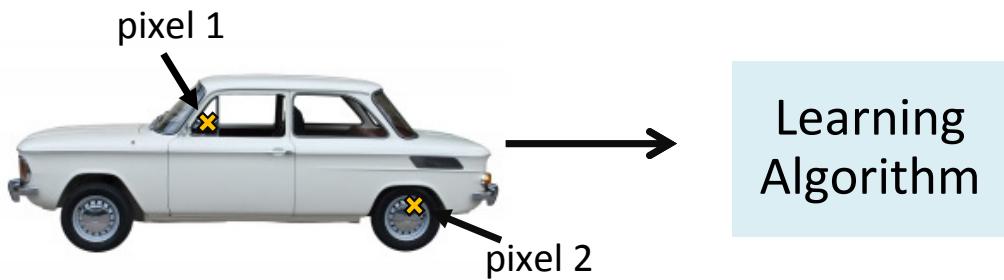


What is this?

Andrew Ng

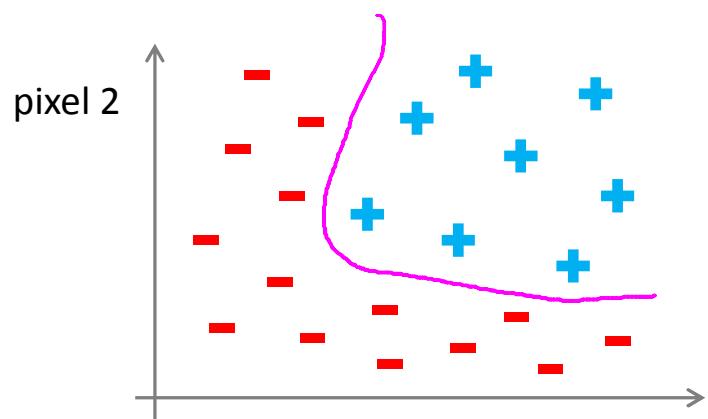
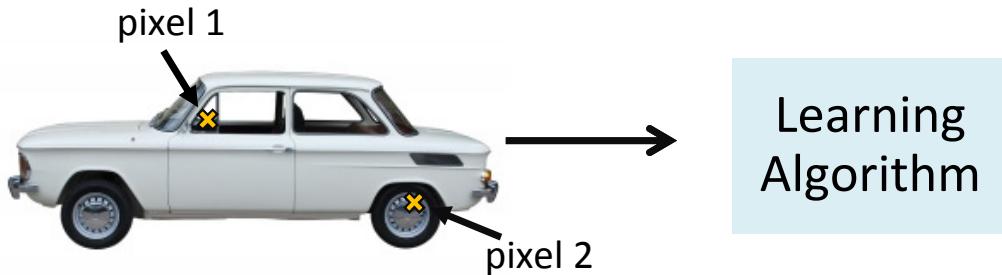


Andrew Ng



- ✚ Cars
- ⊖ “Non”-Cars

Andrew Ng



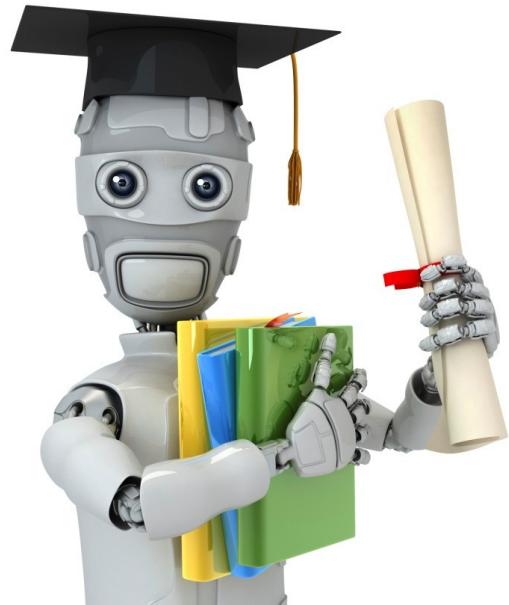
50 x 50 pixel images \rightarrow 2500 pixels

$n = 2500$ (7500 if RGB)

$$x = \begin{bmatrix} \text{pixel 1 intensity} & 0-255 \\ \text{pixel 2 intensity} & \\ \vdots & \\ \text{pixel 2500 intensity} & \end{bmatrix}$$

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

Andrew Ng



Machine Learning

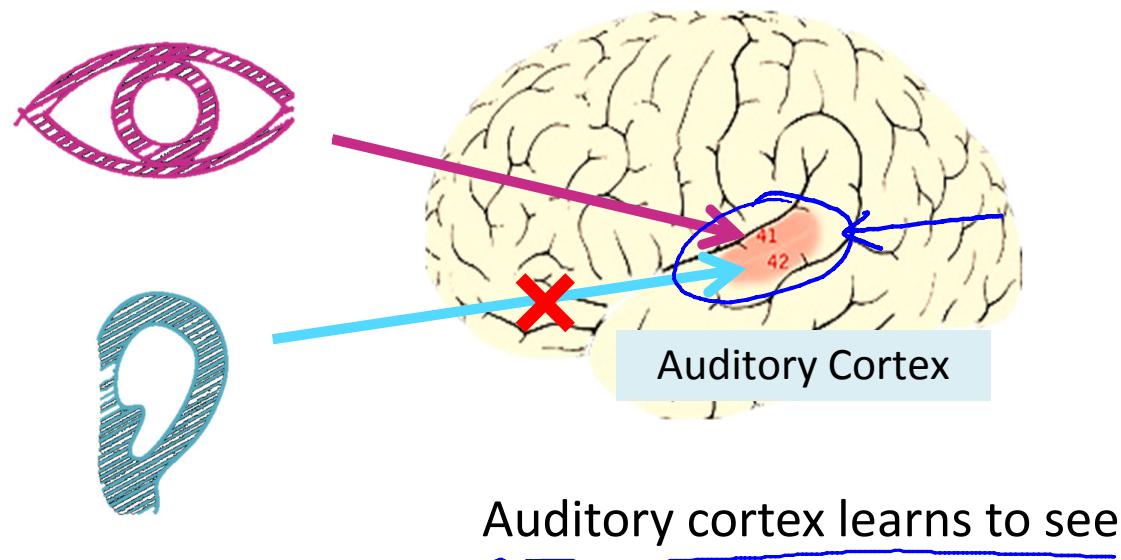
Neural Networks: Representation

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

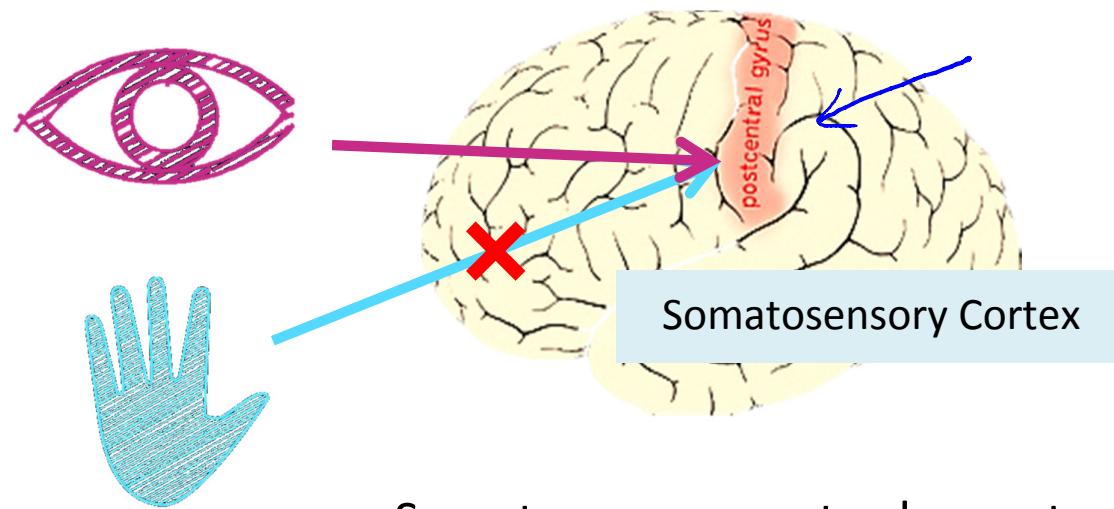
The “one learning algorithm” hypothesis



[Roe et al., 1992]

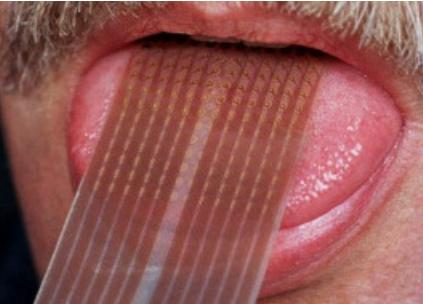
Andrew Ng

The “one learning algorithm” hypothesis



Somatosensory cortex learns to see

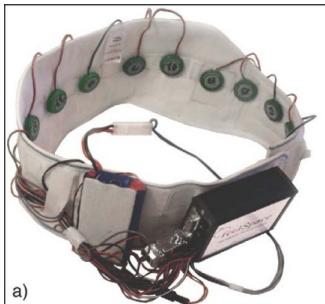
Sensor representations in the brain



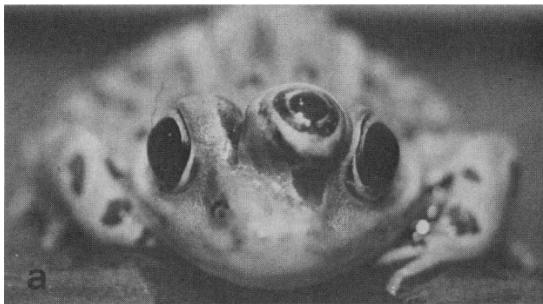
Seeing with your tongue



Human echolocation (sonar)



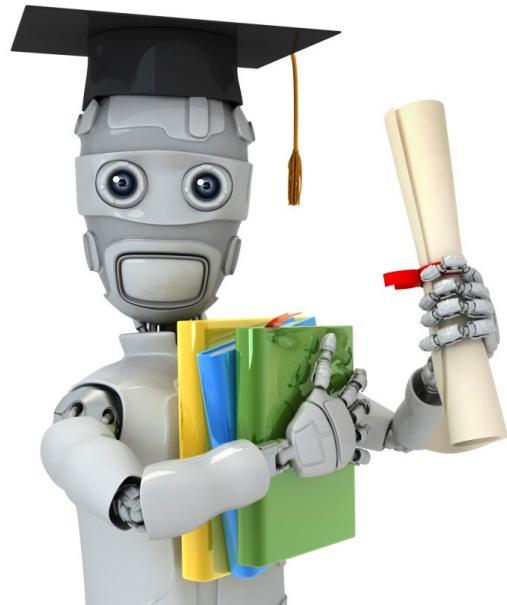
Haptic belt: Direction sense



Implanting a 3rd eye

[BrainPort; Welsh & Blasch, 1997; Nagel et al., 2005; Constantine-Paton & Law, 2009]

Andrew Ng

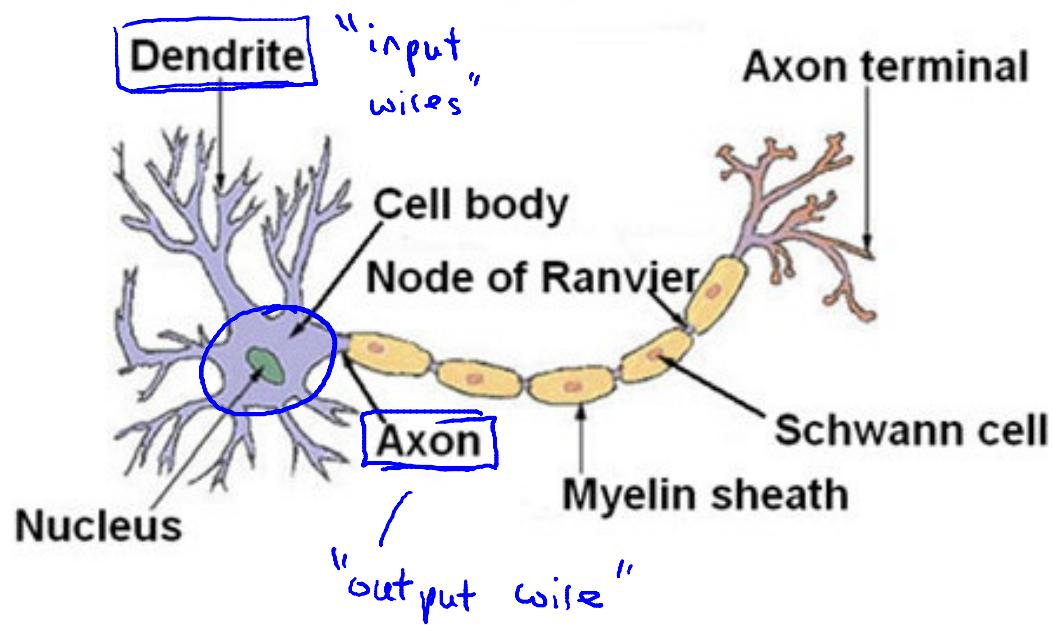


Machine Learning

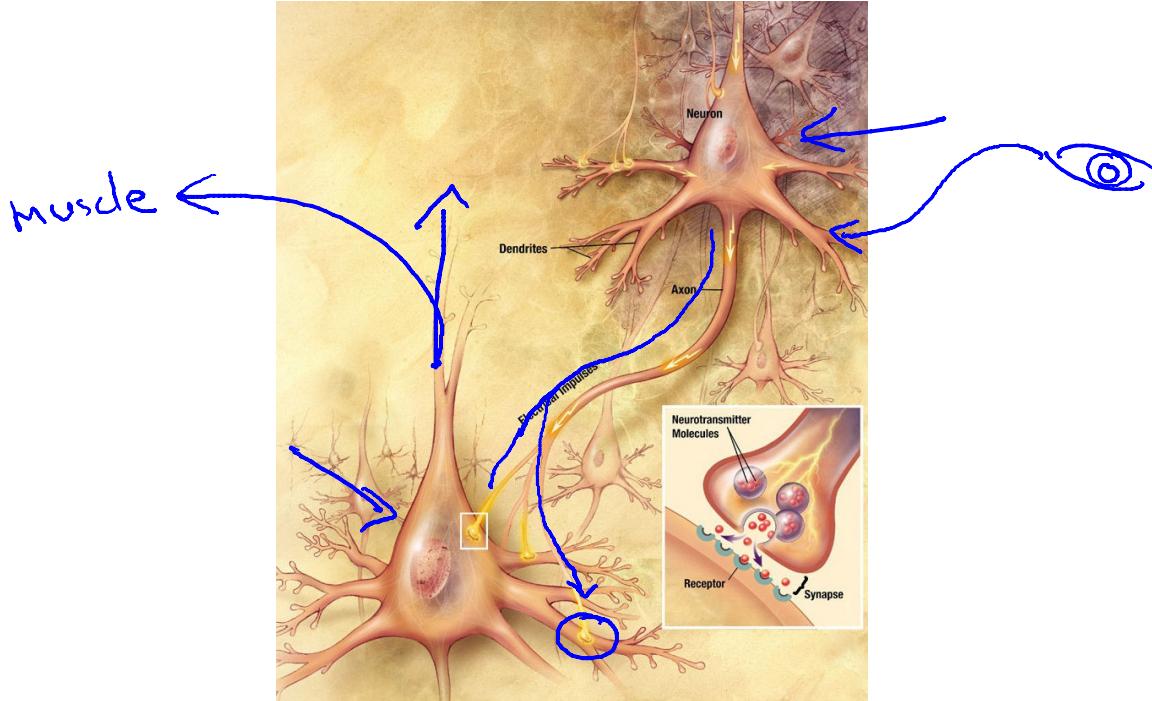
Neural Networks: Representation

Model representation I

Neuron in the brain



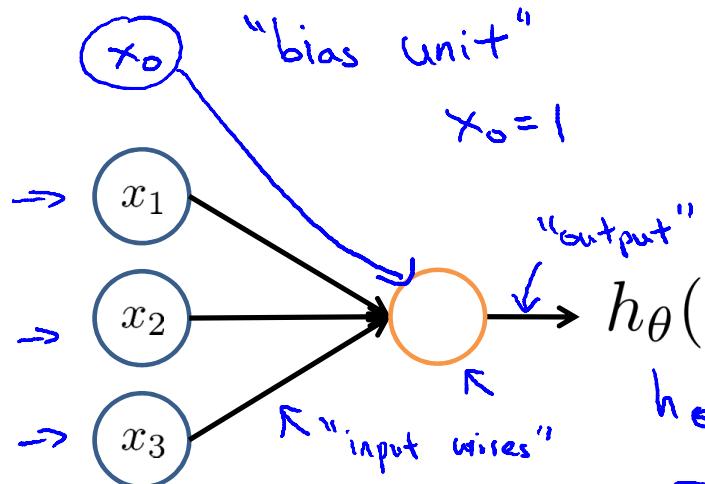
Neurons in the brain



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

Andrew Ng

Neuron model: Logistic unit



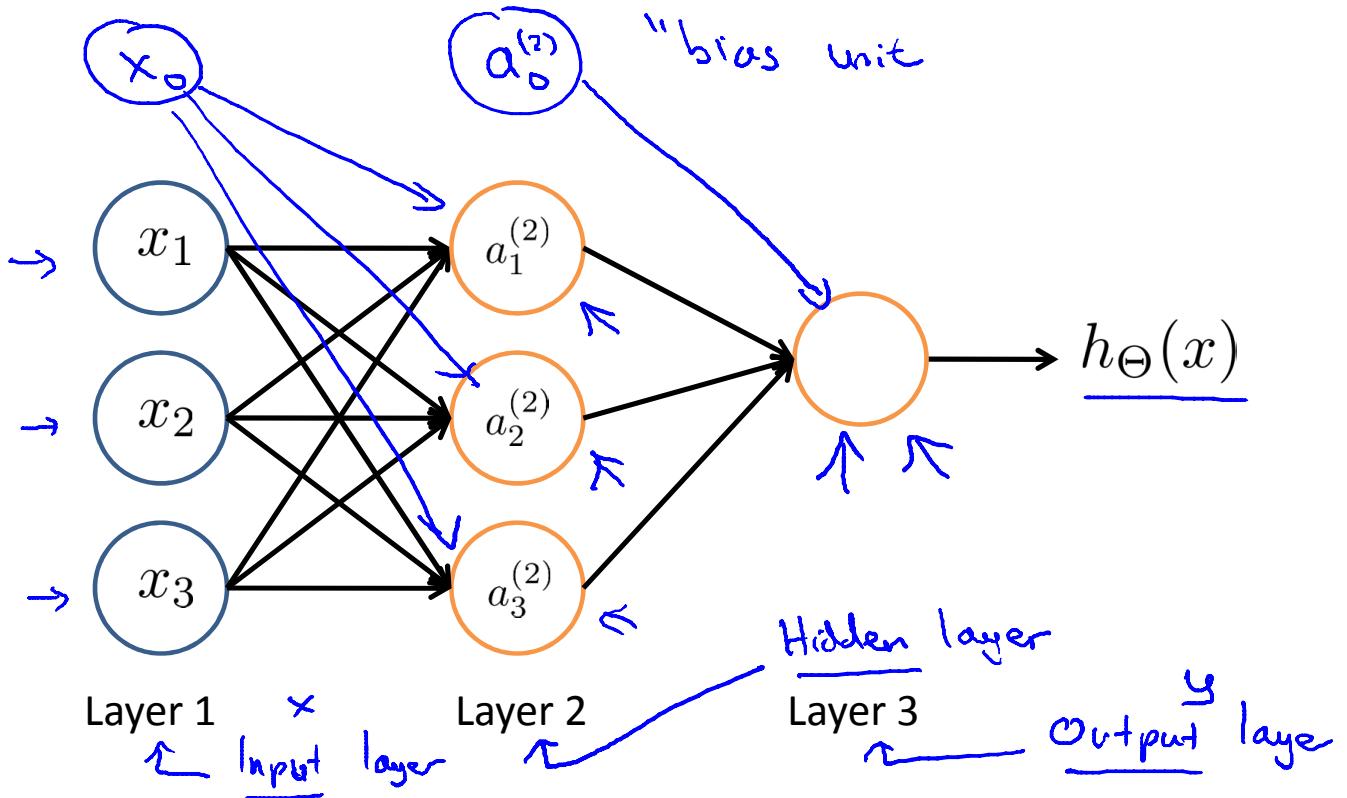
$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

\uparrow
"weights" \leftarrow
(parameters \leftarrow)

Sigmoid (logistic) activation function.

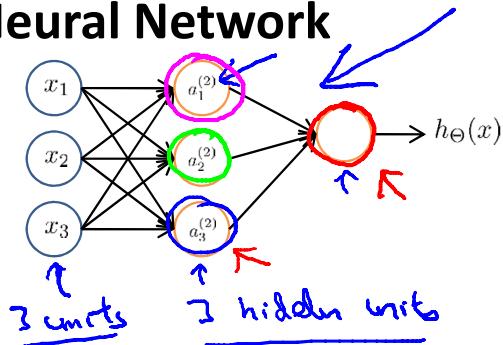
$$g(z) = \frac{1}{1 + e^{-z}}$$

Neural Network



Andrew Ng

Neural Network



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

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

$$\Theta^{(j)} \in \mathbb{R}^{3 \times 4}$$

layer $j + 1$

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

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

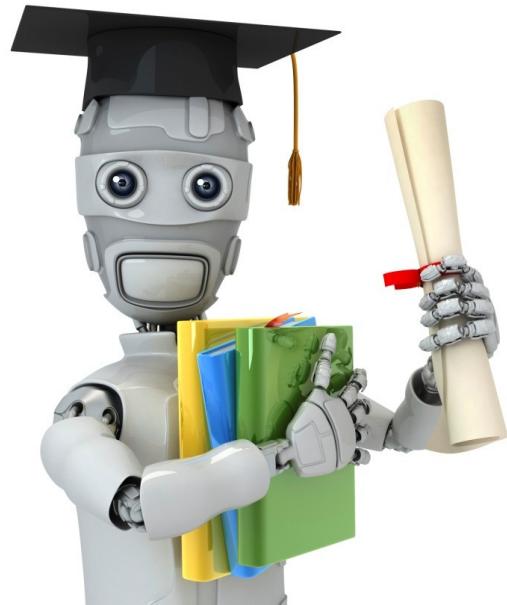
$$\rightarrow a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

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

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

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

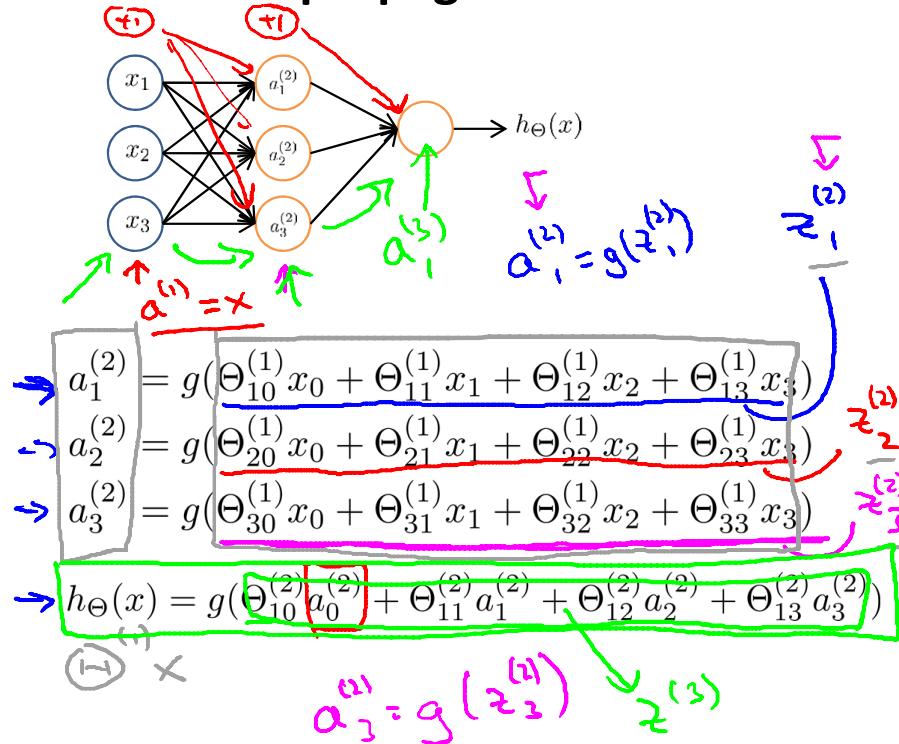
$$s_{j+1} \times (s_j + 1)$$



Machine Learning

Neural Networks: Representation Model representation II

Forward propagation: Vectorized implementation



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

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

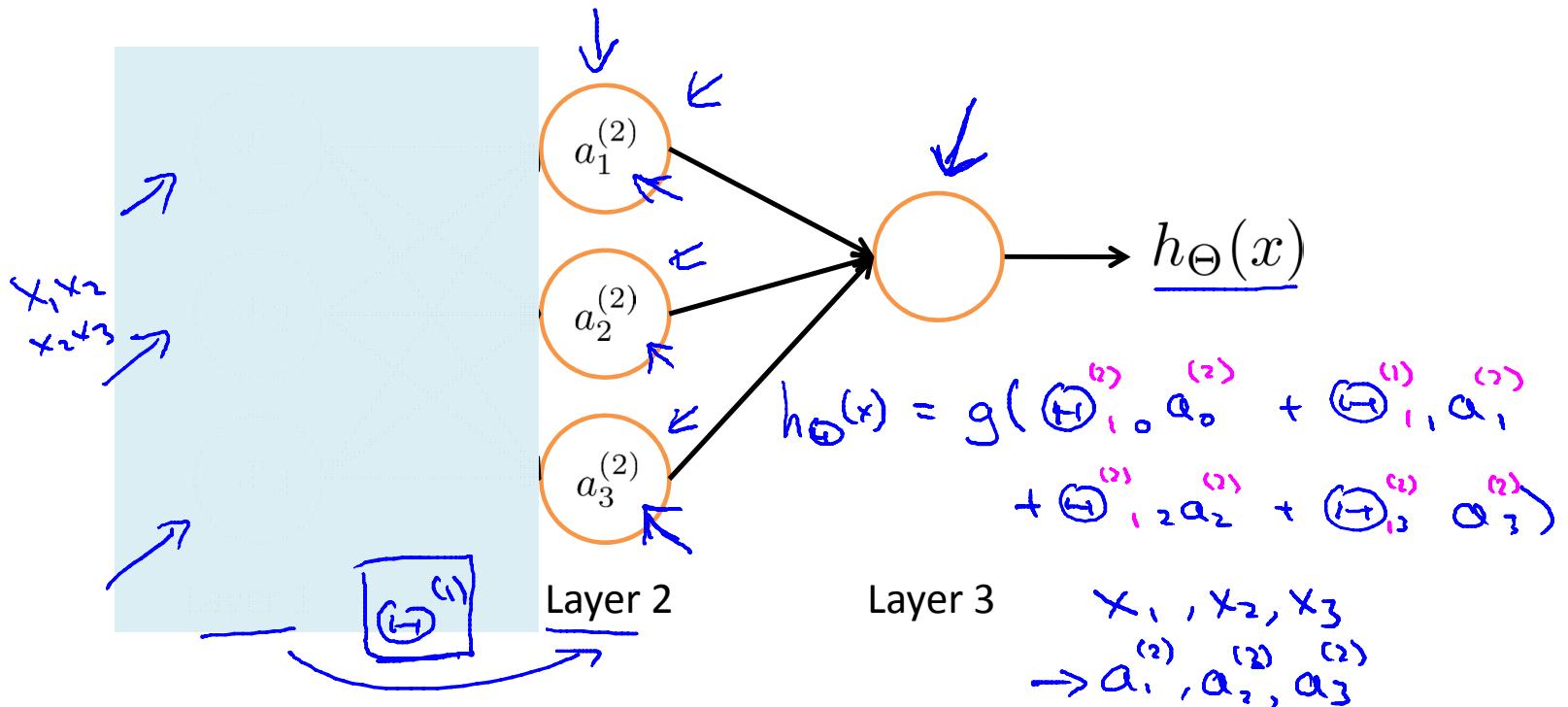
$$a^{(2)} = g(z^{(2)})$$

Add $a_0^{(2)} = 1$. $\rightarrow a^{(2)} \in \mathbb{R}^4$

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

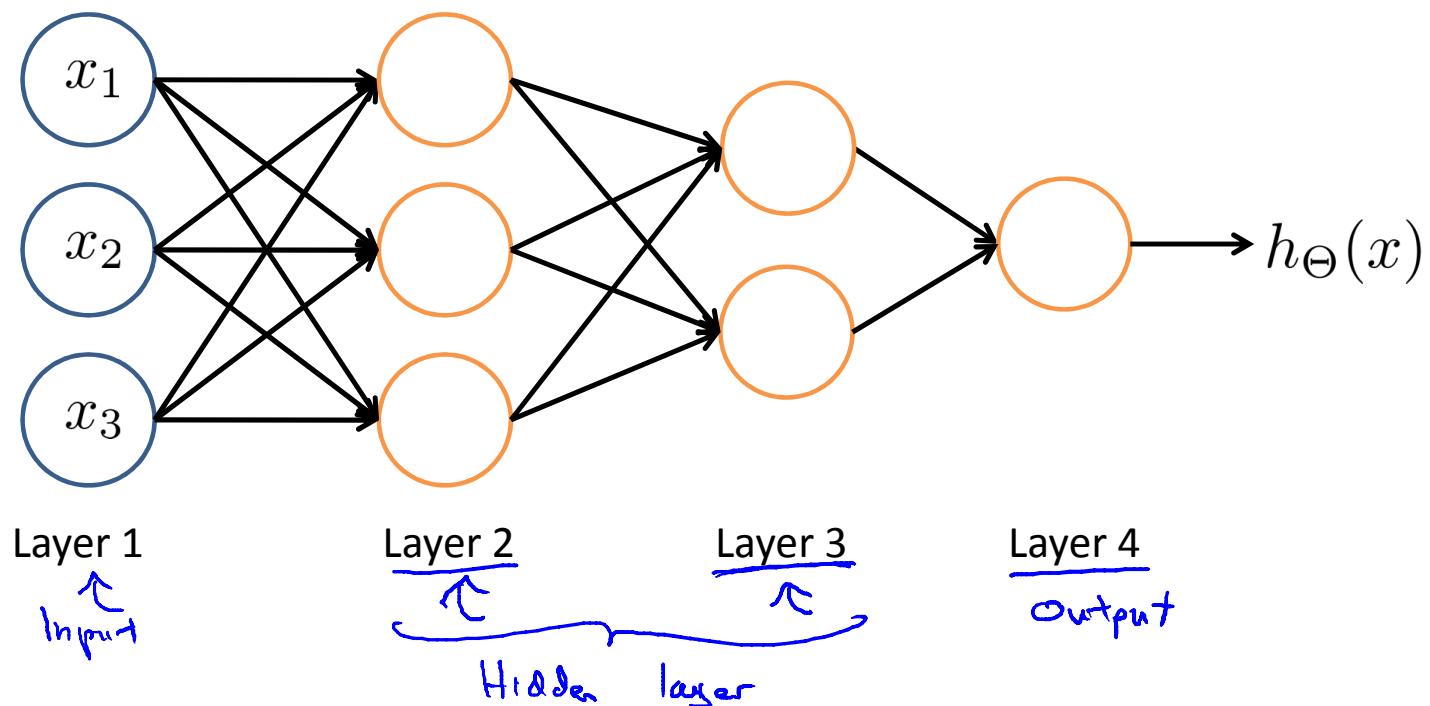
$$h_{\Theta}(x) = a^{(3)} = g(z^{(3)})$$

Neural Network learning its own features

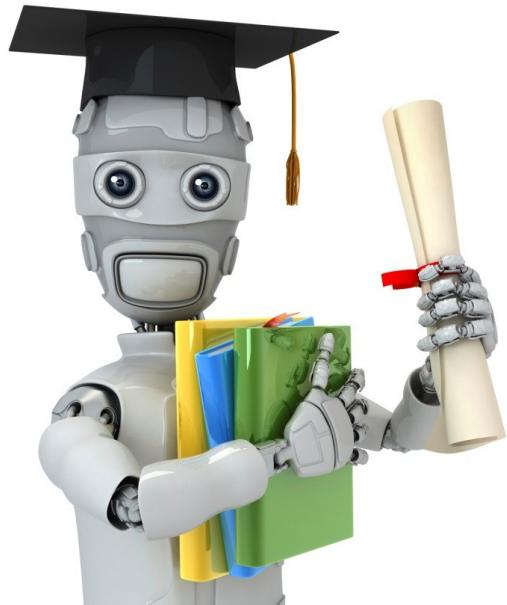


Andrew Ng

Other network architectures



Andrew Ng



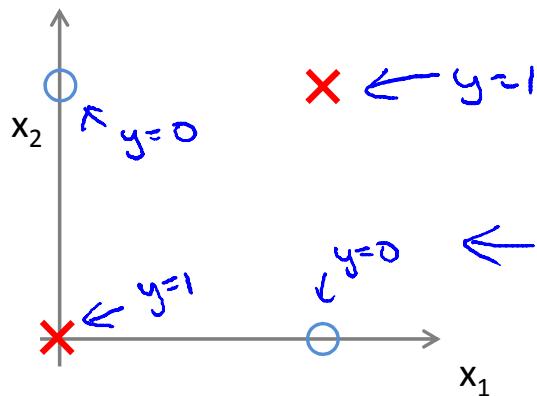
Machine Learning

Neural Networks: Representation

Examples and intuitions I

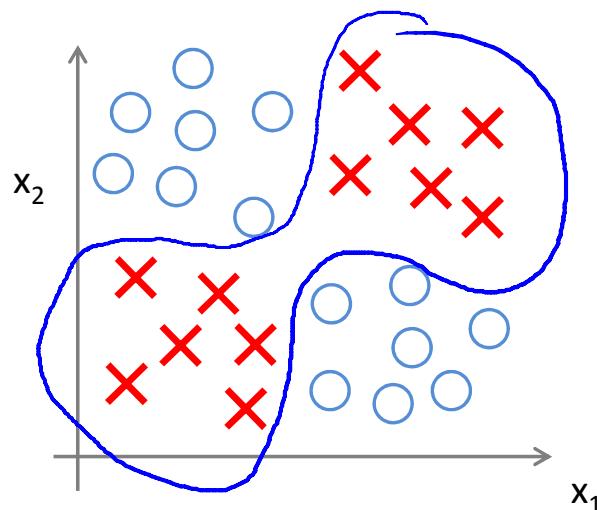
Non-linear classification example: XOR/XNOR

→ x_1, x_2 are binary (0 or 1).



$$y = \underline{x_1 \text{ XOR } x_2}$$

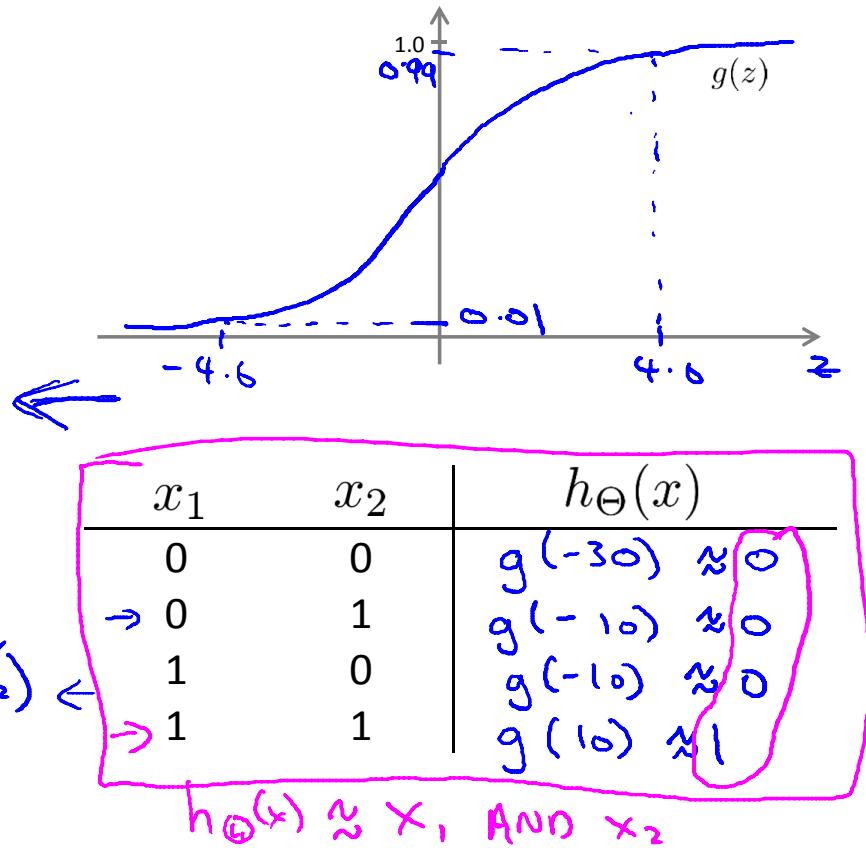
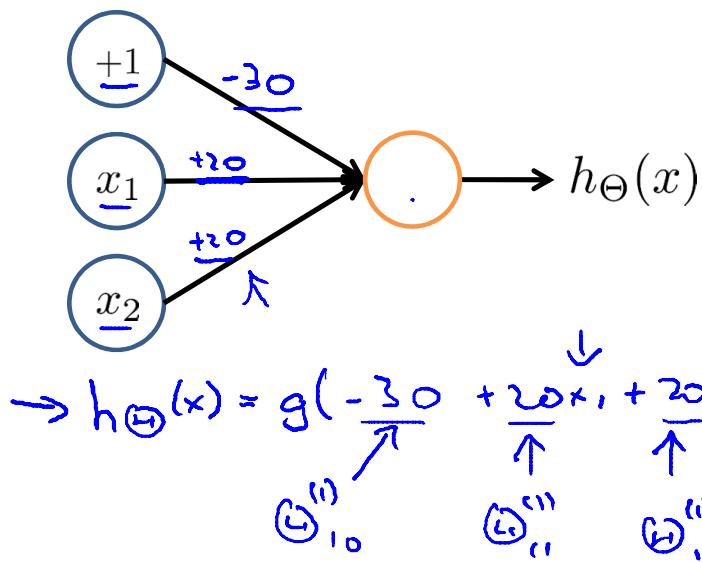
→ $\underline{x_1 \text{ XNOR } x_2}$ ←
→ NOT (x₁ XOR x₂)



Simple example: AND

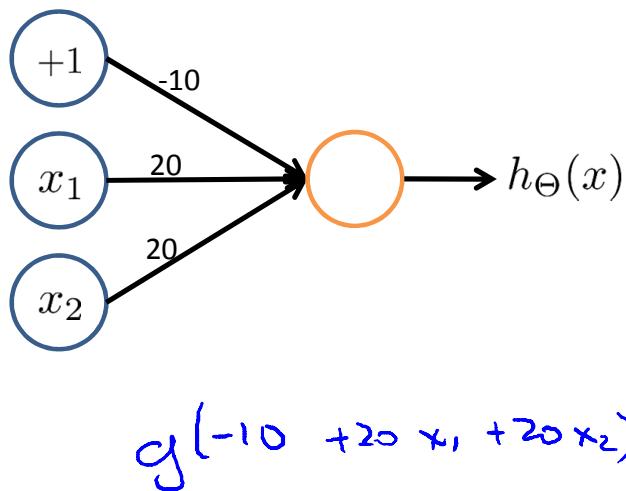
$$\rightarrow x_1, x_2 \in \{0, 1\}$$

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

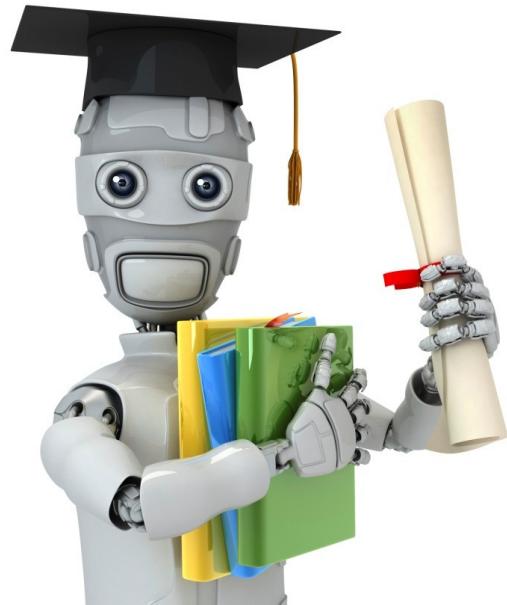


Andrew Ng

Example: OR function



x_1	x_2	$h_{\Theta}(x)$
0	0	$g(-10) \approx 0$
0	1	$g(10) \approx 1$
1	0	≈ 1
1	1	≈ 1



Machine Learning

Neural Networks: Representation

Examples and intuitions II

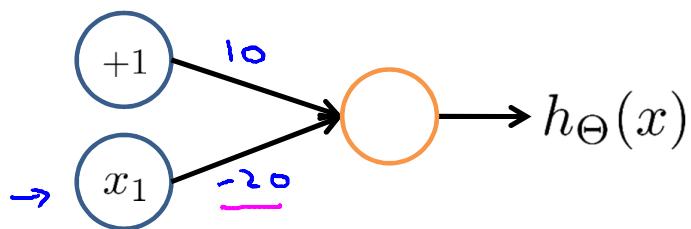
$\rightarrow x_1 \text{ AND } x_2$

$\rightarrow x_1 \text{ OR } x_2$

{0,1}.

Negation:

NOT x_1



x_1	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

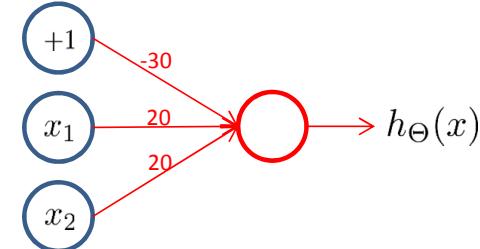
$$h_\Theta(x) = g(10 - 20x_1)$$

$\rightarrow (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$

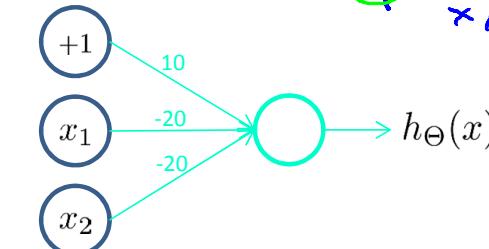
$\underbrace{\quad}_{\leq 1 \text{ if and only if}} \quad \rightarrow x_1 = x_2 = 0$

Andrew Ng

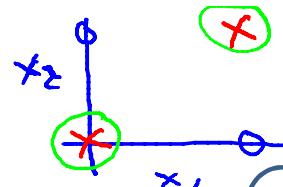
Putting it together: $x_1 \text{ XNOR } x_2$



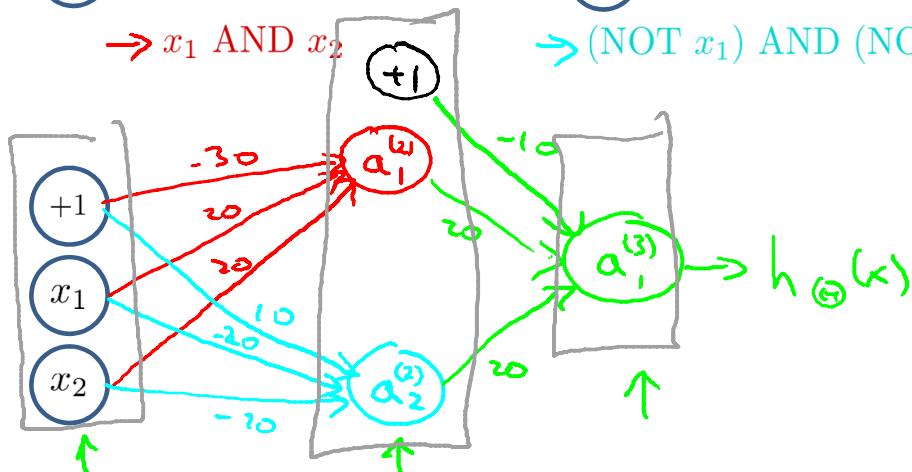
$\rightarrow x_1 \text{ AND } x_2$



$\rightarrow (\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)$



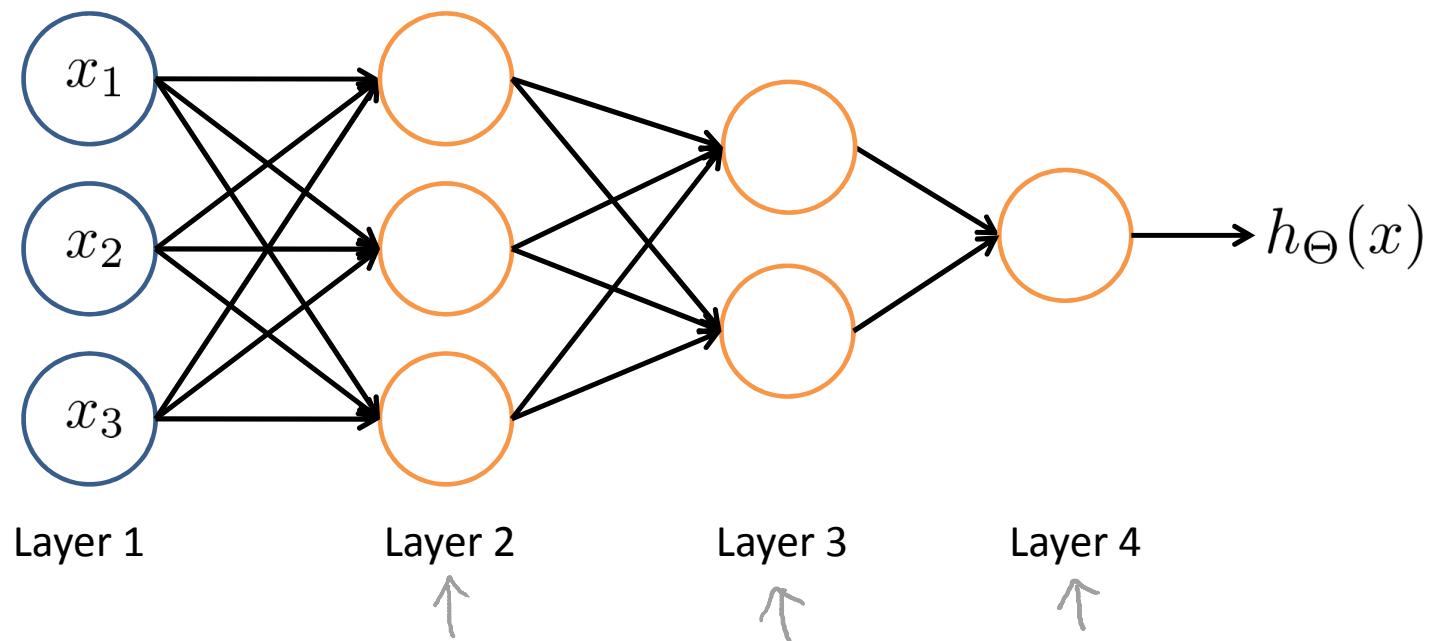
$\rightarrow x_1 \text{ OR } x_2$



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

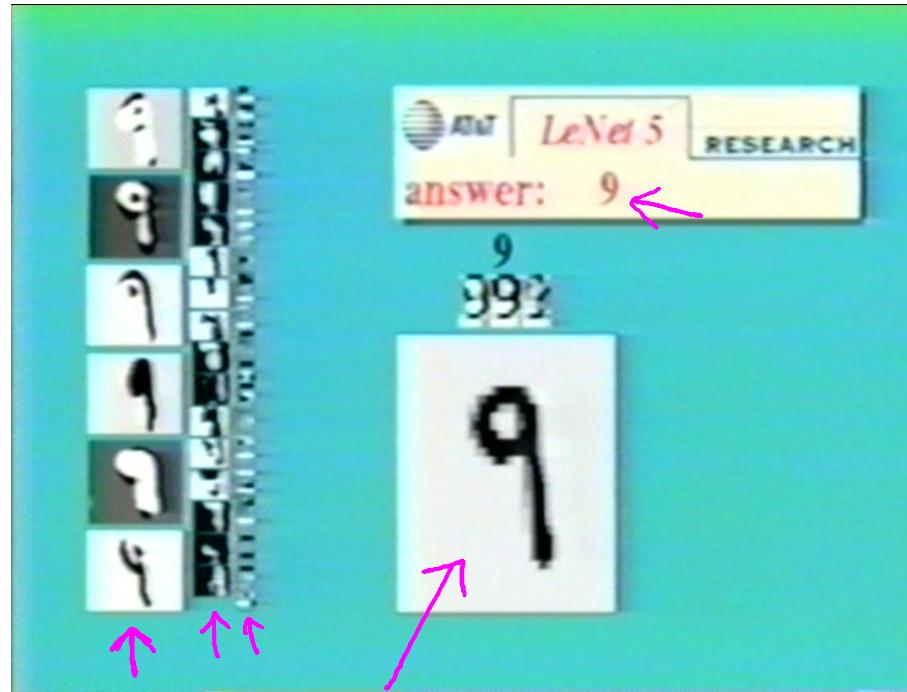
Andrew Ng

Neural Network intuition



Andrew Ng

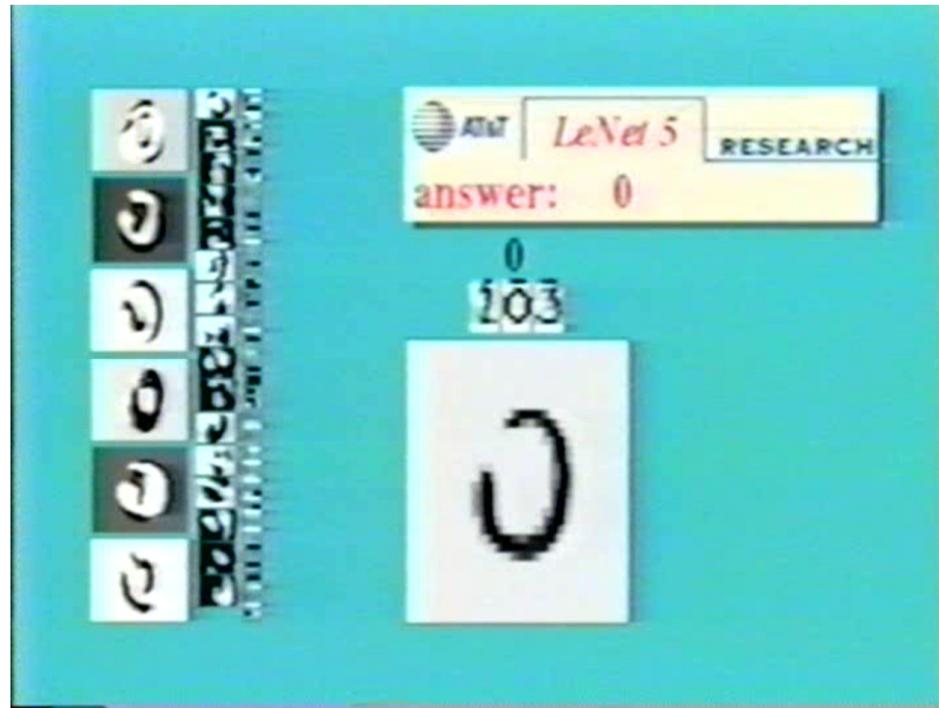
Handwritten digit classification



[Courtesy of Yann LeCun] ↵

Andrew Ng

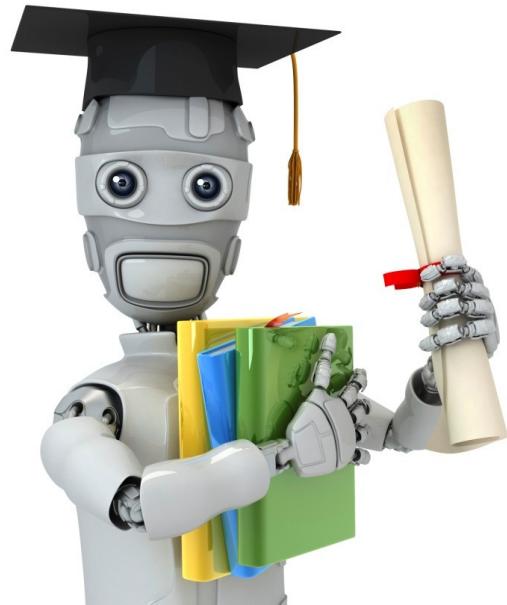
Handwritten digit classification



[Courtesy of Yann LeCun]

Andrew Ng

Andrew Ng



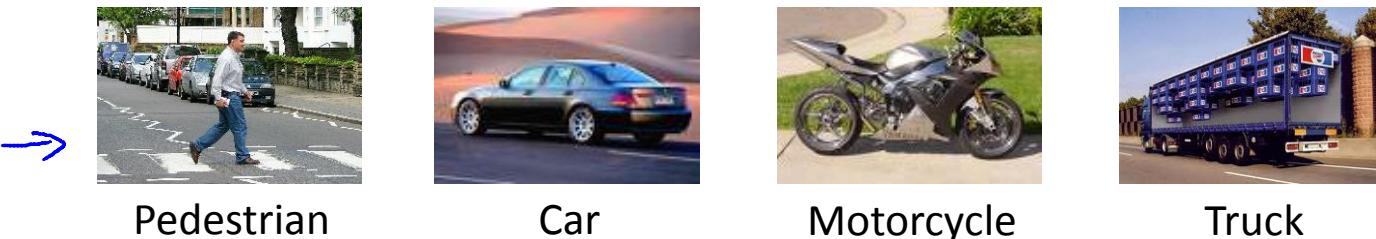
Machine Learning

Neural Networks: Representation

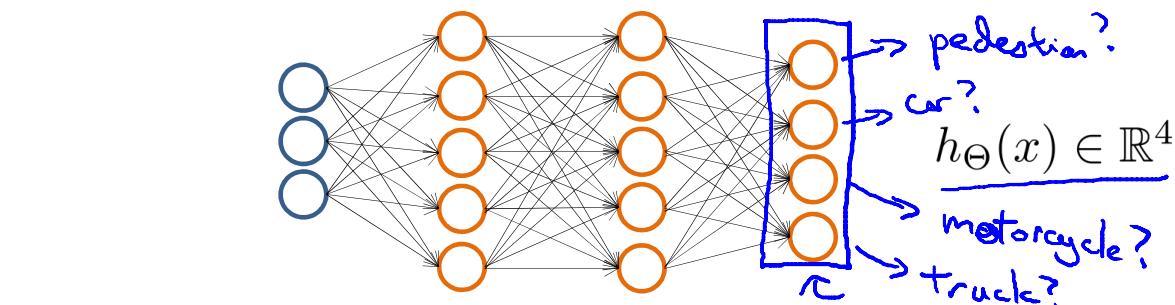
Multi-class classification

Andrew Ng

Multiple output units: One-vs-all.



Pedestrian Car Motorcycle Truck

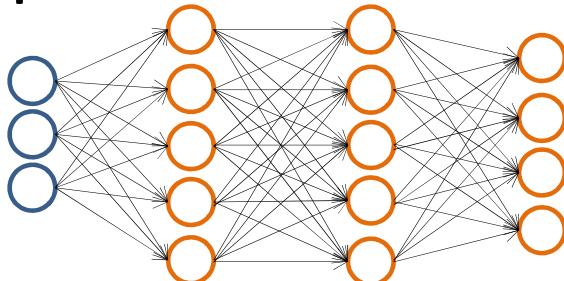


Want $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$,
when pedestrian

$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$,
when car

$h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.
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: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

$\Rightarrow y^{(i)}$ 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

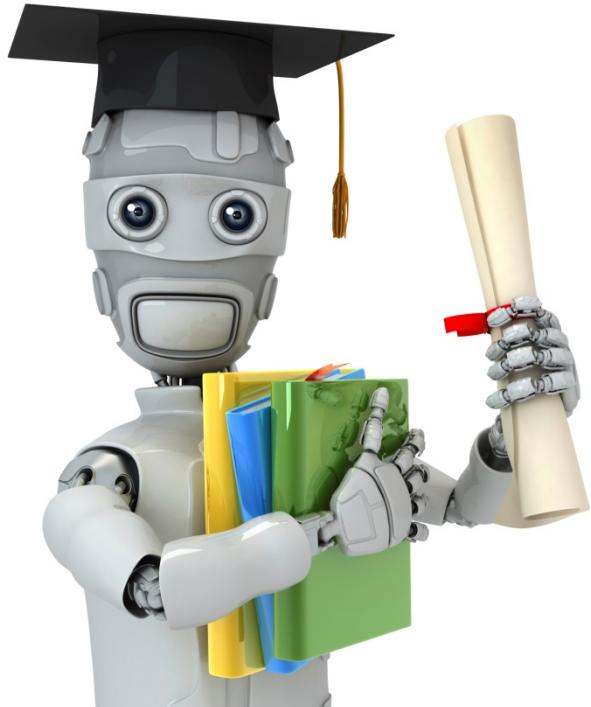
$(x^{(i)}, y^{(i)})$

~~Previously~~
 $y \in \{1, 2, 3, 4\}$
 $\underline{h_{\Theta}(x^{(i)}) \approx y^{(i)}} \in \mathbb{R}^4$

Andrew Ng

Andrew Ng

4.2 Learning

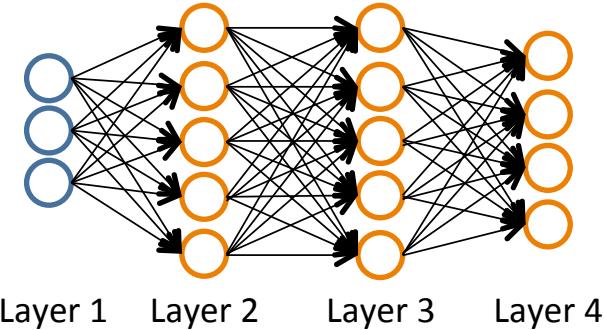


Machine Learning

Neural Networks: Learning

Cost function

Neural Network (Classification)



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

L = total no. of layers in network

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

Binary classification

$$y = 0 \text{ or } 1$$

1 output unit

Multi-class classification (K classes)

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

pedestrian car motorcycle truck

K output units

Cost function

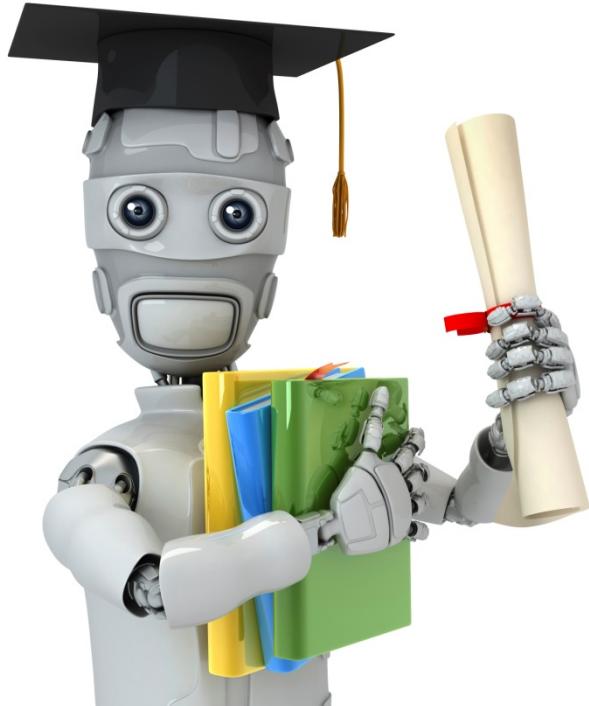
Logistic regression:

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

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

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

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

Need code to compute:

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

$$\Theta_{ij}^{(l)} \in \mathbb{R}$$

Gradient computation

Given one training example $\underline{(x, y)}$:

Forward propagation:

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

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

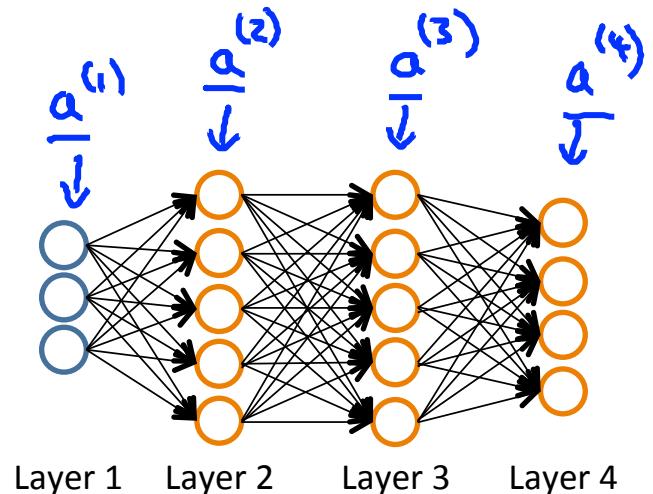
$$\rightarrow a^{(2)} = g(z^{(2)}) \quad (\text{add } \underline{a_0^{(2)}})$$

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

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

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

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

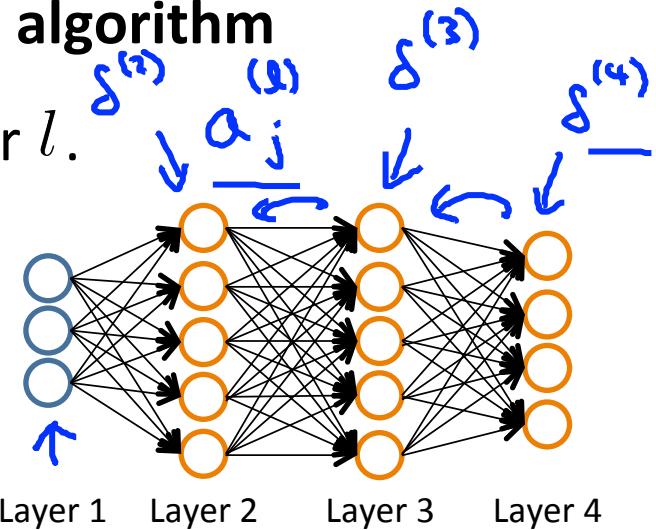


Gradient computation: Backpropagation algorithm

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

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

$$\underline{\delta_j^{(4)}} = \underline{a_j^{(4)}} - \underline{y_j} \quad (\underline{h_{\Theta}(x)})_j \quad \underline{\delta^{(4)}} = \underline{a^{(4)}} - \underline{y}$$



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

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

$$\frac{a^{(3)}}{a^{(2)}} * \frac{(1-a^{(3)})}{(1-a^{(2)})}$$

$$\frac{\partial}{\partial \Theta_{ij}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)} \quad (\text{ignoring } \lambda; \text{ if } \lambda = 0)$$

Backpropagation algorithm

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

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

For $i = 1$ to m \leftarrow $(x^{(i)}, y^{(i)})$

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

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

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

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

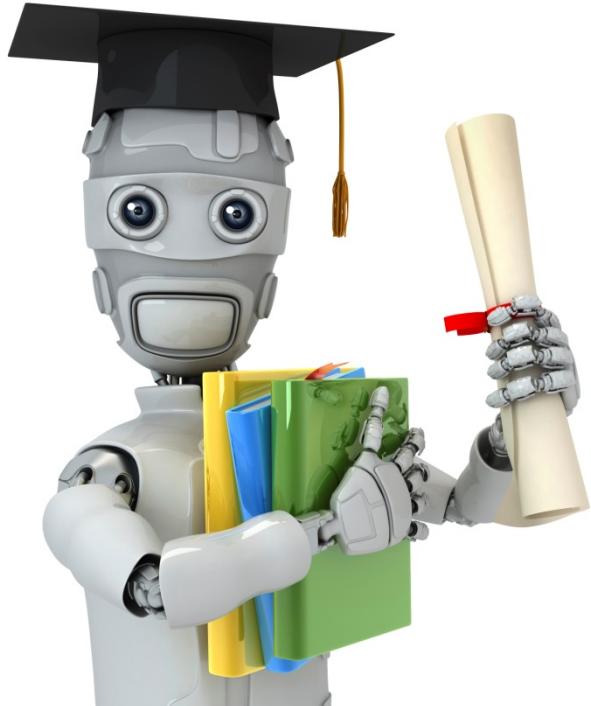
$$\Delta_{ii}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

$$\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (\alpha^{(l)})^T.$$

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

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

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

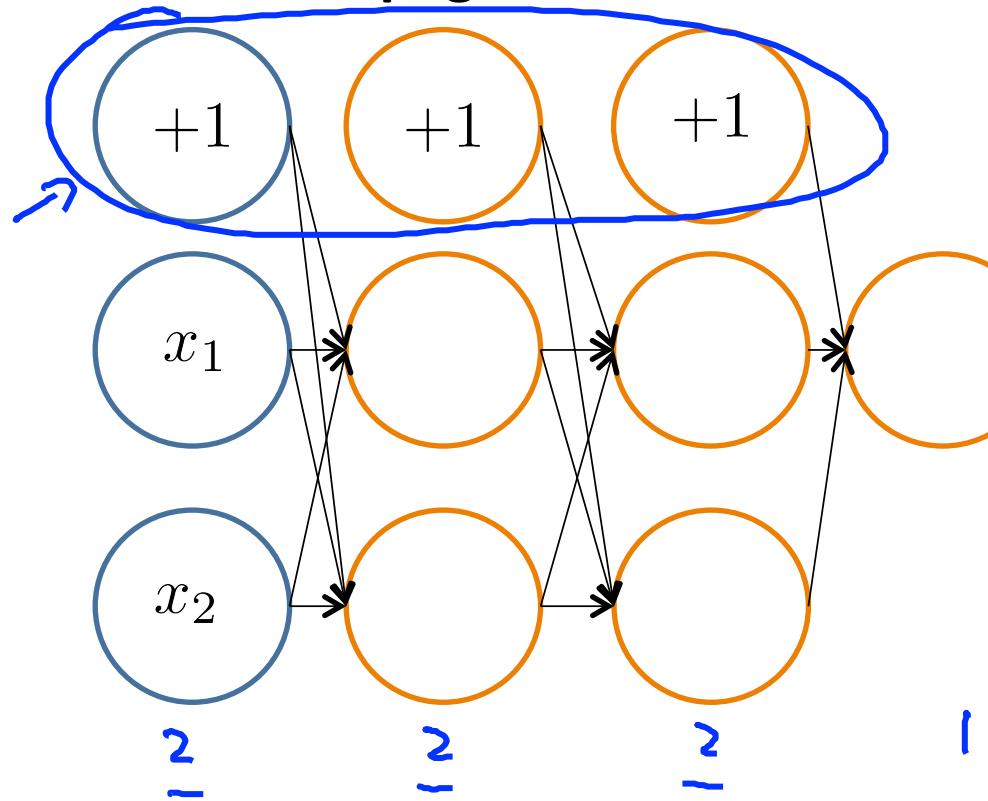


Machine Learning

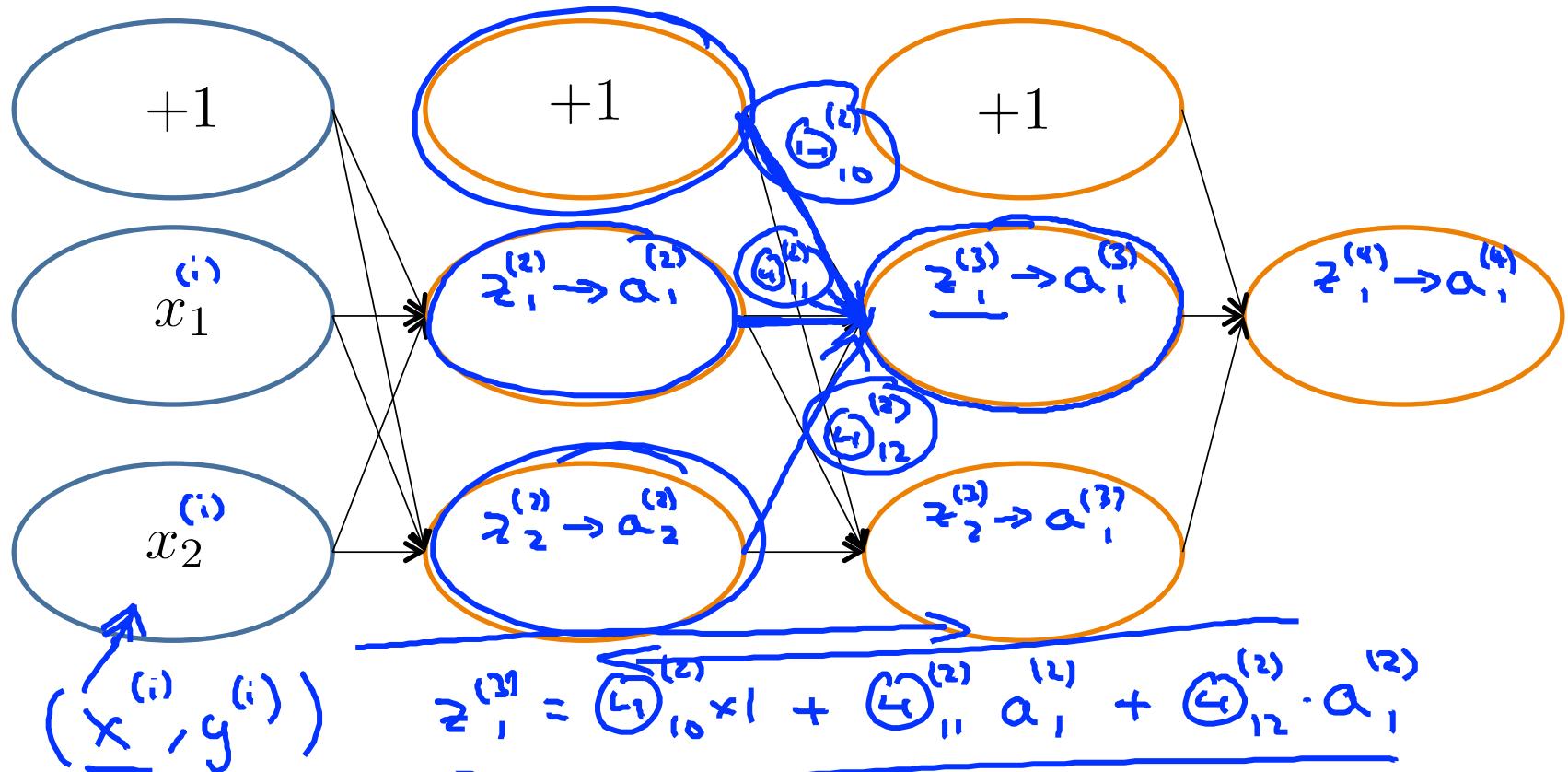
Neural Networks: Learning

Backpropagation intuition

Forward Propagation



Forward Propagation



What is backpropagation doing?

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

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

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

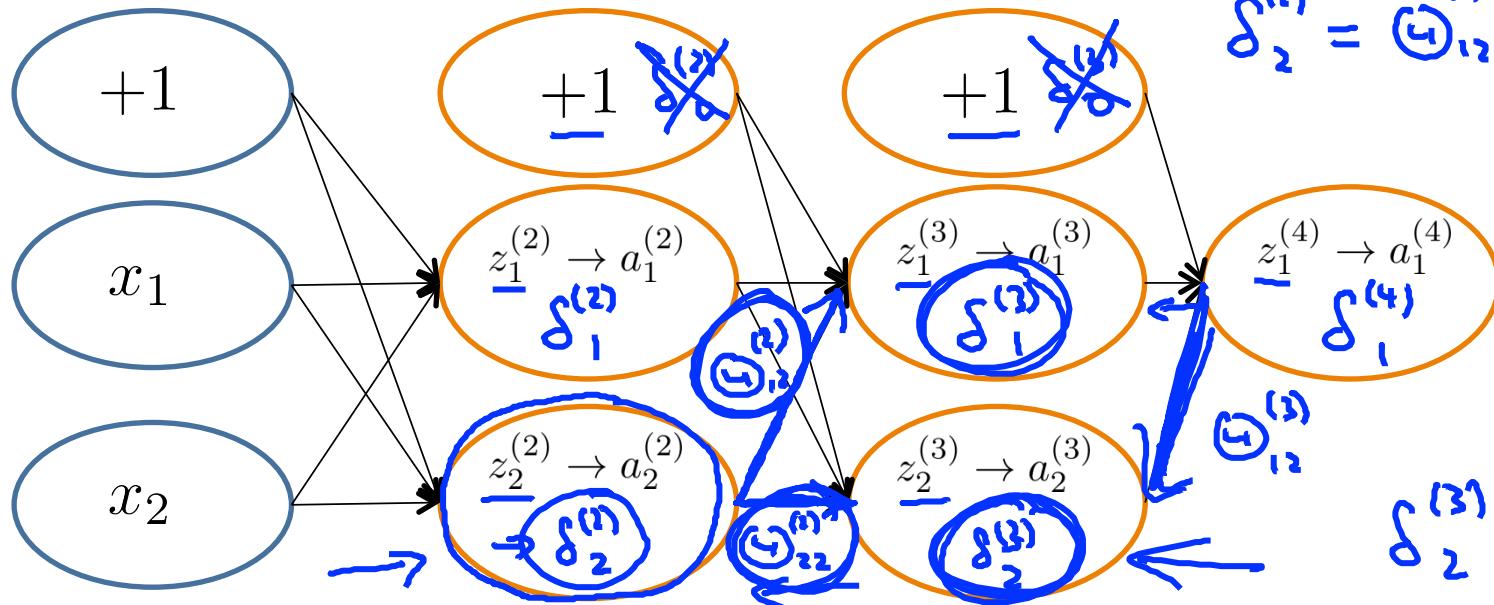
$$\text{cost}(i) = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log h_\Theta(x^{(i)})$$

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

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

$y^{(i)}$

Forward Propagation



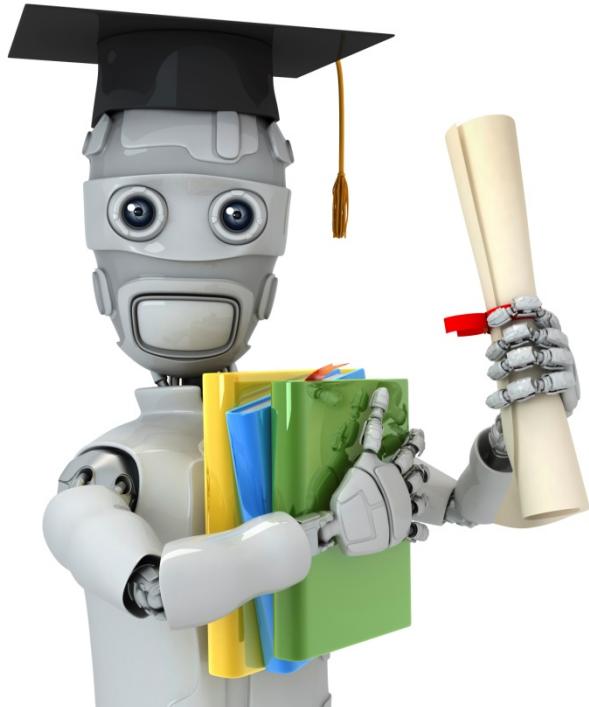
$\rightarrow \delta_j^{(l)}$ = “error” of cost for $a_j^{(l)}$ (unit j in layer l).

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

$$\delta_1^{(4)} = \underline{y^{(i)}} - \underline{a_1^{(4)}}$$

$$\delta_2^{(2)} = \underline{G_{1,2}^{(2)}} \delta_1^{(3)} + \underline{G_{2,2}^{(2)}} \delta_2^{(3)}$$

$$\delta_2^{(3)} = \underline{G_{1,2}^{(3)}} \cdot \delta_1^{(4)}$$



Machine Learning

Neural Networks:

Learning

Implementation note: Unrolling parameters

Advanced optimization

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

Annotations: Handwritten arrows point from the gradient parameter to \mathbb{R}^{n+1} , from the theta parameter to \mathbb{R}^{n+1} labeled "(vectors)", and from the options parameter to a blue arrow pointing upwards.

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

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

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

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

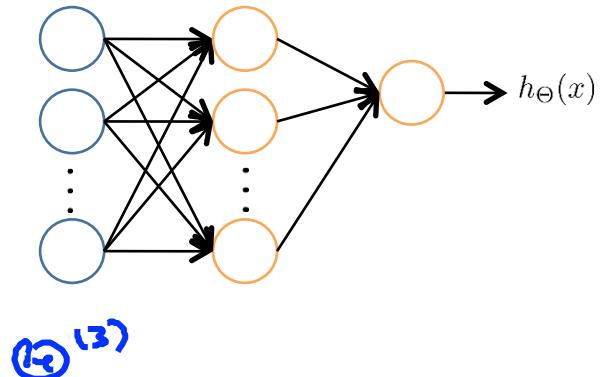
→ `thetaVec = [Theta1(:); Theta2(:); Theta3(:)];`

→ `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);`

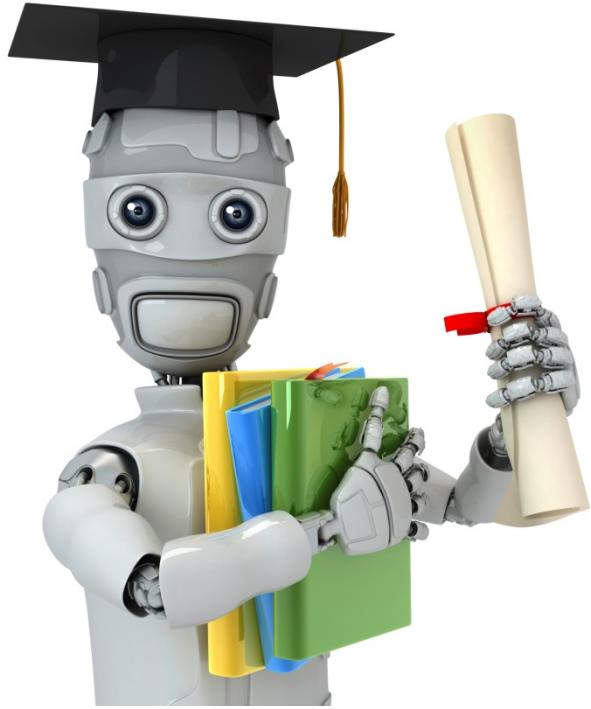


Learning Algorithm

- 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 $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$. reshape
- Use forward prop/back prop to compute $D^{(1)}, D^{(2)}, D^{(3)}$ and $D_{\cdot}^{(1)}, D_{\cdot}^{(2)}, D_{\cdot}^{(3)}$
- Unroll gradientVec to get gradientVec.



Machine Learning

Neural Networks: Learning

Gradient checking

Numerical estimation of gradients



$$\frac{\partial}{\partial \theta} J(\theta) \approx$$

$$\frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}$$

$$\epsilon = 10^{-4}$$

$$\cancel{J(\theta + \epsilon) - J(\theta)}$$

Implement: gradApprox = $(J(\text{theta} + \text{EPSILON}) - J(\text{theta} - \text{EPSILON})) / (2 * \text{EPSILON})$

Parameter vector θ

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

→ $\theta = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$

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

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

⋮

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

```

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;

```

$\frac{\partial}{\partial \theta_i} J(\theta)$.

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_i + \epsilon \\ \vdots \\ \theta_n \end{bmatrix} \rightarrow \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_i - \epsilon \\ \vdots \\ \theta_n \end{bmatrix}$$

Check that gradApprox \approx DVec

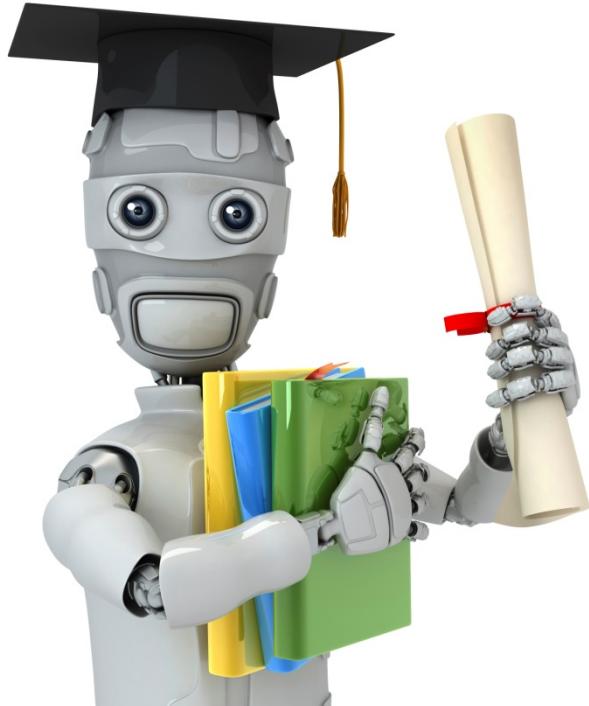
From back prop.

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

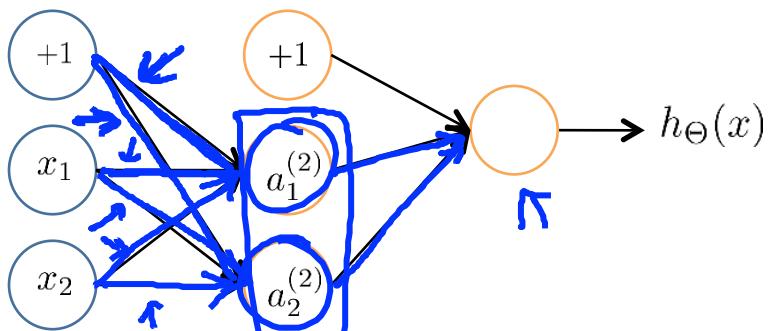
For gradient descent and advanced optimization method, need initial value for Θ .

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

Consider gradient descent

Set initialTheta = zeros(n,1) ?

Zero initialization



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

Also $\delta_i^{(l)} = \ell_i^{(l)}$.

$$\frac{\partial}{\partial \Theta_{01}^{(l)}} J(\Theta) = \frac{\partial}{\partial \Theta_{02}^{(l)}} J(\Theta)$$

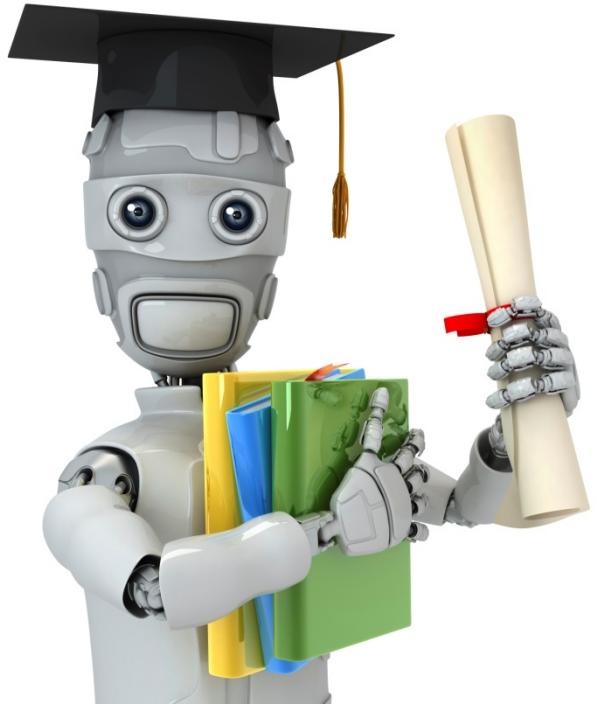
$$\underline{\Theta_{01}^{(l)}} = \underline{\Theta_{02}^{(l)}}$$

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

$$\underline{\Theta_{01}^{(l)}} = \underline{\Theta_{02}^{(l)}}$$

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.
- Random 10×11 matrix (betw. 0 and 1)
- Theta1 = $\boxed{\text{rand}(10, 11) * (2 * \text{INIT_EPSILON})}$ $[-\epsilon, \epsilon]$
- INIT_EPSILON;
- Theta2 = $\boxed{\text{rand}(1, 11) * (2 * \text{INIT_EPSILON})}$
- INIT_EPSILON;



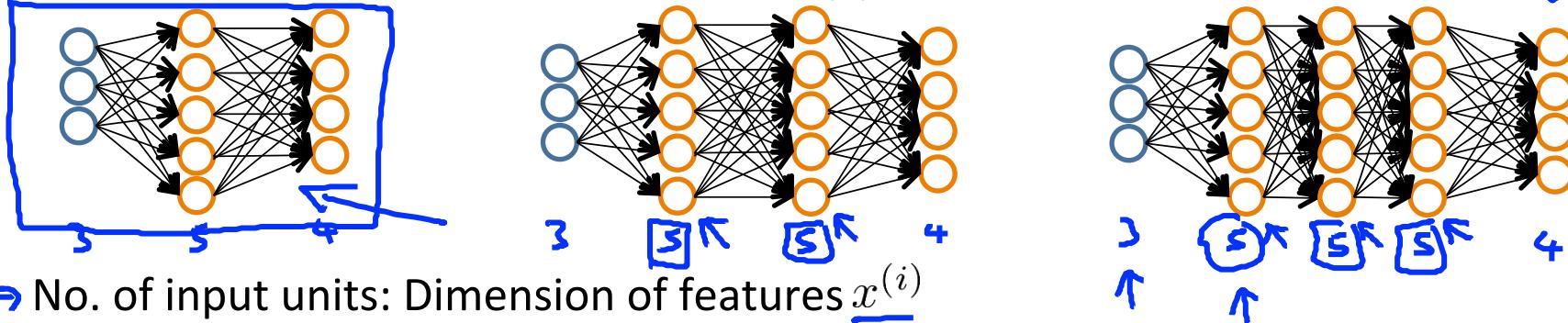
Machine Learning

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)

↑

$$y \in \{1, 2, 3, \dots, 103\}$$

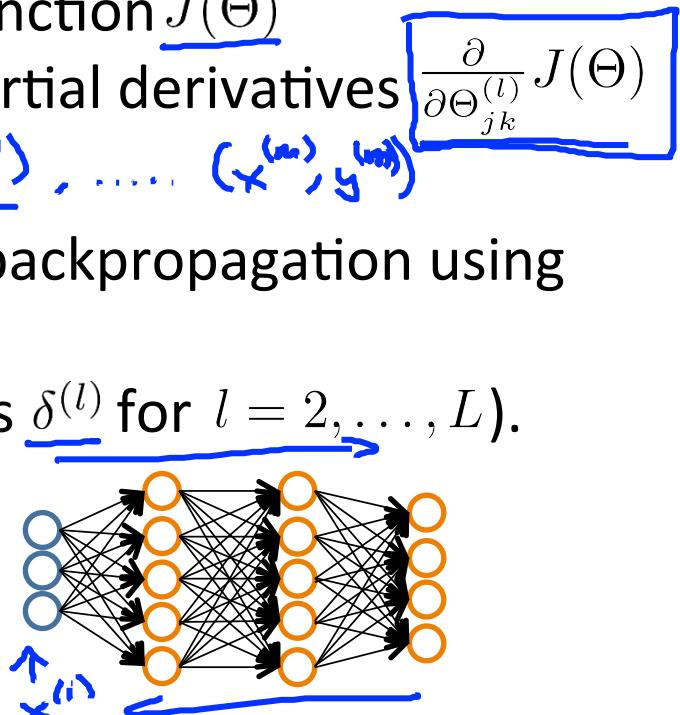
~~$y \in \mathbb{R}$~~

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

←

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 $i = 1:m$ { $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, ..., $(x^{(m)}, y^{(m)})$
 - Perform forward propagation and backpropagation using example $(x^{(i)}, y^{(i)})$
 - Get activations $a^{(l)}$ and delta terms $\delta^{(l)}$ for $l = 2, \dots, L$.
- $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l)} (a^{(l)})^T$
- ...
- ...
 - compute $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$.

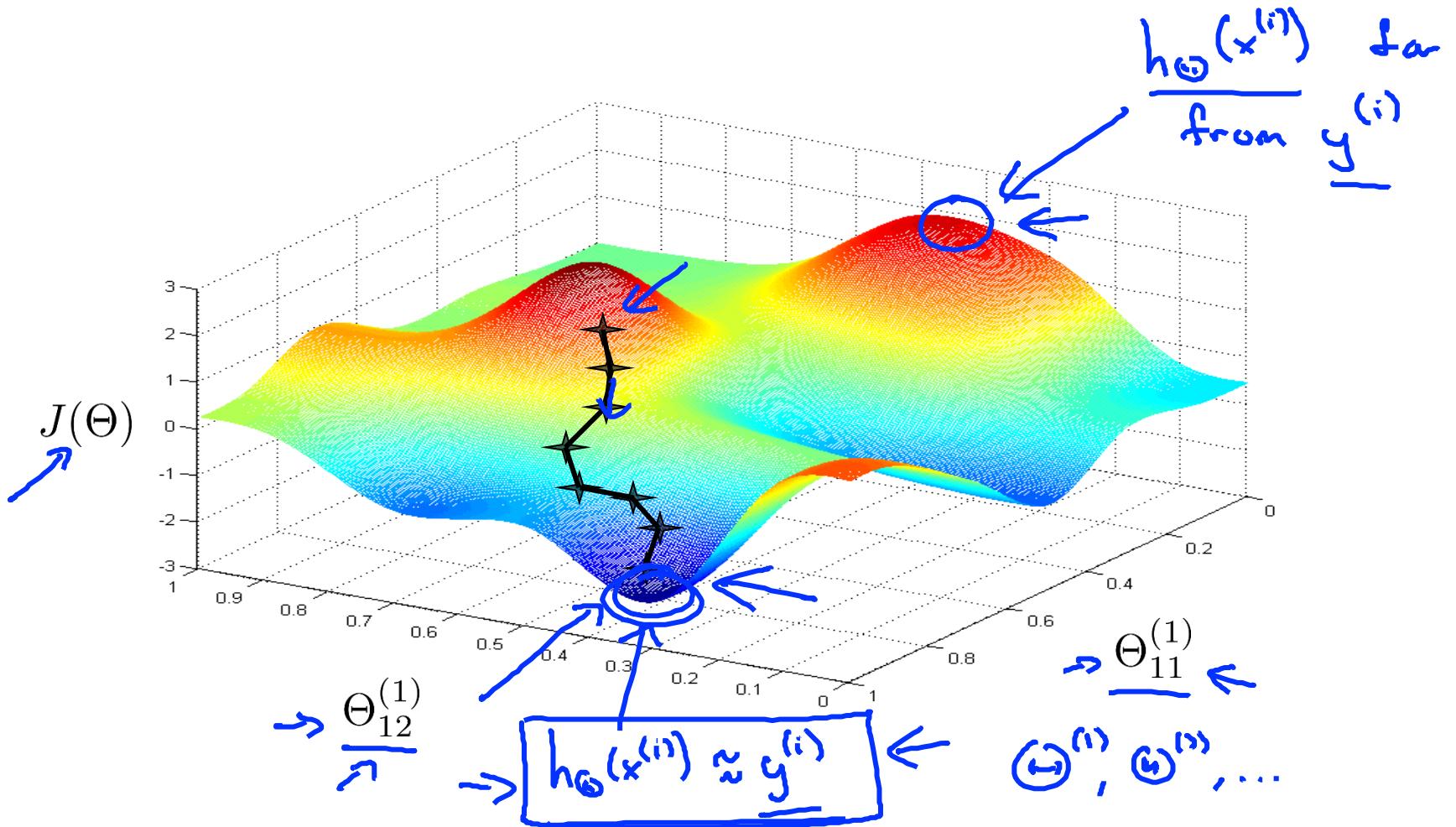


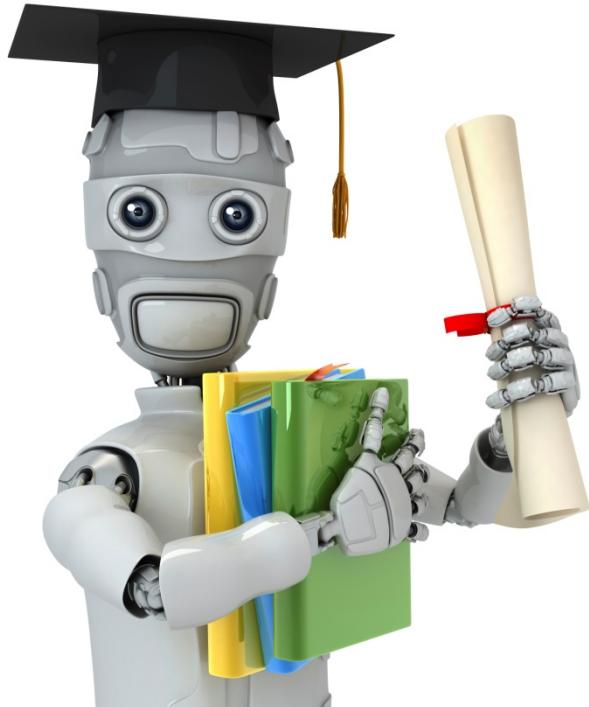
Training a neural network

- 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

$$\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta) \quad \overbrace{\qquad\qquad\qquad}$$

$J(\Theta)$ — non-convex.

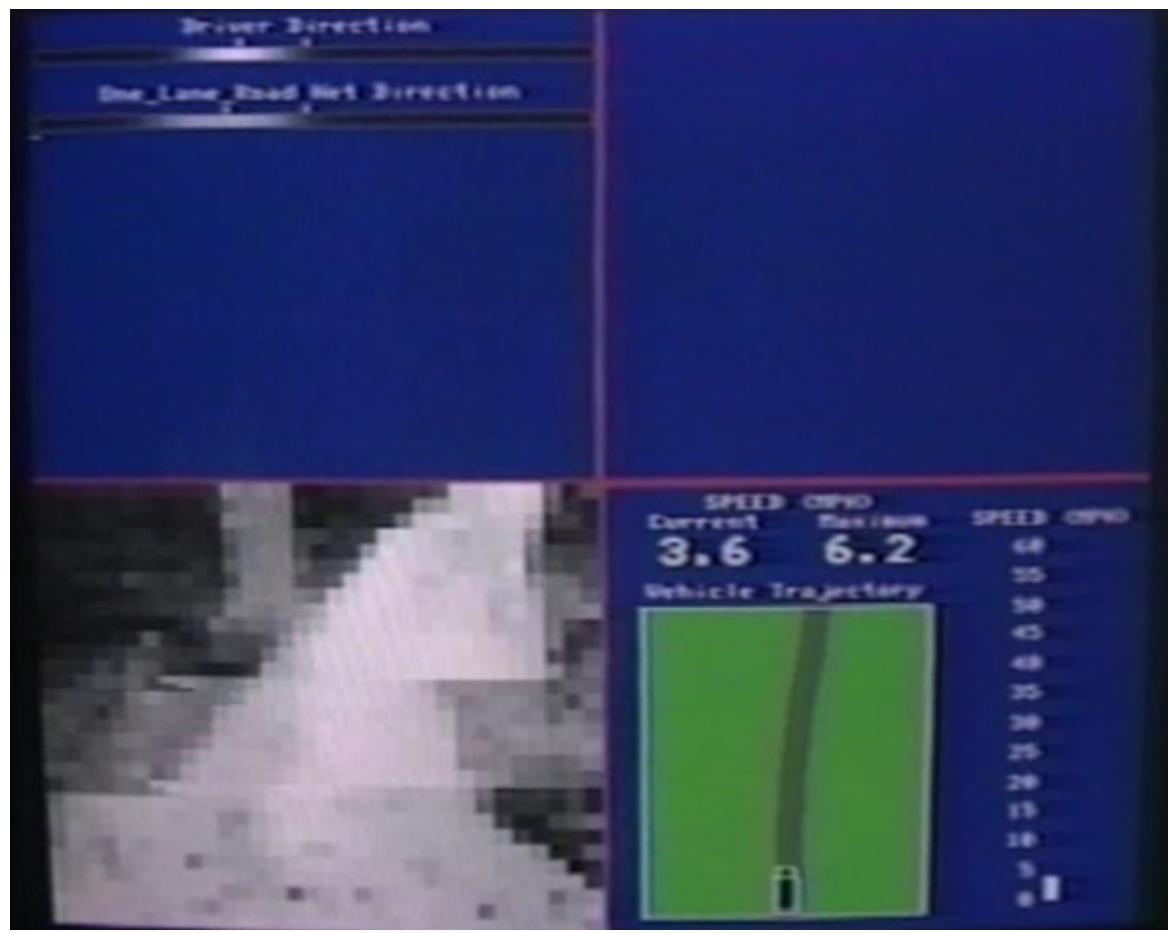




Machine Learning

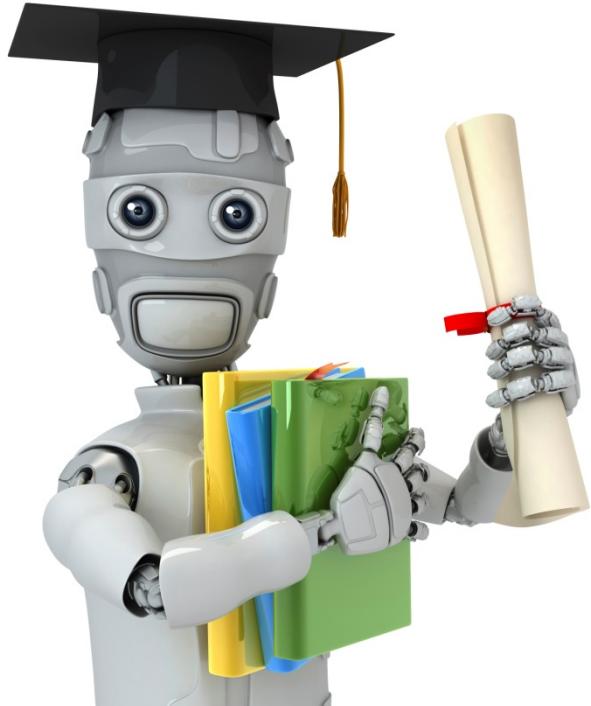
Neural Networks: Learning

Backpropagation
example: Autonomous
driving (optional)



[Courtesy of Dean Pomerleau]

4.3 NN



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.

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

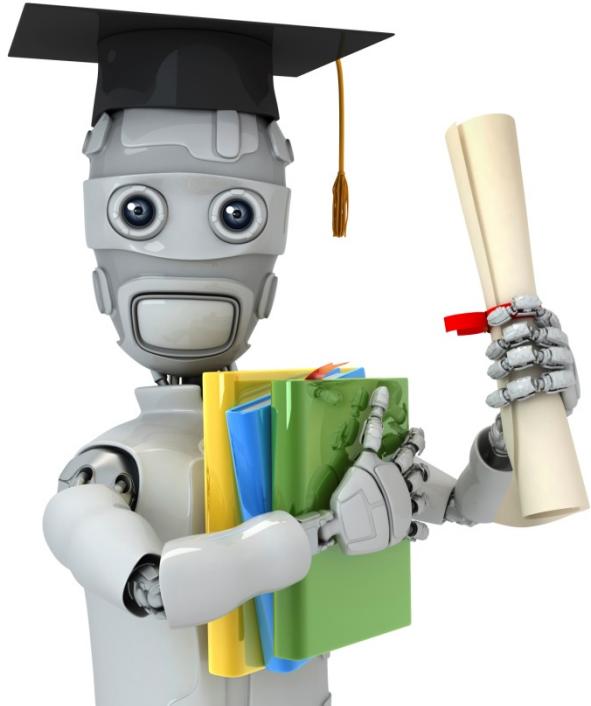
However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

- - Get more training examples
- Try smaller sets of features $x_1, x_2, x_3, \dots, x_{100}$
- - 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.

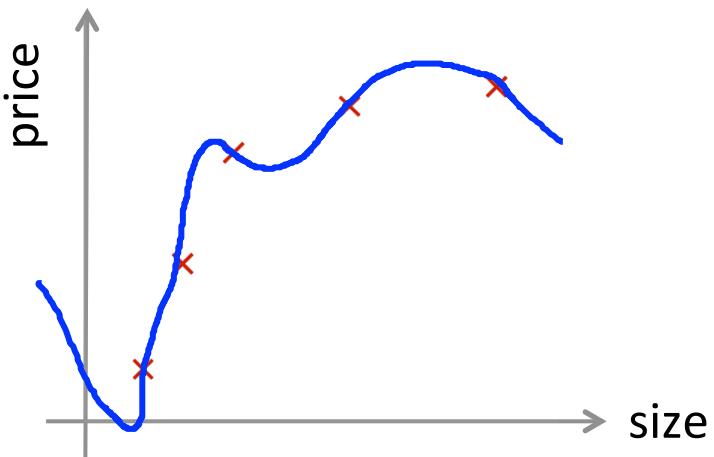


Machine Learning

Advice for applying
machine learning

Evaluating a
hypothesis

Evaluating your hypothesis



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

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
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
<hr/>	
1427	199
1380	212
1494	243

70% Training set

30% Test set

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

$(x_{test}^{(1)}, y_{test}^{(1)})$
 $(x_{test}^{(2)}, y_{test}^{(2)})$
⋮
 $(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

m_{test} = no. of test example
 $(x_{test}^{(1)}, y_{test}^{(1)})$

Training/testing procedure for linear regression

- - Learn parameter $\underline{\theta}$ from training data (minimizing training error $J(\theta)$)
 $\underbrace{J(\theta)}$ 70%
- Compute test set error:

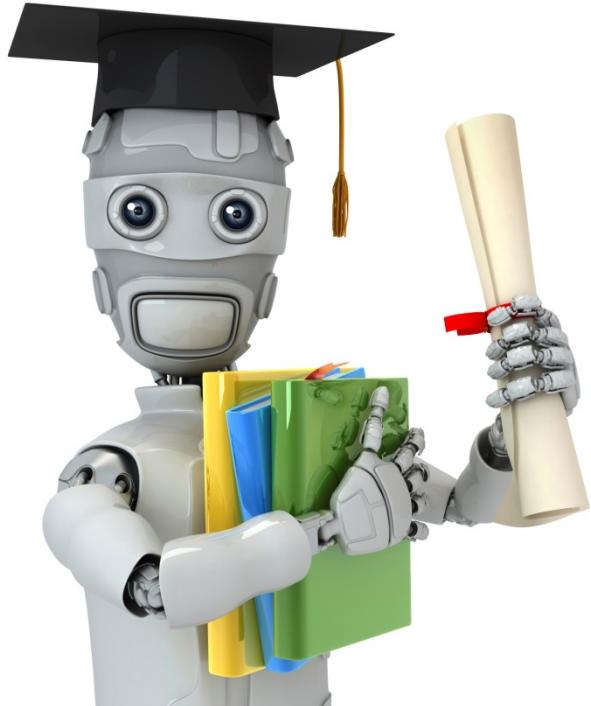
$$J_{\text{test}}(\theta) = \frac{1}{2m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \left(\underline{h_{\theta}(x_{\text{test}}^{(i)}) - y_{\text{test}}^{(i)}} \right)^2$$

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):

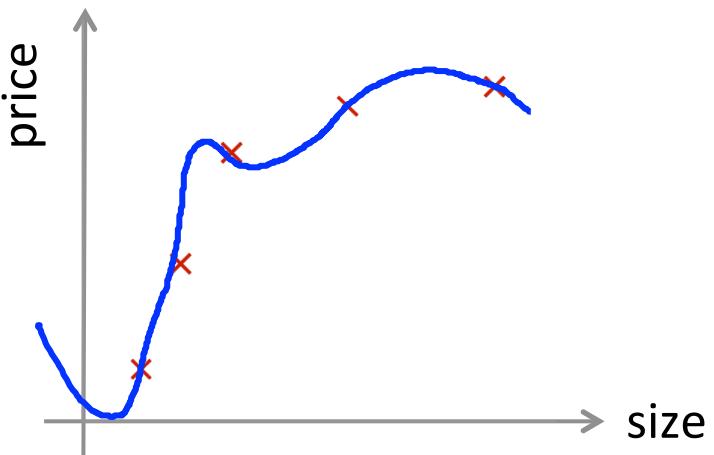


Machine Learning

Advice for applying machine learning

Model selection and
training/validation/test
sets

Overfitting example



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

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

$$d=1 \quad 1. \quad h_{\theta}(x) = \theta_0 + \theta_1 x$$

$$d=2 \quad 2. \quad h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

$$d=3 \quad 3. \quad h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_3 x^3$$

⋮

⋮

$$d=10 \quad 10. \quad h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10}$$

$\rightarrow d = \text{degree of polynomial}$

$$\Theta^{(1)} \rightarrow J_{test}(\Theta^{(1)})$$

$$\Theta^{(2)} \rightarrow J_{test}(\Theta^{(2)})$$

$$\Theta^{(3)} \rightarrow J_{test}(\Theta^{(3)})$$

⋮

$$\Theta^{(10)} \rightarrow J_{test}(\Theta^{(10)})$$

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

$\Theta^{(5)}$

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

$\Theta_0, \Theta_1, \dots$

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.

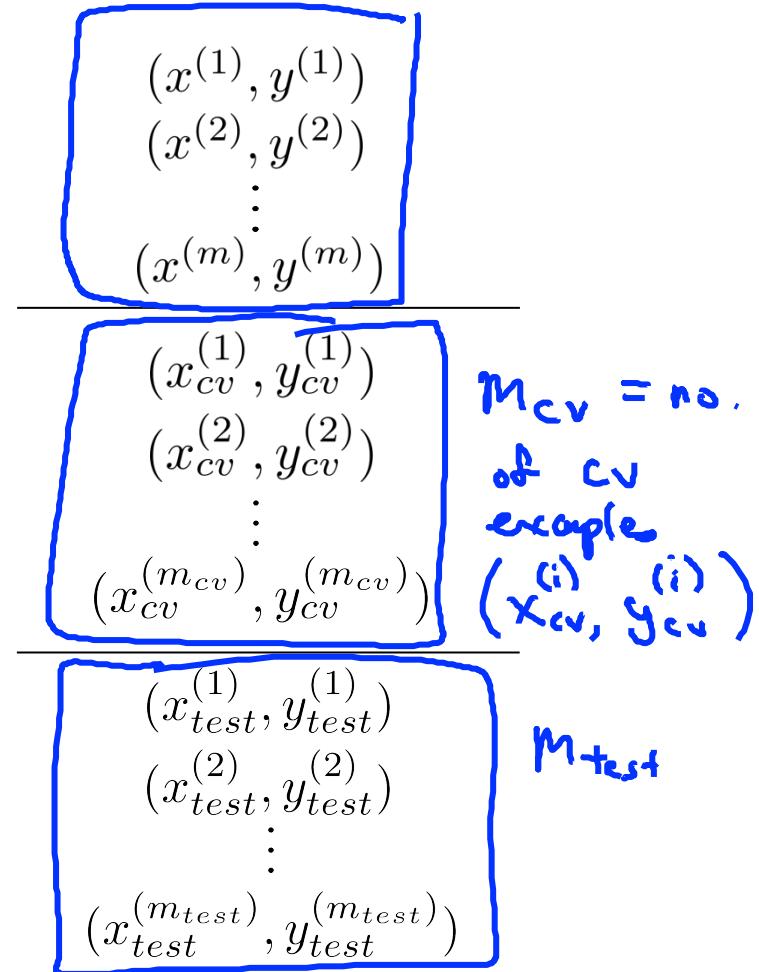
Evaluating your hypothesis

Dataset:

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

Annotations:

- A blue bracket on the left side of the table is labeled "60%".
- A blue bracket on the right side of the first six rows is labeled "Training set".
- A blue bracket on the right side of the last two rows is labeled "Test set".
- A blue bracket on the right side of the last two rows is labeled "Cross validation set (CV)".
- Two arrows point from the "Training set" and "Cross validation set (CV)" brackets to boxes containing training data points.
- An arrow points from the "Test set" bracket to a box containing test data points.



Train/validation/test error

Training error:

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

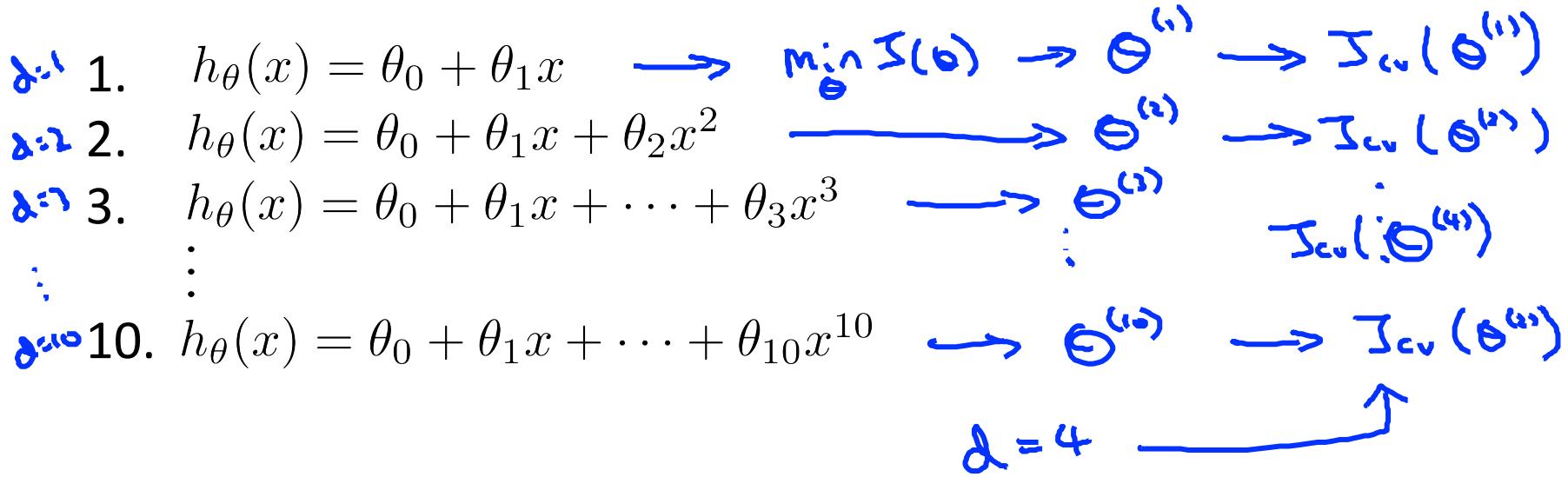
Cross Validation error:

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

Test error:

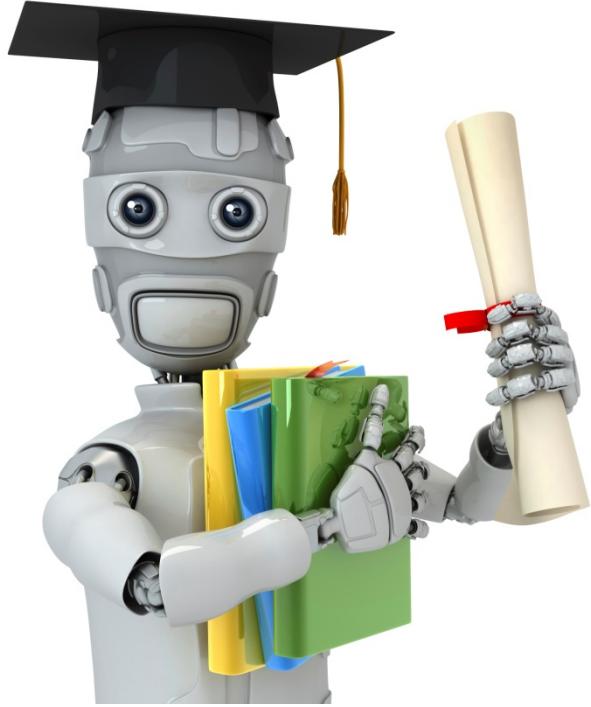
$$\rightarrow J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

Model selection



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

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

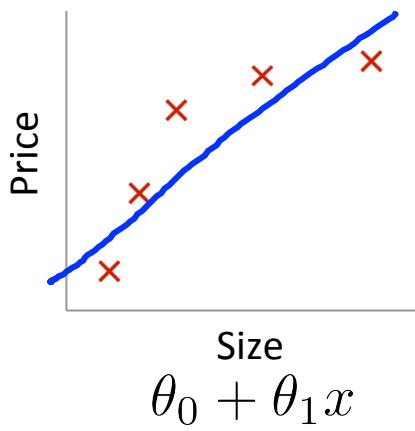


Machine Learning

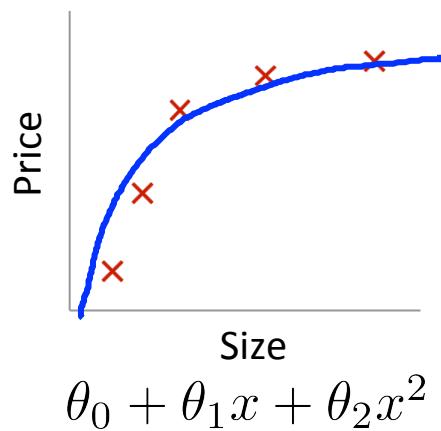
Advice for applying machine learning

Diagnosing bias vs. variance

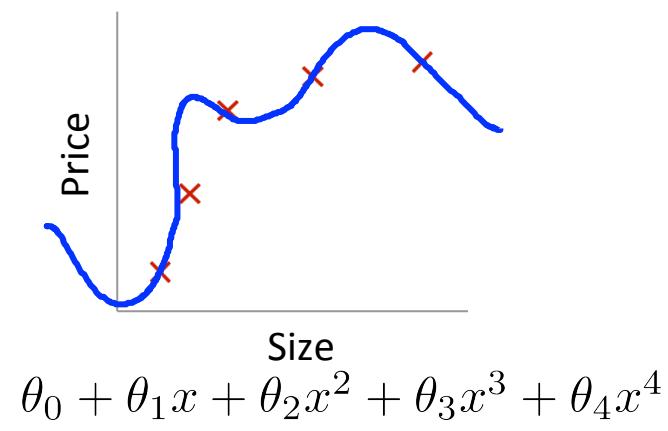
Bias/variance



High bias
(underfit)
 $d=1$



“Just right”
 $d=2$

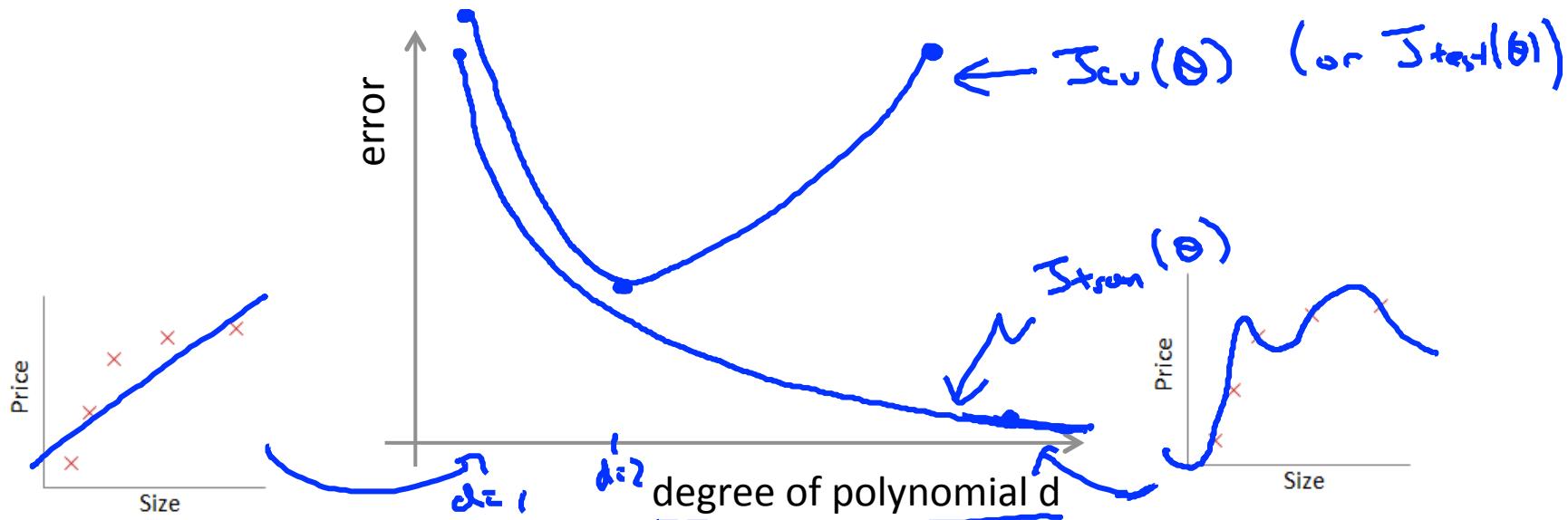


High variance
(overfit)
 $d=4$

Bias/variance

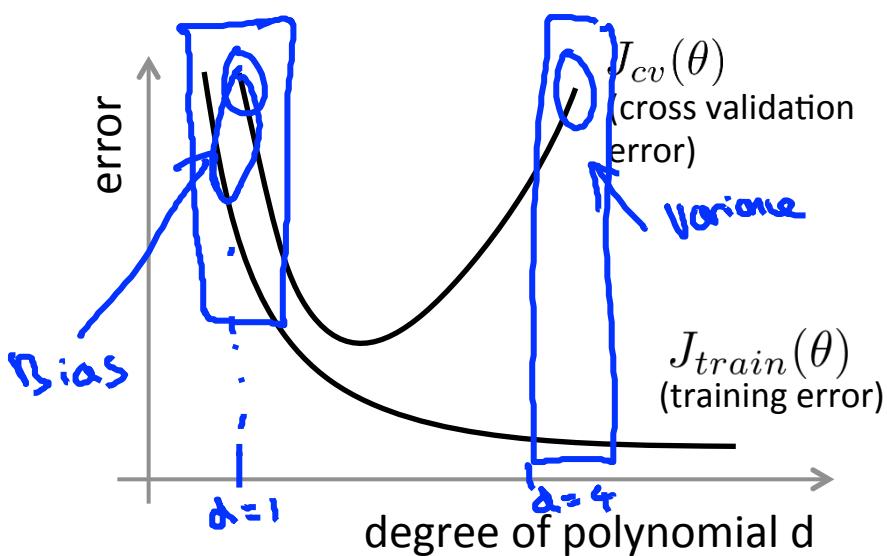
Training error: $\underline{J_{train}(\theta)} = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$

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



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?



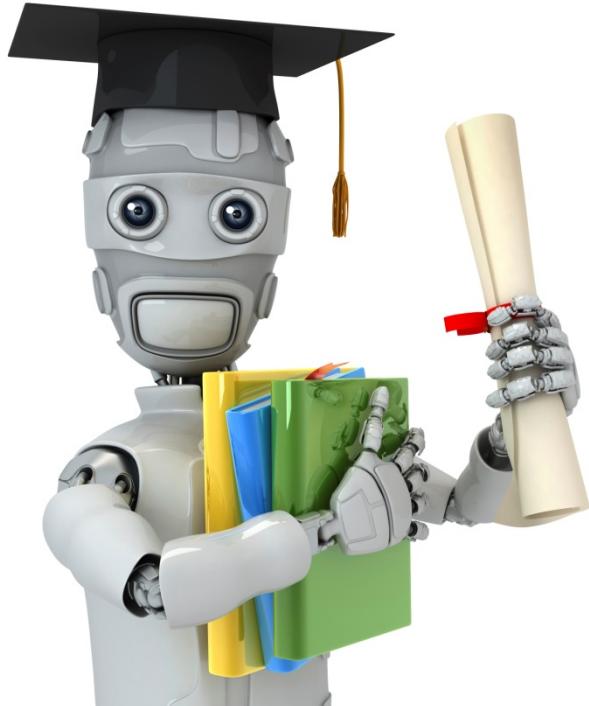
Bias (underfit):

$\rightarrow J_{train}(\theta)$ will be high }
 $J_{cv}(\theta) \approx J_{train}(\theta)$ }

Variance (overfit):

$\rightarrow J_{train}(\theta)$ will be low }
 $J_{cv}(\theta) \gg J_{train}(\theta)$ }

»



Machine Learning

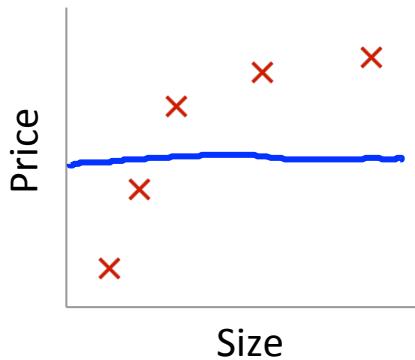
Advice for applying machine learning

Regularization and bias/variance

Linear regression with regularization

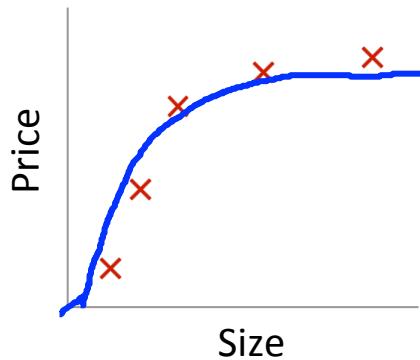
Model:
$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

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

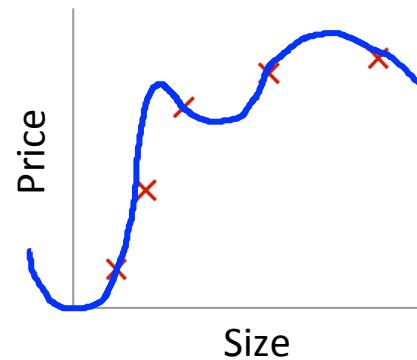


Large λ ←

→ High bias (underfit)
→ $\lambda = 10000$. $\theta_1 \approx 0, \theta_2 \approx 0, \dots$
 $h_{\theta}(x) \approx \theta_0$



Intermediate λ ←
“Just right”



→ Small λ
High variance (overfit)
→ $\lambda = 0$

Choosing the regularization parameter λ

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

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

$$\Rightarrow J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \quad \underbrace{\qquad\qquad\qquad}_{J(\theta)}$$

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

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

J_{train}
 J_{cv}
 J_{test}

Choosing the regularization parameter λ

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

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

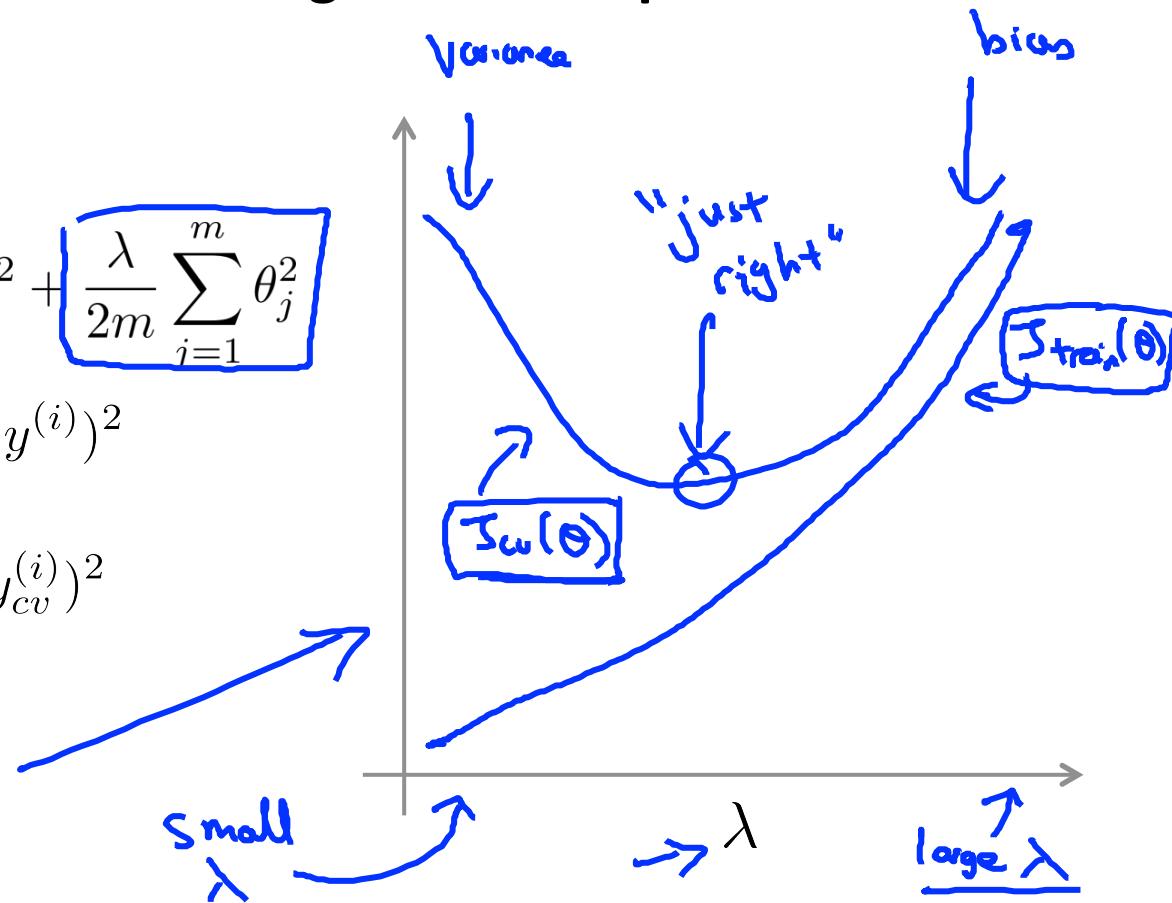
1. Try $\lambda = 0$ $\xrightarrow{\min_{\Theta} J(\Theta)} \Theta^{(0)} \rightarrow J_{cv}(\Theta^{(0)})$
 2. Try $\lambda = 0.01$ $\xrightarrow{\min_{\Theta} J(\Theta)} \Theta^{(1)} \rightarrow J_{cv}(\Theta^{(1)})$
 3. Try $\lambda = 0.02$ $\xrightarrow{\dots} \Theta^{(2)} \rightarrow J_{cv}(\Theta^{(2)})$
 4. Try $\lambda = 0.04$ \vdots
 5. Try $\lambda = 0.08$ \vdots
 6. Try $\lambda = 0.12$
 7. Try $\lambda = 0.16$
 8. Try $\lambda = 0.20$
 9. Try $\lambda = 0.24$
 10. Try $\lambda = 0.28$
 11. Try $\lambda = 0.32$
 12. Try $\lambda = 10$ $\xrightarrow{\Theta^{(12)}} J_{cv}(\Theta^{(12)})$
- Pick (say) $\theta^{(5)}$. Test error: $J_{test}(\Theta^{(5)})$

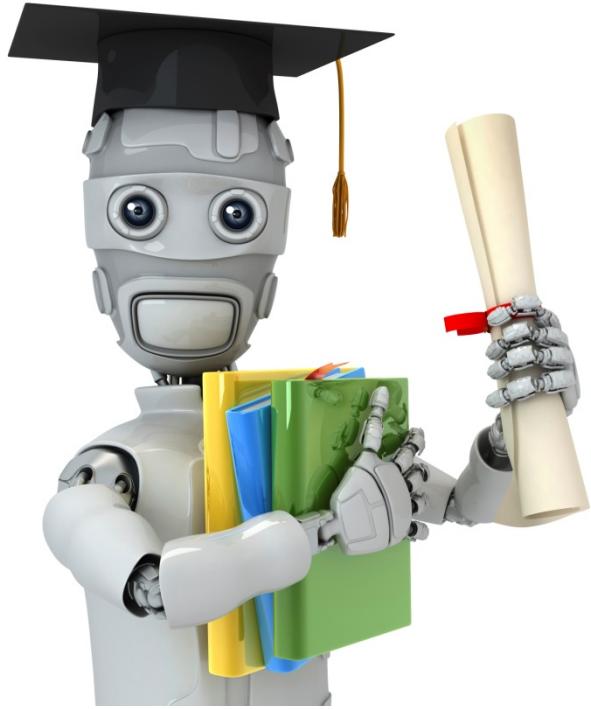
Bias/variance as a function of the regularization parameter λ

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

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

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





Machine Learning

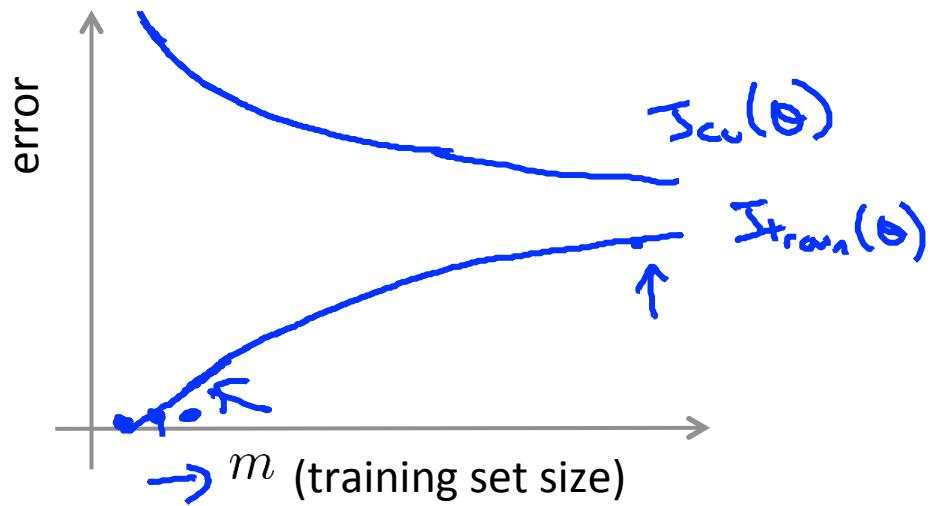
Advice for applying machine learning

Learning curves

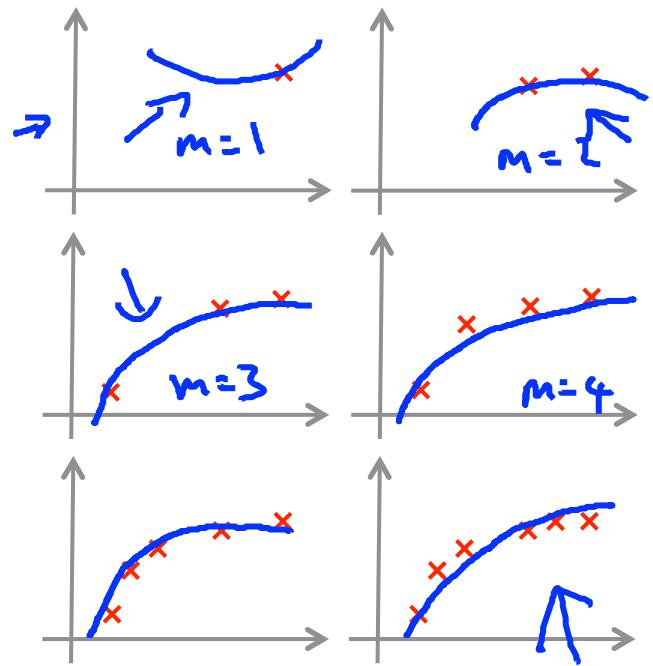
Learning curves

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

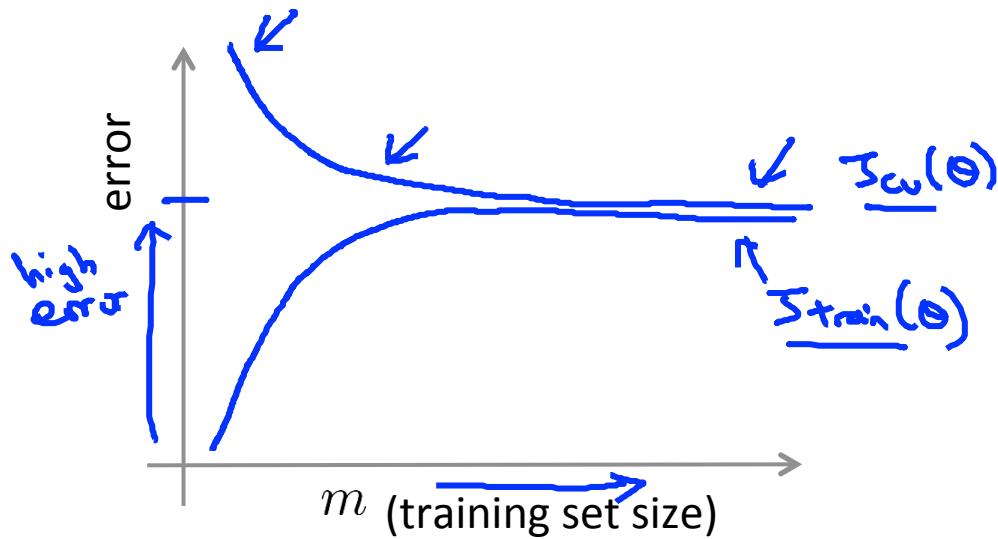
$$\rightarrow J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$



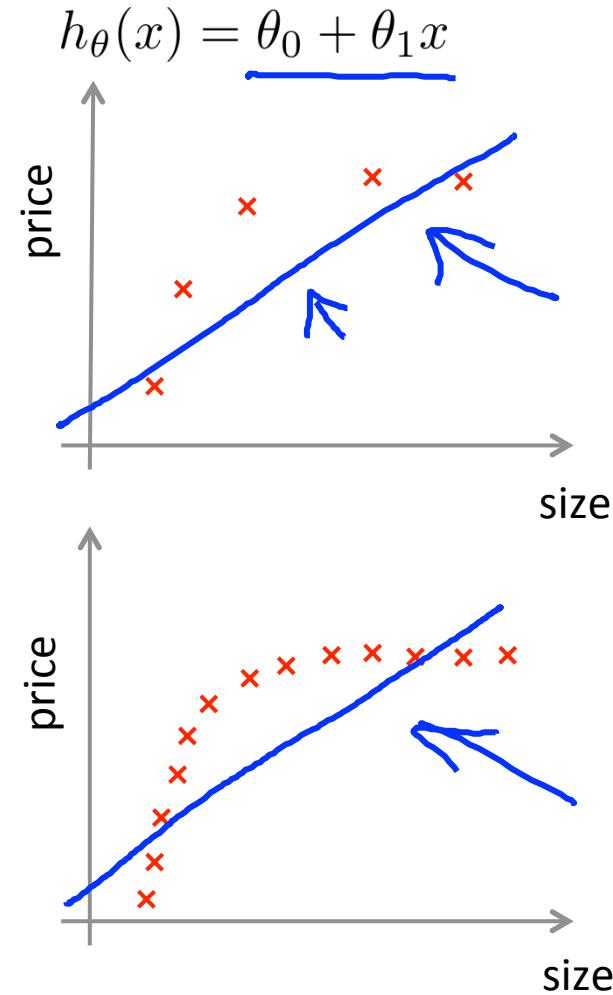
$$h_\theta(x) = \underline{\theta_0 + \theta_1 x + \theta_2 x^2}$$



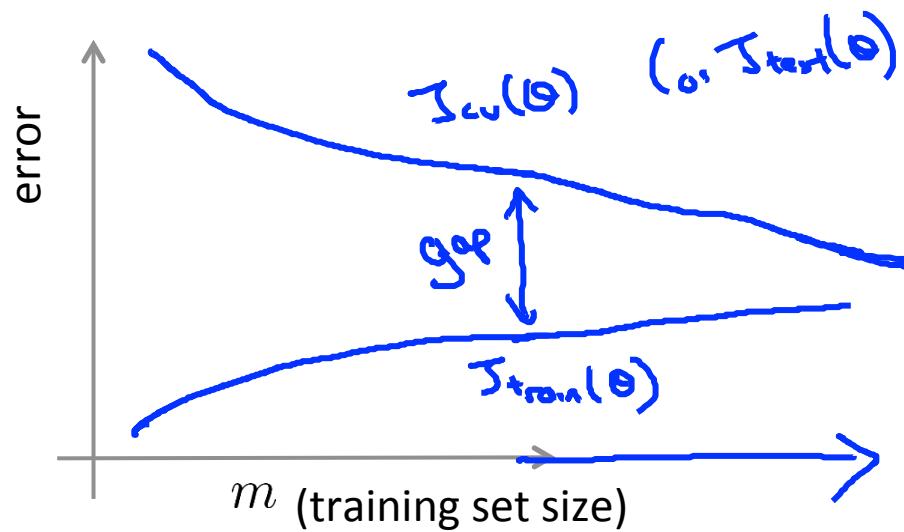
High bias



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



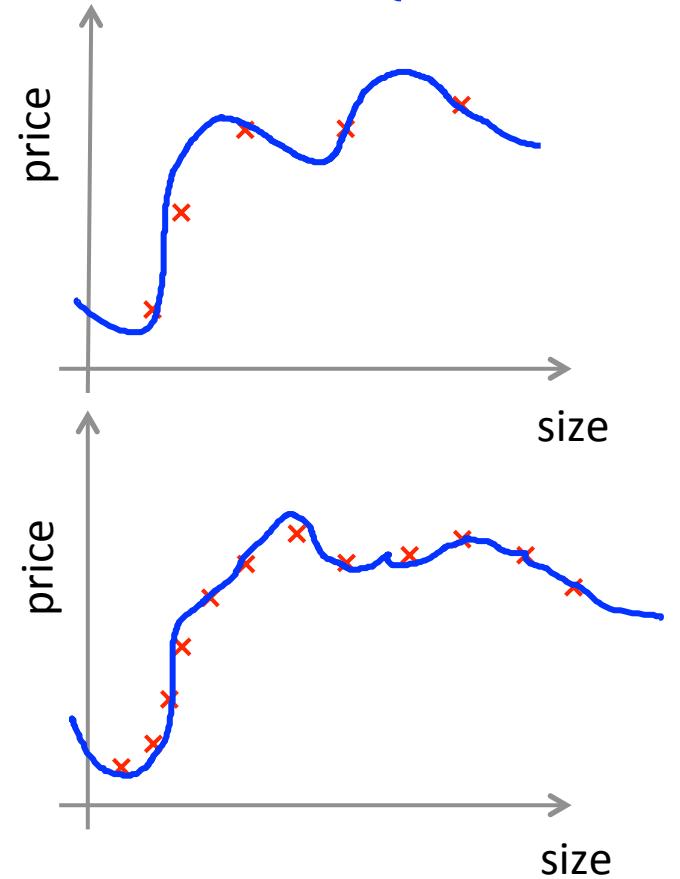
High variance

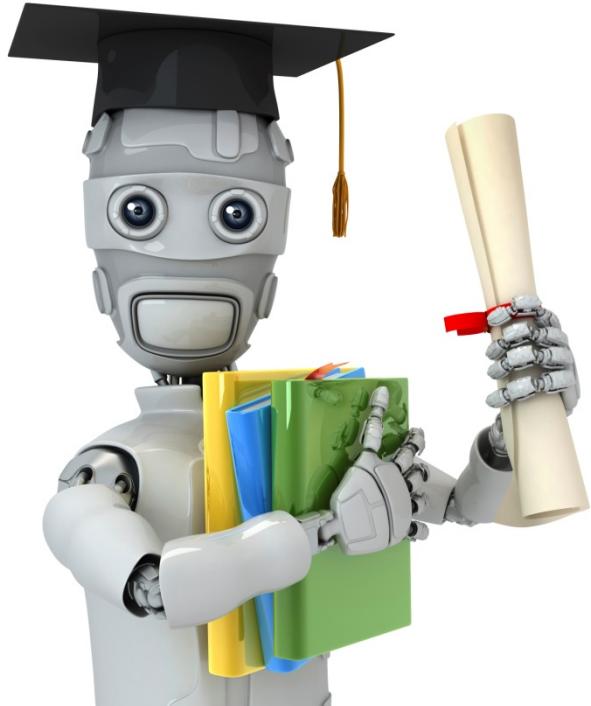


If a learning algorithm is suffering from high variance, getting more training data is likely to help. ↪

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

(and small λ)





Machine Learning

Advice for applying machine learning

Deciding what to try next (revisited)

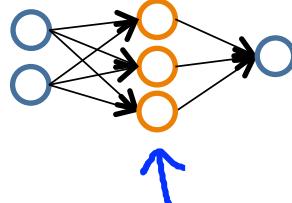
Debugging a learning algorithm:

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 → fixes high variance
- Try smaller sets of features → fixes high variance
- Try getting additional features → fixes high bias
- Try adding polynomial features(x_1^2, x_2^2, x_1x_2 , etc) → fixes high bias.
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance

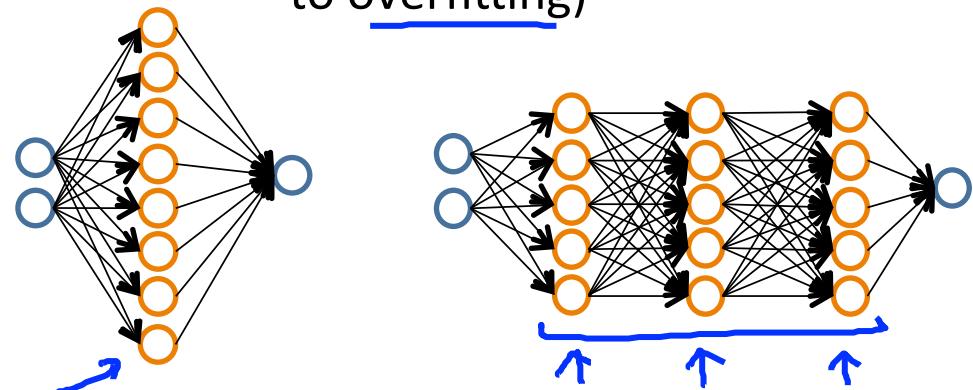
Neural networks and overfitting

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



Computationally cheaper

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

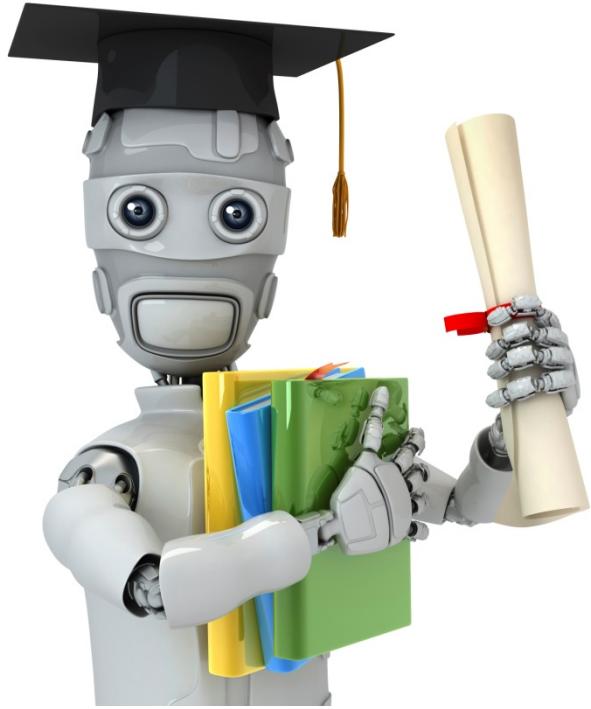


Computationally more expensive.

Use regularization (λ) to address overfitting.

$$\mathcal{J}_{\text{reg}}(\Theta)$$





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.

Spam (1)

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

Non-spam (0)

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.

E.g. deal, buy, discount, andrew, now, ...

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad \begin{array}{l} \text{andrew} \\ \text{buy} \\ \text{deal} \\ \text{discount} \\ \vdots \\ \text{now} \end{array} \quad x \in \mathbb{R}^{100}$$

$$x_j = \begin{cases} 1 & \text{if word } j \text{ appears} \\ 0 & \text{otherwise.} \end{cases}$$

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

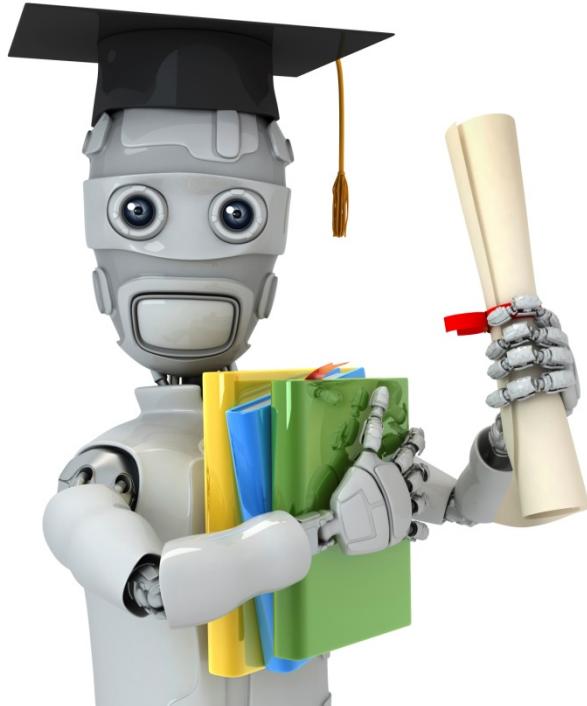
Deal of the week! Buy now!

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 *pharma, replica, steal passwords, ...*
- (ii) What cues (features) you think would have helped the algorithm classify them correctly.

Pharma: 12

Replica/fake: 4

→ Steal passwords: 53

Other: 31

- Deliberate misspellings: 5
(m0rgage, med1cine, etc.)
- Unusual email routing: 16
- Unusual (spamming) punctuation: 32

The importance of numerical evaluation

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

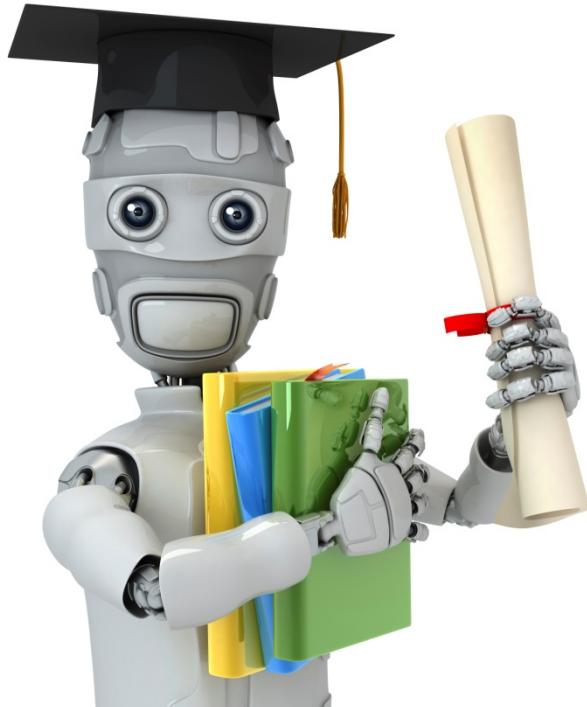
Can use “stemming” software (E.g. “Porter stemmer”) universe/university.

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: 5% error With stemming: 3% error

Distinguish upper vs. lower case (Mom/mom): 3.2%



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.

skewed classes.

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

0.5% error

→ 99.2% accy (0.8% error)

→ 99.5% accy (0.5% error)

Precision/Recall

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

Actual class	
Predicted 1	0
1	True positive False negative
0	False positive True negative

$$y=0 \\ \text{recall} = 0$$

→ Precision

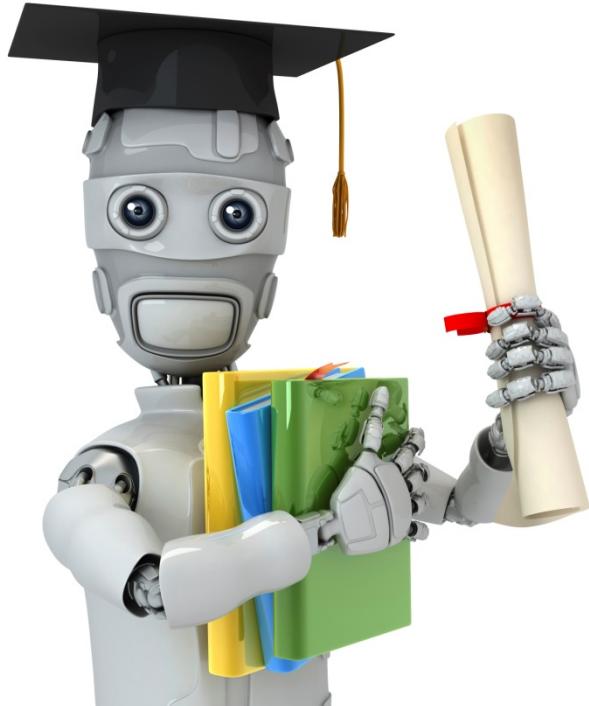
(Of all patients where we predicted $y = 1$, what fraction actually has cancer?)

$$\frac{\text{True positives}}{\#\text{predicted positive}} = \frac{\text{True positive}}{\text{True pos} + \text{False pos}}$$

→ Recall

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

$$\frac{\text{True positives}}{\#\text{actual positives}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}}$$



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 $h_\theta(x) \geq 0.5$ ~~0.7~~ ~~0.9~~ $0.3 \leftarrow$

Predict 0 if $h_\theta(x) < 0.5$ ~~0.7~~ ~~0.9~~ 0.3

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

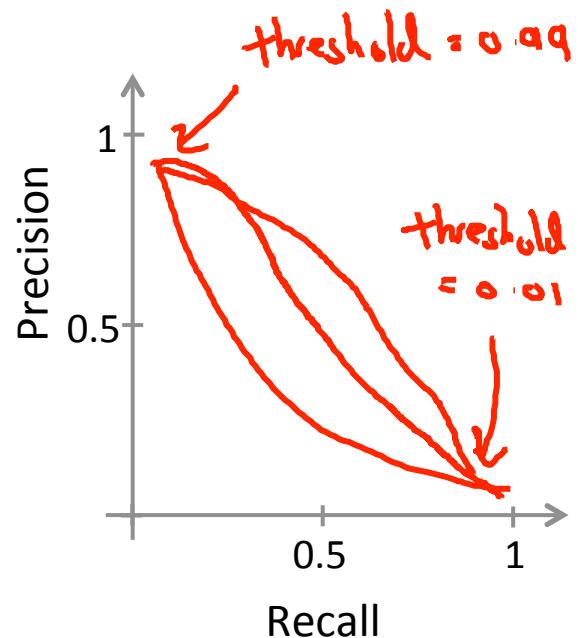
→ Higher precision, lower recall

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

→ Higher recall, lower precision.

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

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



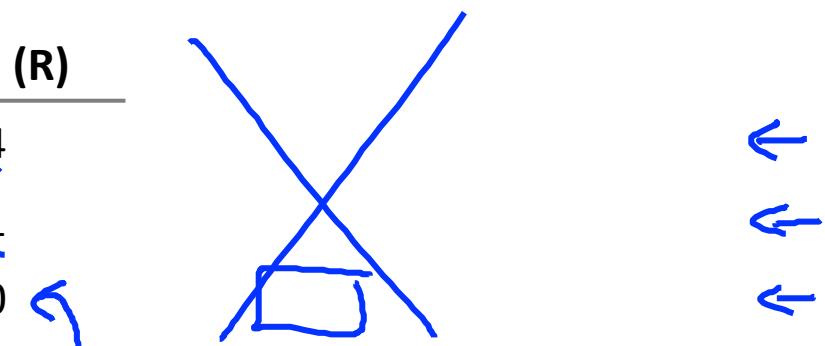
F_1 Score (F score)

How to compare precision/recall numbers?

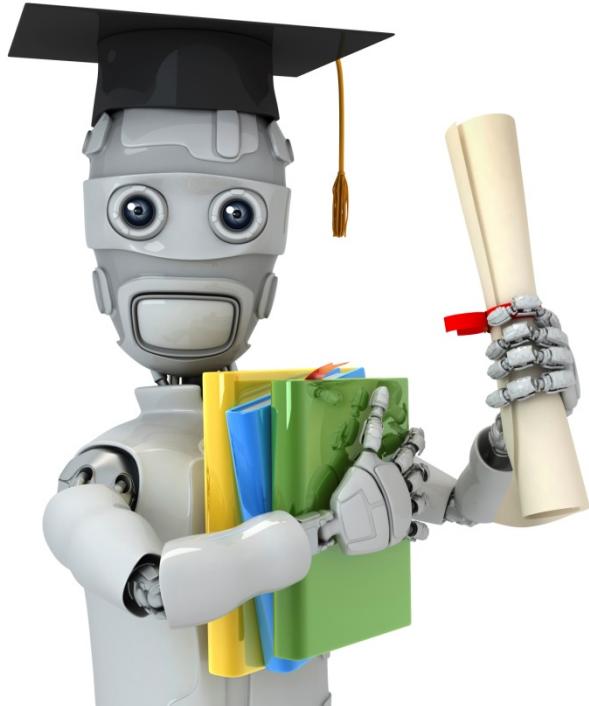
	Precision(P)	Recall (R)
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1.0

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

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



$$\begin{aligned} P=0 \quad \text{or} \quad R=0 &\Rightarrow F\text{-score} = 0 \\ P=1 \quad \text{and} \quad R=1 &\Rightarrow F\text{-score} = 1 \end{aligned}$$



Machine Learning

Machine learning system design

Data for machine learning

Designing a high accuracy learning system

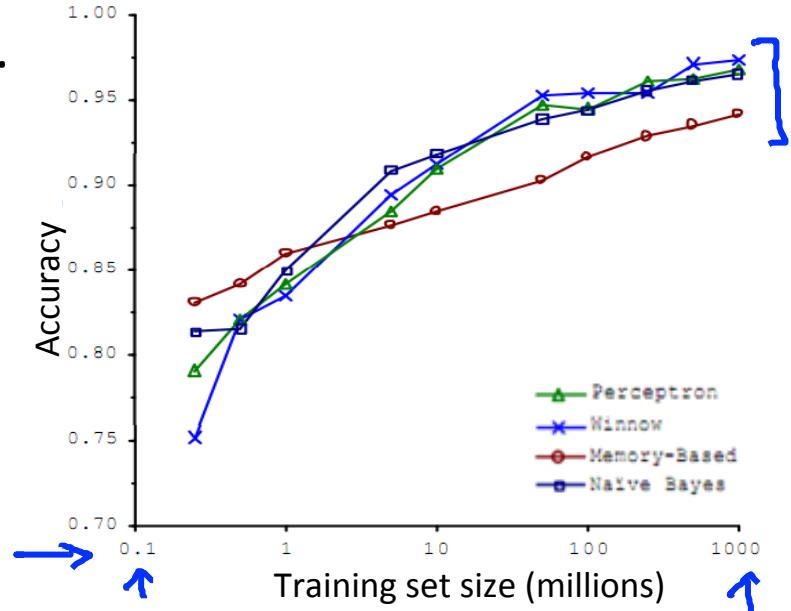
E.g. Classify between confusable words.

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

→ For breakfast I ate two 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.”



[Banko and Brill, 2001]

Large data rationale

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

Example: For breakfast I ate two 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). low bias algorithms. ←

→ $J_{\text{train}}(\theta)$ will be small.

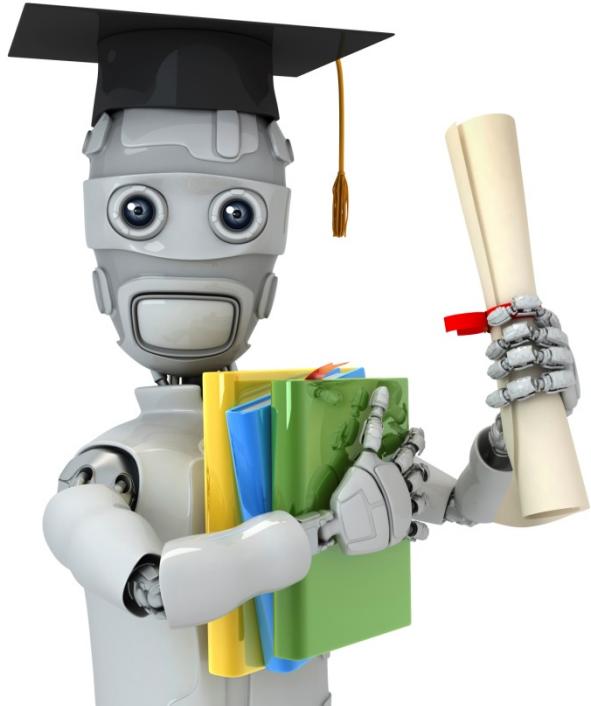
Use a very large training set (unlikely to overfit)

low variance ←

→ $J_{\text{train}}(\theta) \approx J_{\text{test}}(\theta)$

→ $J_{\text{test}}(\theta)$ will be small

Chapter 5 SVM



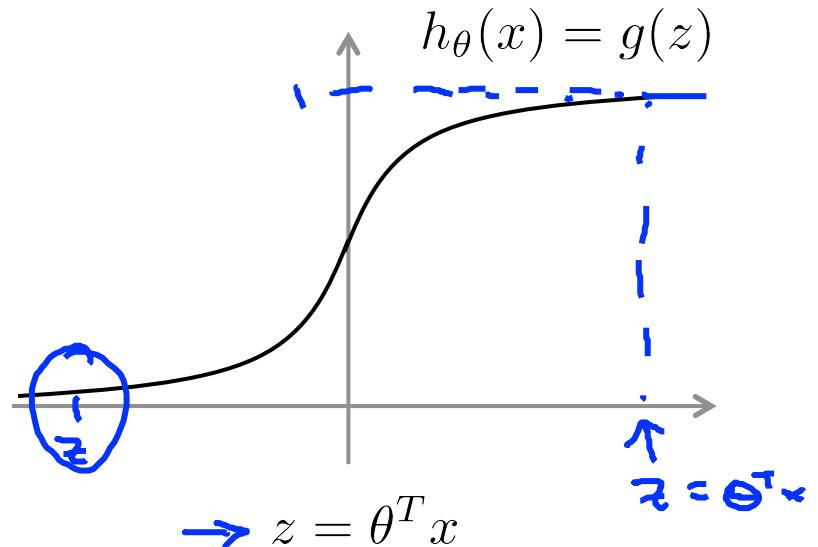
Machine Learning

Support Vector Machines

Optimization objective

Alternative view of logistic regression

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



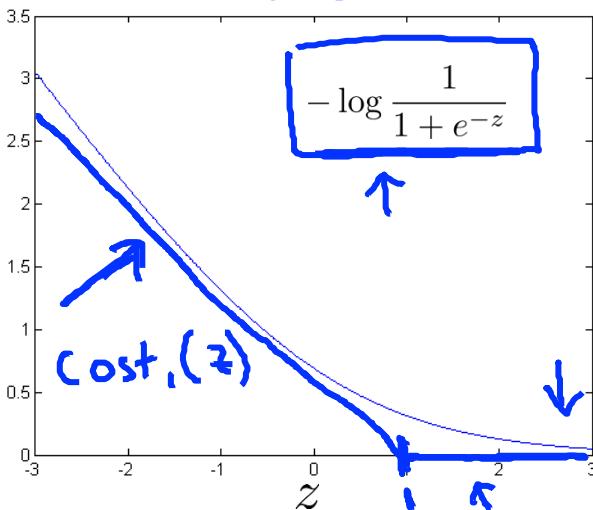
If $y = 1$, we want $h_{\theta}(x) \approx 1$, $\theta^T x \gg 0$
If $y = 0$, we want $h_{\theta}(x) \approx 0$, $\theta^T x \ll 0$

Alternative view of logistic regression

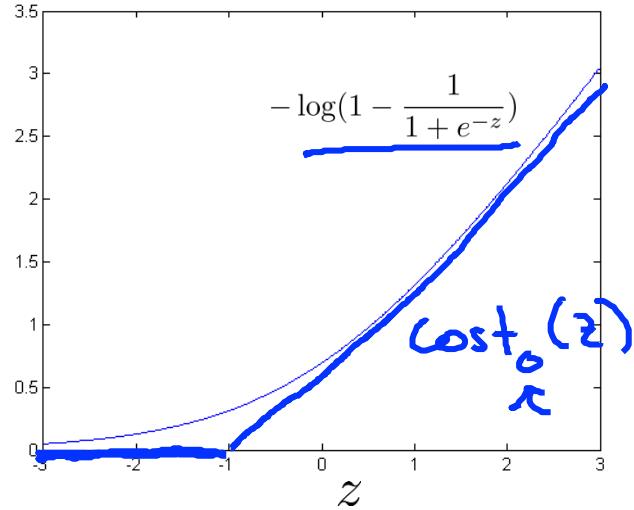
Cost of example: $-(y \log h_\theta(x) + (1 - y) \log(1 - h_\theta(x)))$

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

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



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



Support vector machine

Logistic regression:

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

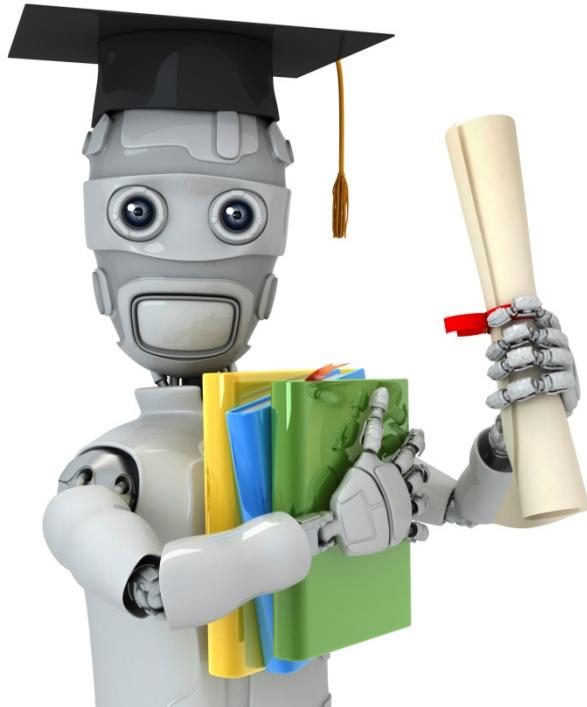
Support vector machine:

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

SVM hypothesis

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

Hypothesis:



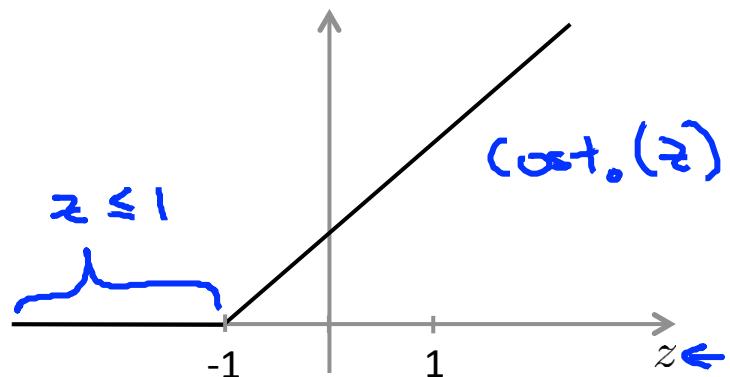
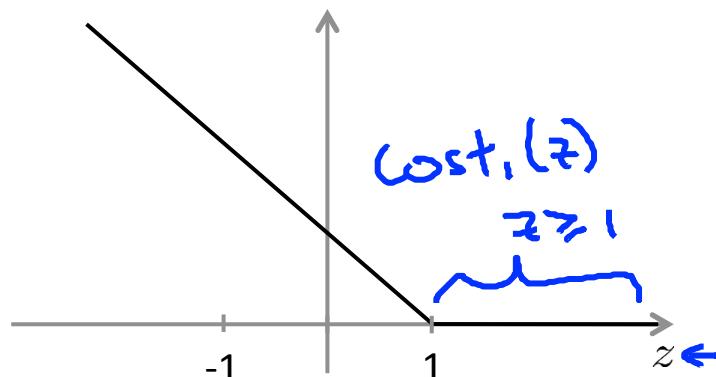
Machine Learning

Support Vector Machines

Large Margin Intuition

Support Vector Machine

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



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

$\theta^T x \geq 1$

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

$\theta^T x \leq -1$

$$C = 100,000$$

SVM Decision Boundary

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

Whenever $y^{(i)} = 1$: $\theta^T x^{(i)} \geq 1$

$$\theta^T x^{(i)} \geq 1$$

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

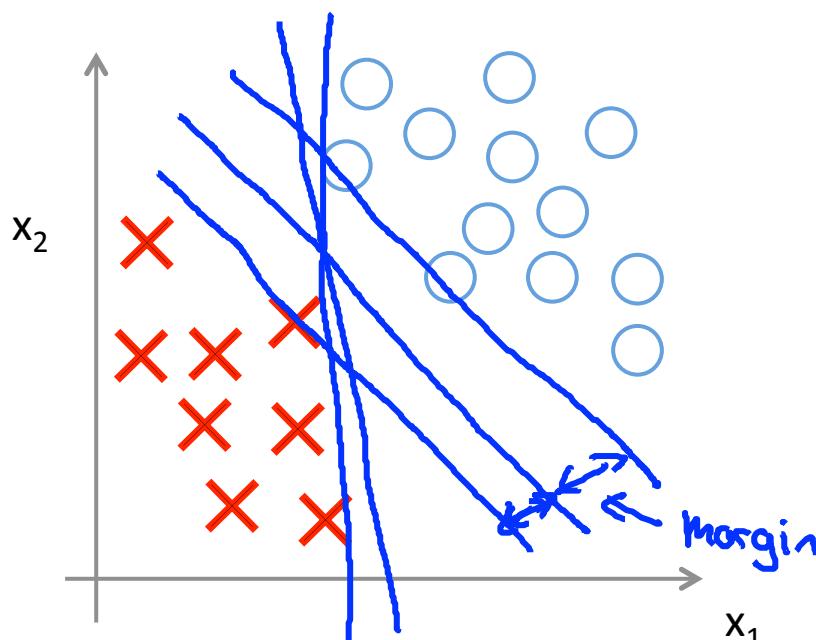
$$\text{s.t. } \theta^T x^{(i)} \geq 1 \quad \text{if } y^{(i)} = 1$$

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

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

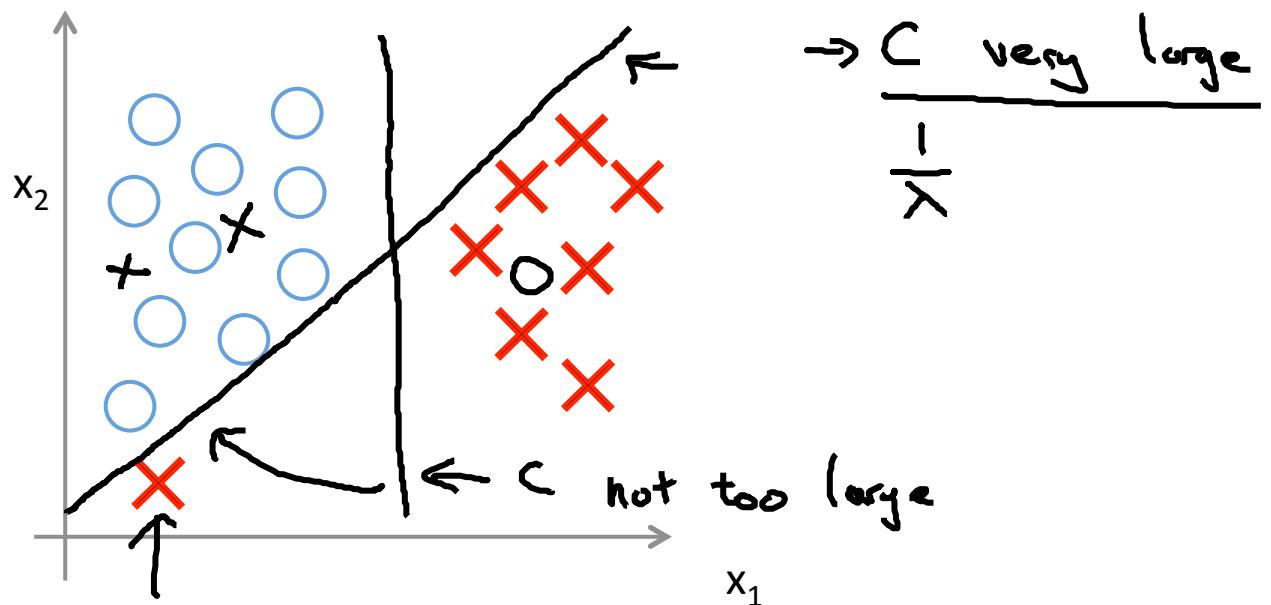
$$\theta^T x^{(i)} \leq -1$$

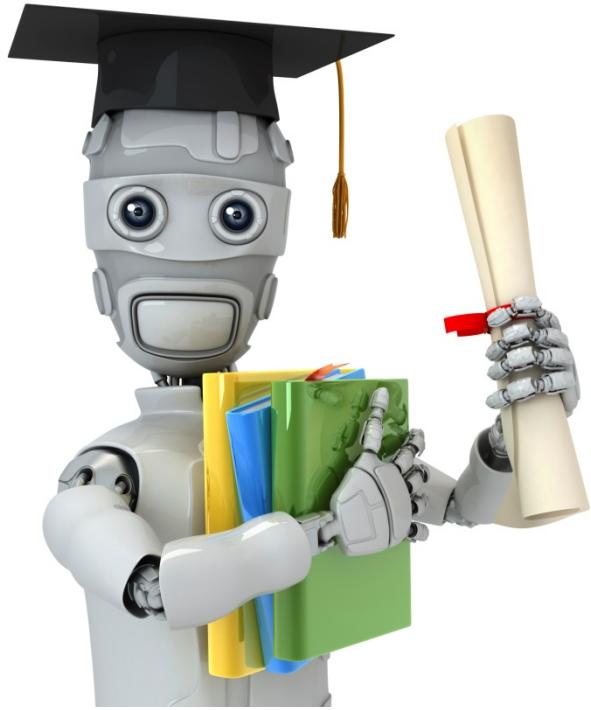
SVM Decision Boundary: Linearly separable case



Large margin classifier

Large margin classifier in presence of outliers



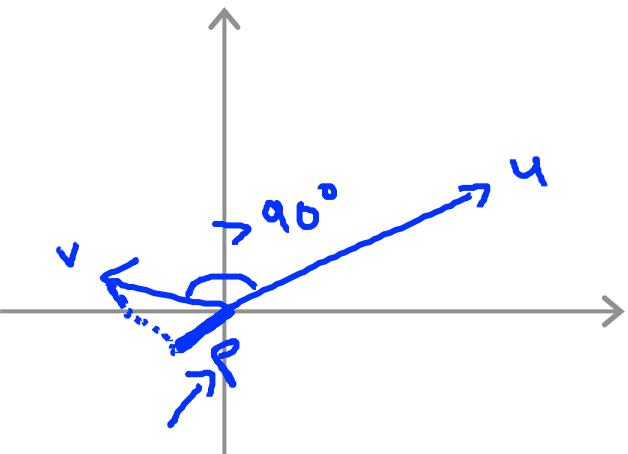
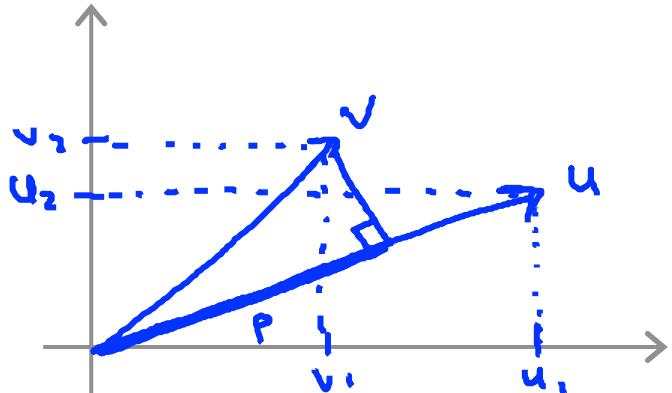


Machine Learning

Support Vector Machines

The mathematics
behind large margin
classification (optional)

Vector Inner Product



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

$$u^T v = ? \quad [u_1 \ u_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$\|u\| = \text{length of vector } u \\ = \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$$

$p = \text{length of projection of } v \text{ onto } u.$

Signed

$$u^T v = \underline{p} \cdot \underline{\|u\|} \leftarrow = v^T u \\ = u_1 v_1 + u_2 v_2 \leftarrow p \in \mathbb{R}$$

$$u^T v = p \cdot \|u\|$$

$$p < 0$$

$$\omega = (\sqrt{\omega})^2$$

SVM Decision Boundary

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} (\theta_1^2 + \theta_2^2) = \frac{1}{2} (\boxed{\theta_1^2 + \theta_2^2})^2 = \frac{1}{2} \|\theta\|^2$$

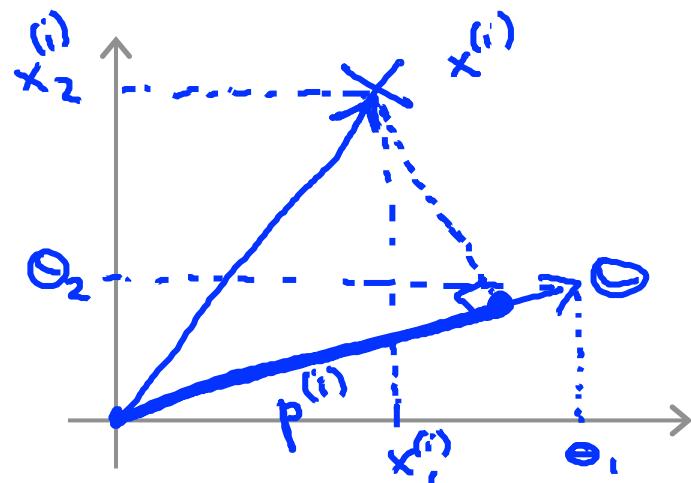
s.t. $\boxed{\theta^T x^{(i)} \geq 1}$ if $y^{(i)} = 1$
 $\rightarrow \theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}, \theta_0 = 0$$

Simplification: $\underline{\theta_0 = 0}$, $\underline{n=2}$

$$\theta^T x^{(i)} = ?$$

\uparrow
 $U^T V$



$$\theta^T x^{(i)} = \boxed{P^{(i)} \|\theta\|} \leftarrow$$

$$= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} \leftarrow$$

SVM Decision Boundary

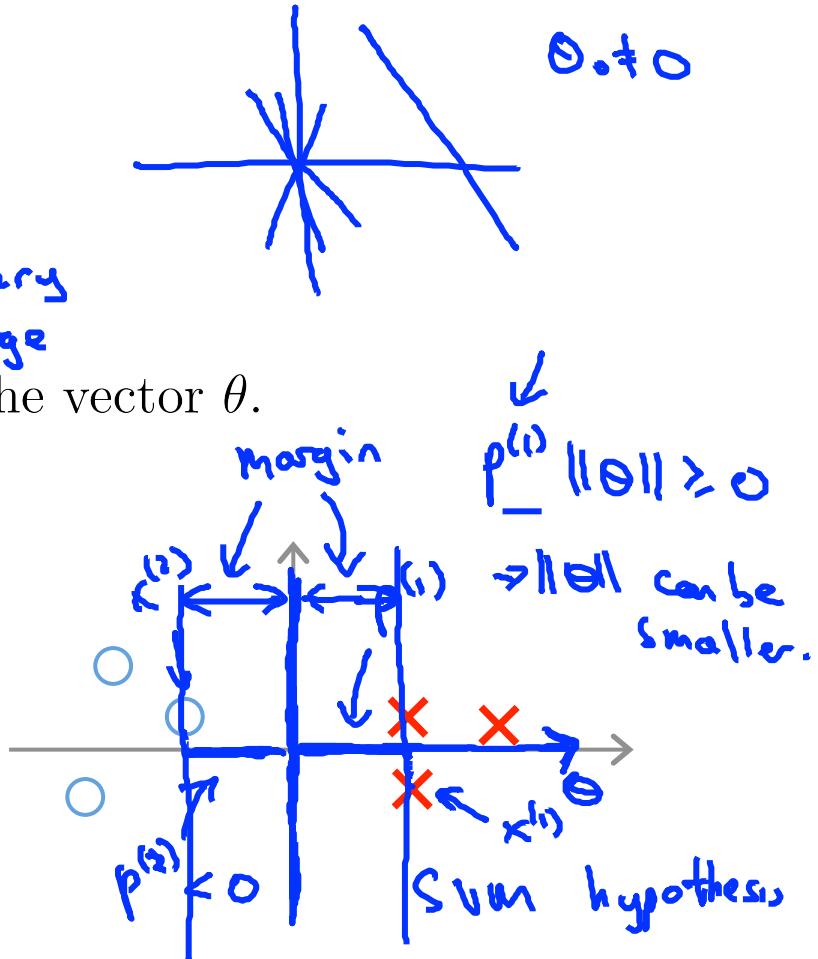
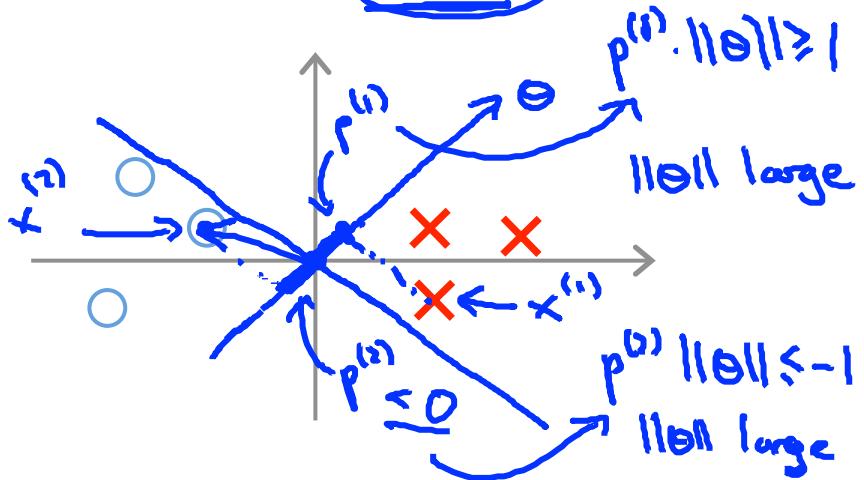
$$\Rightarrow \min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 = \frac{1}{2} \|\theta\|^2 \leftarrow$$

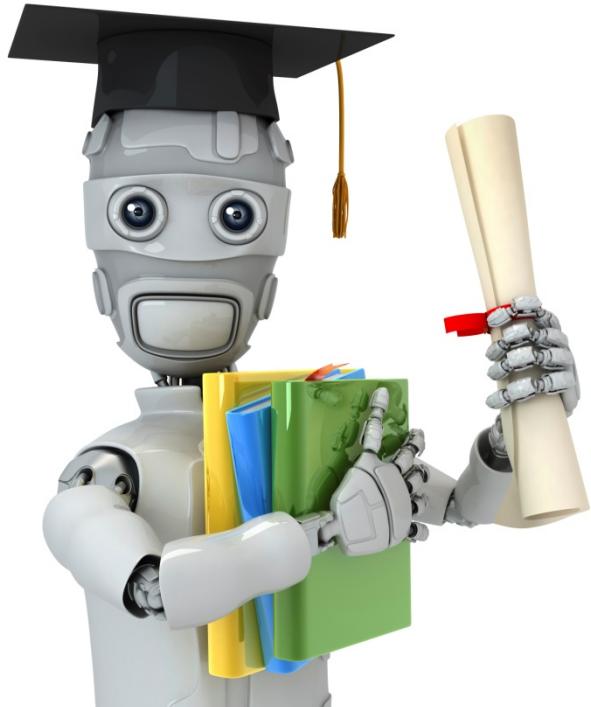
s.t. $\boxed{p^{(i)} \cdot \|\theta\| \geq 1}$ if $y^{(i)} = 1$

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

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

Simplification: $\theta_0 = 0$



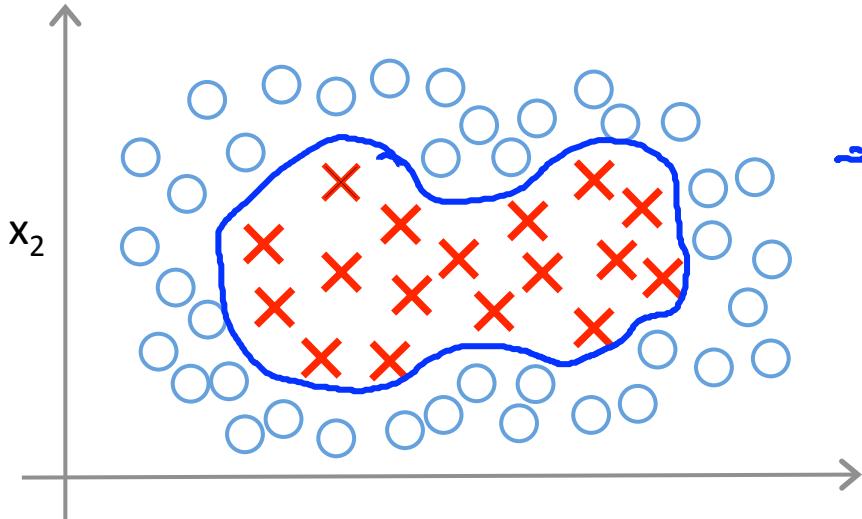


Machine Learning

Support Vector Machines

Kernels I

Non-linear Decision Boundary



Predict $y = 1$ if

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

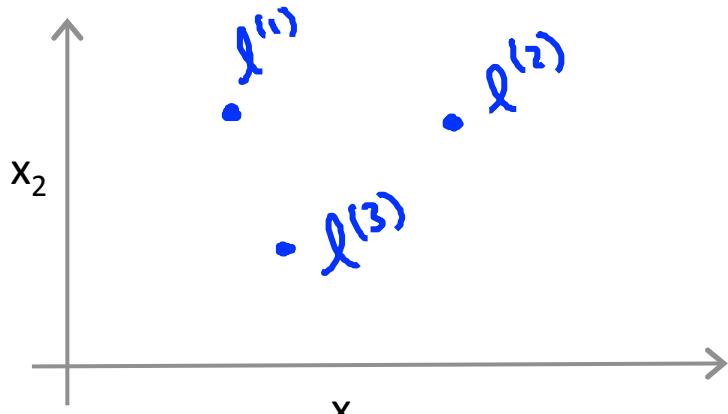
$$h_{\theta}(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \dots \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

$$\rightarrow \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$$

$$f_1 = x_1, \quad f_2 = x_2, \quad f_3 = x_1 x_2, \quad f_4 = x_1^2, \quad f_5 = x_2^2, \dots$$

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 $l^{(1)}, l^{(2)}, l^{(3)}$

Given x :

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$
$$f_2 = \text{similarity}(x, l^{(2)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$$
$$f_3 = \text{similarity}(x, l^{(3)}) = \exp(\dots)$$

\nwarrow Kernel (Gaussian kernels) $\nearrow k(x, l^{(1)})$

Kernels and Similarity

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

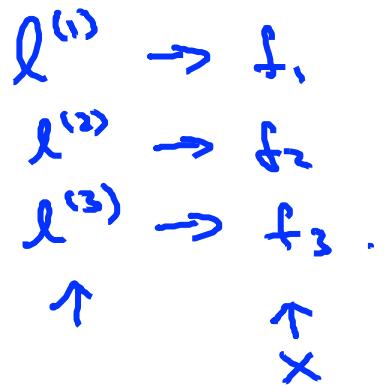


If $x \approx l^{(1)}$:

$$f_1 \underset{\uparrow}{\approx} \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$$

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

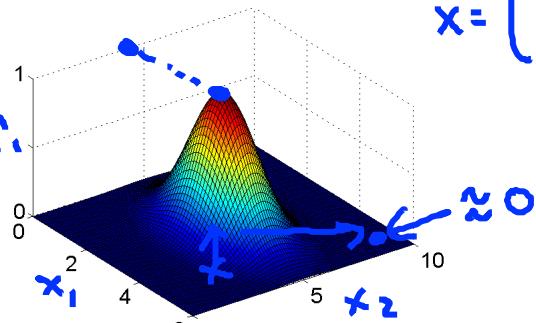
$$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0.$$



Example:

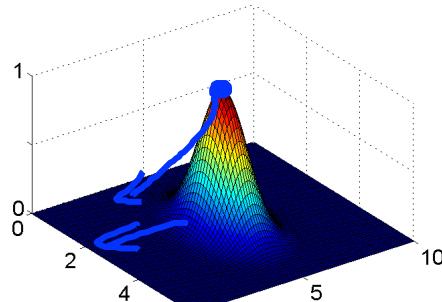
$$\rightarrow l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

$$\rightarrow \sigma^2 = 1$$

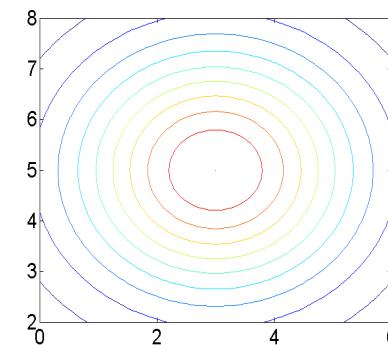
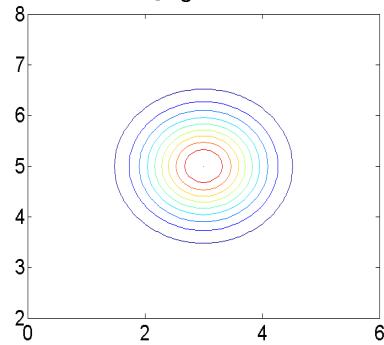
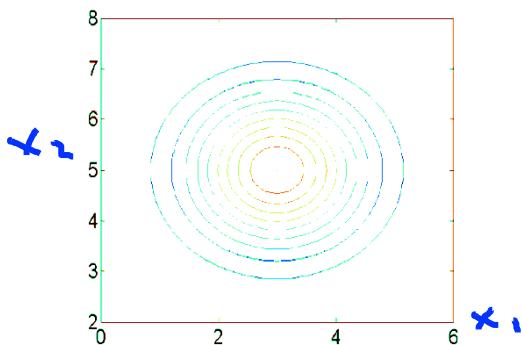
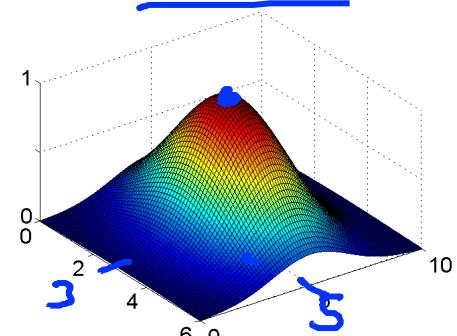


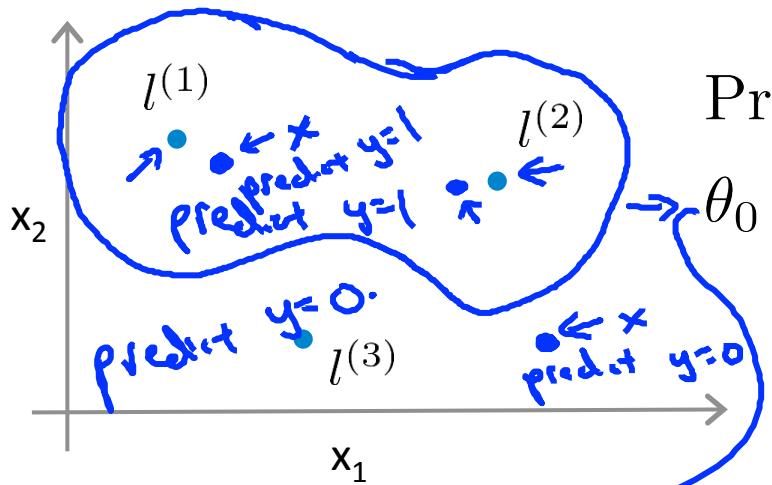
$$x = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

$$\sigma^2 = 0.5$$



$$\sigma^2 = 3$$





Predict "1" when

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$$



$$\underline{\theta_0 = -0.5, \theta_1 = 1, \theta_2 = 1, \theta_3 = 0}$$

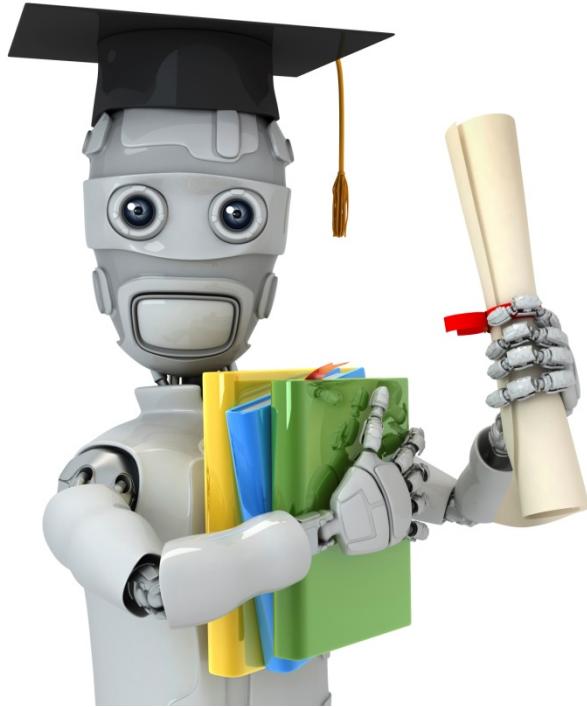
$$f_1 \approx 1, f_2 \approx 0, f_3 \approx 0.$$

$$\theta_0 + \theta_1 \cdot 1 + \theta_2 \cdot 0 + \theta_3 \cdot 0$$

$$= -0.5 + 1 = 0.5 \geq 0$$

$$f_1, f_2, f_3 \approx 0$$

$$\rightarrow \underline{\theta_0 + \theta_1 f_1 + \dots} \approx -0.5 < 0$$

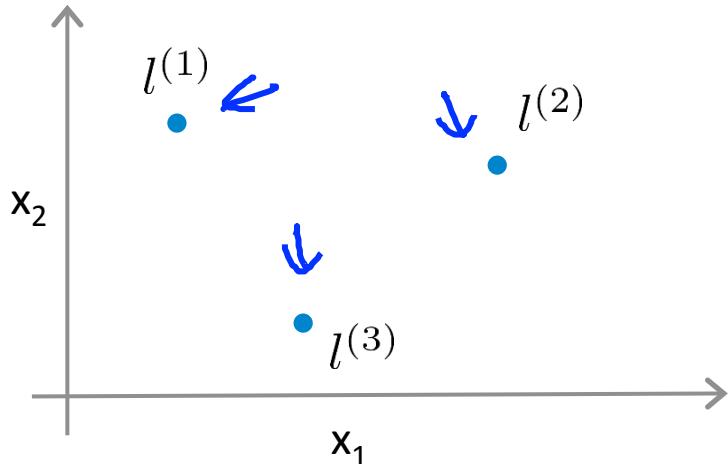


Machine Learning

Support Vector Machines

Kernels II

Choosing the landmarks

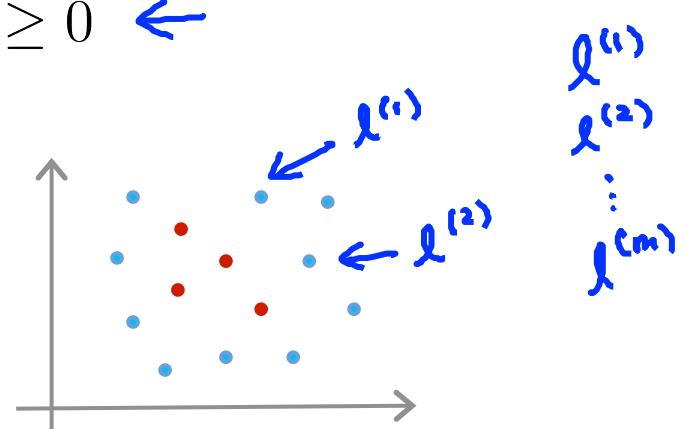
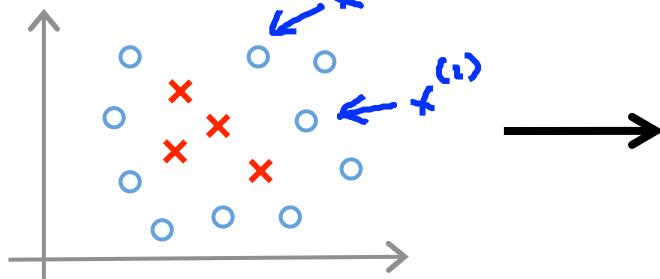


Given x :

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

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 \underline{x} :

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

$$f = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1$$

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

$$\begin{aligned} \underline{x}^{(i)} \rightarrow & \begin{bmatrix} f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} = \begin{bmatrix} \text{sim}(x^{(i)}, l^{(1)}) \\ \text{sim}(x^{(i)}, l^{(2)}) \\ \vdots \\ \text{sim}(x^{(i)}, l^{(m)}) \end{bmatrix} \\ & f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = \exp\left(-\frac{\|x^{(i)} - l^{(i)}\|^2}{2\sigma^2}\right) = 1 \end{aligned}$$

$$\begin{aligned} &x^{(i)} \in \mathbb{R}^{n+1} \quad (\text{or } \mathbb{R}^n) \\ &\rightarrow f^{(i)} = \begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix} \\ &f_0^{(i)} = 1 \end{aligned}$$

Andrew Ng

SVM with Kernels

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

→ Predict "y=1" if $\underline{\theta}^T \underline{f} \geq 0$



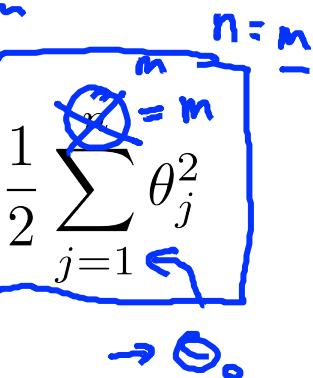
$$\underline{\theta} \in \mathbb{R}^{m+1}$$

$$\theta_0 f_0 + \theta_1 f_1 + \dots + \theta_m f_m$$

Training:

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

$$\cancel{\theta^T f^{(i)}} \quad \theta^T f^{(i)}$$



$$\begin{bmatrix} - & \sum_j \theta_j^2 \\ - & \end{bmatrix} = \theta^T \theta \quad \leftarrow \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_m \end{bmatrix} \quad (\text{ignoring } \theta_0) \\ \rightarrow \underline{\theta^T M \theta} \quad \leftarrow \| \theta \|^2$$

$$m = 10,000$$

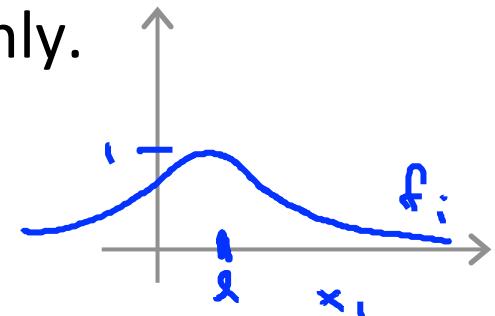
SVM parameters:

$C \left(= \frac{1}{\lambda} \right)$. \rightarrow Large C: Lower bias, high variance. (small λ)
 \rightarrow Small C: Higher bias, low variance. (large λ)

σ^2 Large σ^2 : Features f_i vary more smoothly.

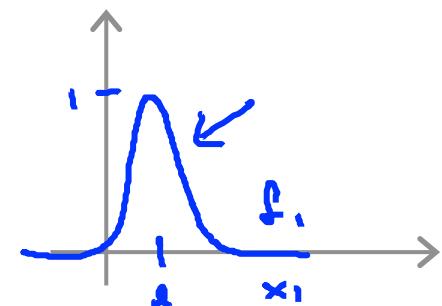
\rightarrow Higher bias, lower variance.

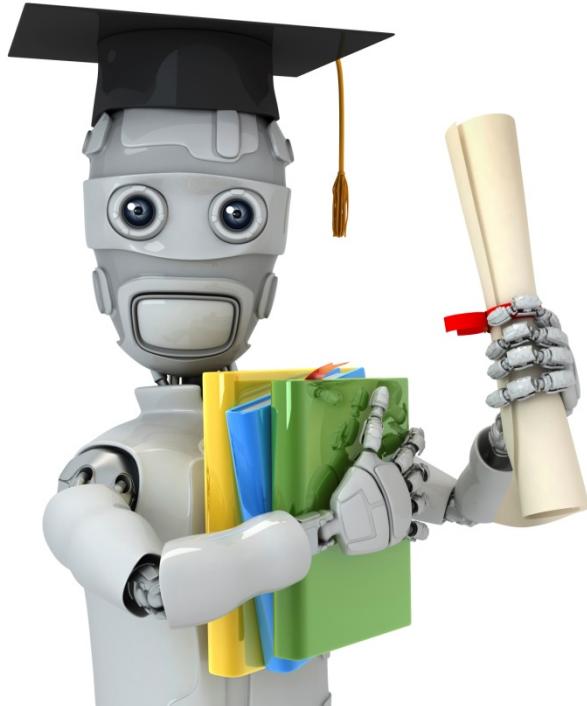
$$\exp \left(- \frac{\|x - \mu_i\|^2}{2\sigma^2} \right)$$



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 $\underline{\theta^T x} \geq 0$

$$\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n \geq 0$$

→ n large, m small $x \in \mathbb{R}^{n+1}$

→ Gaussian kernel:

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

Need to choose $\frac{\sigma^2}{\tau}$.

$x \in \mathbb{R}^n$, n small
and/or m large



Kernel (similarity) functions:

function $f = \text{kernel}(x_1, x_2)$

$$f = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

return

$x^{(i)}$

$\lambda^{(j)} = x^{(j)}$

$x \rightarrow$
 f_1
 f_2
 \vdots
 f_m

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

$$\begin{aligned} & \boxed{\|x - l\|^2} \quad x \in \mathbb{R}^{n+1} \\ & \|v\|^2 = v_1^2 + v_2^2 + \dots + v_n^2 \\ & = \underbrace{(x_1 - l_1)^2 + (x_2 - l_2)^2 + \dots + (x_n - l_n)^2}_{\substack{1-5 \text{ bedrooms}}} \end{aligned}$$

Other choices of kernel

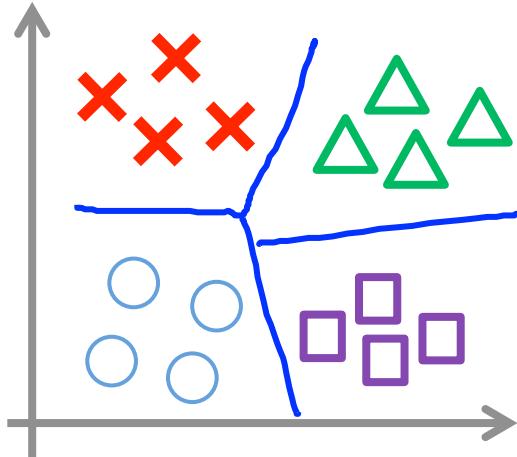
Note: Not all similarity functions $\text{similarity}(x, l)$ make valid kernels.

→ (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: $k(x, l) = \underbrace{(x^T l)}_1 + \underbrace{(x^T l)^2}_2 + \underbrace{(x^T l + 1)^3}_3 + \underbrace{(x^T l + 5)^4}_4$
- More esoteric: String kernel, chi-square kernel, histogram intersection kernel, ...
 $\text{sim}(x, l)$

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 $\theta^{(1)}, \theta^{(2)}, \dots, \underline{\theta^{(K)}}$
Pick class i with largest $(\underline{\theta^{(i)}})^T x$

$$\begin{array}{c} \uparrow \\ y=1 \\ \nwarrow \\ y=2 \\ \dots \\ \uparrow \\ \underline{\theta^{(K)}} \end{array}$$

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): (e.g. $n \geq m$, $n = \underline{10,000}$, $m = \underline{10 \dots 1000}$)

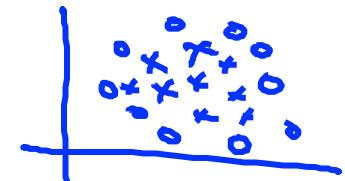
→ Use logistic regression, or SVM without a kernel ("linear kernel")

→ If n is small, m is intermediate: ($n = \underline{1-1000}$, $m = \underline{10 - 10,000}$) ←

→ Use SVM with Gaussian kernel

If n is small, m is large: ($n = \underline{1-1000}$, $m = \underline{50,000+}$)

→ 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.